RM0387
Reference manual

SPC574Sx 32-bit Power Architecture® microcontroller for automotive ASILD applications

Overview

This document defines the functionality of the SPC574Sx microcontroller for use by software and hardware developers. The SPC574Sx is built on Power Architecture® technology and integrates technologies that are important for today's and tomorrow's automotive applications.

SPC574Sx targets Chassis & Safety applications such as EPS (Electric Power Steering), ESC (Electronic Stability Control), Dual 3-phase brushless DC FOC (Field Oriented Control) motor controls, Hybrid drive, DC-DC converters/inverters and high-end airbag applications which require a high safety integrity level.

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

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

Audience

This manual is intended for system software and hardware developers and applications programmers who want to develop products with the SPC574Sx device. It is assumed that the reader understands operating systems, microprocessor system design, basic principles of software and hardware, and basic details of the Power Architecture.

Contents

1 Preface 45

1.1 Register conventions 45

1.2 Acronyms and abbreviations 45

2 Introduction 46

2.1 SPC574Sx microcontroller 46

2.1.1 Target applications 46

2.1.2 Overall architecture 46

2.1.3 Core features 46

2.1.4 Memory hierarchy 47

2.2 Features 48

2.3 Feature summary 50

2.4 Packages 51

2.5 Block diagram 51

3 System architecture 53

3.1 Platform Architecture 53

3.1.1 XBAR Masters 53

3.1.2 Functionality 55

3.1.3 Performances 56

3.2 Embedded memories 58

3.2.1 SRAM 58

3.2.2 Flash memory 59

3.2.3 End-to-end Error Correction Code (e2eECC) 61

3.2.4 Security features 63

3.3 Motor control Architecture 64

4 Signal description 73

4.1 Pin descriptions 73

4.1.1 Pin startup and reset states 73

4.1.2 Power supply and reference voltage pins 73

4.1.3 System pins 75

4.1.4 Debug and trace pins 76

4.1.5 Pin multiplexing 76

4.2 Package pads/pins 180

5 Memory map 185

5.1 Architecture 185

5.2 Memory map 185

6 Functional safety 195

6.1 Introduction 195

6.2 Assumptions and usage constraints 195

6.3 Module categorization 196

6.4 Safety architecture overview 197

6.4.1 Lockstep 198

6.4.2 End to end (E2E) protection on the data path and memory ECC 200

6.4.3 Additional address coverage 201

6.4.4 I/O safety 218

6.4.5 Clock 218

6.4.6 Power 218

6.4.7 Failure handling 218

6.4.8 Failure indication 237

6.4.9 Latent fault detection 237

6.4.10 Common Cause Failures (CCFs) 238

6.4.11 Resets 238

6.4.12 LBIST / MBIST architecture 239

6.5 Register classifications 241

7 Device configuration 242

7.1 Introduction 242

7.2 Core modules 242

7.2.1 Core Reset Settings 242

7.2.2 Special Purpose Register Summary 244

7.3 System modules 247

7.3.1 SIUL2 configuration 247

7.3.2 Crossbar switch 247

7.3.3 System Memory Protection Unit (SMPU) configuration 249

7.3.4 Peripheral bridge configuration 251

7.3.5 Interrupt controller (INTC) configuration 252

7.3.6 DMA controller configuration 260
7.3.7 DMA channel multiplexer (DMACHMUX) configuration 261
7.3.8 Wakeup unit (WKPU) configuration 268
7.3.9 Crossbar integrity checker (XBIC) 269
7.4 Clocking 269
7.5 Memories and memory interfaces 269
7.5.1 Flash memory controller (PFLASH) configuration 269
7.5.2 Decorated Storage Memory Controller (DSMC) 270
7.6 Analog modules 271
7.6.1 Temperature sensor configuration 271
7.6.2 SARADC 271
7.7 Motor control modules 319
7.7.1 ECTU 319
7.7.2 eTimer 321
7.7.3 FlexPWM 321
7.8 Timers 333
7.8.1 System Timer Module (STM) configuration 333
7.8.2 Software Watchdog Timer (SWT) configuration 334
7.8.3 Periodic Interupt Timer (PIT) configuration 334
7.9 Communication interfaces 335
7.9.1 CAN subsystem configuration 336
7.9.2 DSPI configuration 336
7.9.3 FlexRay Configuration 338
7.9.4 SENT Receiver (SRX) configuration 339
7.9.5 LinFlexD 341
7.10 Reset and Boot modules 343
7.10.1 BAF configuration 343
7.10.2 System Status and Configuration Module (SSCM) configuration 344
7.11 Safety modules 345
7.11.1 CRC – Number of contexts 345
7.11.2 MEMU configuration 345
7.11.3 FCCU configuration 353
7.11.4 Register protection (REG_PROT) configuration 354
7.11.5 Password and Device Security Module (PASS) configuration 363
7.11.6 STCU2 configuration 367
7.12 Debug and trace 385

7.12.1 NPC configuration 385

8 Reset and boot 387

8.1 Introduction 387

8.1.1 Architecture 387

8.1.2 Functionality 387

8.2 Hardware reset stage 388

8.2.1 Destructive reset events 388

8.2.2 Functional reset sources 389

8.2.3 Functional reset escalation 390

8.2.4 Destructive reset escalation 391

8.3 Reset sequence 391

8.4 PORST pin functionality 394

8.5 ESR0 pin functionality 396

8.6 PORST and ESR0 pin status during power on, destructive and functional reset 397

8.7 System boot 397

8.8 HW boot record search (SSCM search) 398

8.9 SW boot record search (BAF search) 400

8.10 Performances 401

9 Device Configuration Format (DCF) Records 402

9.1 Introduction 402

9.2 DCF clients 402

9.3 DCF records 403

9.3.1 UTEST DCF records 406

9.4 DCF client table 406

10 Power management 418

10.1 Overview 418

10.1.1 Power management framework 418

10.1.2 Power management supply description 419

10.2 Architecture 420

10.2.1 Start-up sequence 423

10.2.2 Voltage regulator mode 423

10.2.3 Ballast 423

10.2.4 Reduced power modes 424
10.2.5 Voltage monitors 425
10.2.6 Device operating status versus power conditions 430
10.2.7 BIST details 432

11 Security 436
11.1 Basic security 436
11.2 Advanced security 436
11.3 Detailed security information 436

12 Core e200z420n3 Description 437
12.1 Overview of the e200z420n3 Core 437
12.2 Register Model 437
12.3 Dual-issue operation 440
12.4 Reservation instructions and cache interactions 441
12.5 Signal Processing Extension/ Embedded Floating-point Status and Control Register (SPEFSCR) . . . 441
12.6 Cache 444
12.6.1 Cache overview 444
12.6.2 L1 Cache Control and Status Register 0 (L1CSR0) 444
12.6.3 L1 Cache Control and Status Register 1 (L1CSR1) 447
12.6.4 L1 Cache Configuration Register 0 (L1CFG0) 449
12.6.5 L1 Cache Configuration Register 1 (L1CFG1) 451
12.6.6 Data Cache Software Coherency 451
12.6.7 Data Cache Hardware Coherency 452
12.6.8 Cache Invalidate by Set and Way 452
12.6.9 Cache EDC/ECC Parity Protection 454
12.6.10 Cache Error Injection 455
12.7 Exceptions 456
12.7.1 Exception Syndrome Register (ESR) 457
12.7.2 Machine State Register (MSR) 458
12.7.3 Machine Check Syndrome Register (MCSR) 459
12.7.4 Interrupt Vector Prefix Registers (IVPR) 463
12.7.5 Interrupt Definitions 463
12.8 Memory Protection Unit (MPU) 473
12.8.1 MPU Overview 473

12.8.2 Access Protection Operation 473
12.8.3 Software Interface and MPU Instructions 474
12.8.4 MPU Read Entry Instruction (mpure) 474
12.8.5 MPU Write Entry Instruction (mpuwe) 474
12.8.6 MPU Synchronize Instruction (mpusync) 474
12.8.7 MMU/MPU Configuration Register (MMUCFG) 475
12.8.8 MPU0 Configuration Register (MPU0CFG) 475
12.8.9 MPU0 Control and Status Register 0 (MPU0CSR0) 476
12.8.10 MPU Assist Registers (MAS) 479
12.8.11 MAS Registers Summary 484
12.9 Local memories 485
12.9.1 Local Instruction and Data Memory Overview 485
12.9.2 Local Memory Control and Configuration 485
12.10 End-to-End ECC Support 500
12.10.1 End-to-End ECC Control and Configuration 500

13 Crossbar switch (XBAR) 505
13.1 Introduction 505
13.1.1 Features 505
13.1.2 Memory map and register definition 505
13.2 Functional description 509
13.2.1 General operation 509
13.2.2 Register coherency 509
13.2.3 Arbitration 510
13.3 Initialization/application information 511

14 System Integration Unit Lite2 (SIUL2) 512
14.1 Introduction 512
14.1.1 Overview 512
14.1.2 Features 514
14.1.3 Register protection 514
14.2 Memory map and register description 514
14.2.1 Memory map 514
14.2.2 Register descriptions 517
14.3 Functional description 532
14.3.1 General 532

14.3.2 Pad control . . . 532
14.3.3 General purpose input or output pads (GPIO) . . . 533
14.3.4 External interrupts/DMA requests (EIRQ pins) . . . 534

15 Crossbar Integrity Checker (XBIC) . . . 537
15.1 Overview . . . 537
15.2 Features . . . 537
15.3 Block diagram . . . 538
15.4 External signal description . . . 539
15.5 Memory map and register definition . . . 539
15.5.1 Memory map . . . 539
15.5.2 Register descriptions . . . 540
15.6 Functional description . . . 544

16 Peripheral bridge . . . 547
16.1 Introduction . . . 547
16.1.1 Features . . . 547
16.1.2 General operation . . . 547
16.2 Memory map/register definition . . . 547
16.2.1 Master Privilege Registers (PBRIDGE_MPRA) . . . 550
16.2.2 Peripheral Access Control Registers (PBRIDGE_PACR*x*) . . . 551
16.2.3 Off-Platform Peripheral Access Control Registers (PBRIDGE_OPACR*x*) . . . 553
16.3 Functional description . . . 556
16.3.1 Access support . . . 556

17 System Memory Protection Unit (SMPU) . . . 557
17.1 Overview . . . 557
17.2 Block diagram . . . 557
17.3 Features . . . 558
17.4 Memory map and register definition . . . 559
17.4.1 Memory map . . . 559
17.4.2 Register descriptions . . . 560
17.5 Functional description . . . 567
17.5.1 Access evaluation macro . . . 567
17.5.2 Putting it all together and error terminations . . . 569

17.6 Initialization information 569
17.7 Application information 569

18 Intelligent AHB Gasket (IAHBG) 572
18.1 Introduction 572
18.2 Timing modes 572
18.2.1 1:1 572
18.2.2 2:1 572
18.2.3 1:2 573

19 Interrupt Controller (INTC) 574
19.1 Introduction 574
19.2 Block diagram 574
19.3 Features 576
19.4 Modes of operation 576
19.4.1 Software vector mode 576
19.4.2 Hardware vector mode 577
19.5 Memory map and register definition 577
19.5.1 Memory map 577
19.5.2 Register descriptions 578
19.6 Functional description 584
19.6.1 Interrupt request sources 584
19.6.2 Priority management 585
19.6.3 Handshaking with processor 586
19.7 Initialization/application information 588
19.7.1 Initialization flow 588
19.7.2 Interrupt exception handler 589
19.7.3 ISR, RTOS, and task hierarchy 591
19.7.4 Order of execution 592
19.7.5 Priority ceiling protocol 593
19.7.6 Selecting priorities according to request rates and deadlines 596
19.7.7 Software-settable interrupt requests 596
19.7.8 Lowering priority within an ISR 597
19.7.9 Negating an interrupt request outside of its ISR 597
19.7.10 Examining LIFO contents 598
19.8 Interrupt sources 598

20 Enhanced Direct Memory Access (eDMA) 599

20.1 Introduction 599

20.1.1 Features 599

20.2 Modes of operation 600

20.2.1 Normal mode 600

20.2.2 Debug mode 601

20.2.3 Wait mode 601

20.3 Memory map/register definition 601

20.3.1 Control Register (DMA_CR) 603

20.3.2 Error Status Register (DMA_ES) 605

20.3.3 Enable Request Register Low (DMA_ERQL) 607

20.3.4 Enable Error Interrupt Register Low (DMA_EEIL) 609

20.3.5 Set Enable Request Register (DMA_SERQ) 613

20.3.6 Clear Enable Request Register (DMA_CERQ) 613

20.3.7 Set Enable Error Interrupt Register (DMA_SEEI) 614

20.3.8 Clear Enable Error Interrupt Register (DMA_CEEI) 615

20.3.9 Clear Interrupt Request Register (DMA_CINT) 615

20.3.10 Clear Error Register (DMA_CERR) 616

20.3.11 Set START Bit Register (DMA_SSRT) 617

20.3.12 Clear DONE Status Bit Register (DMA_CDNE) 617

20.3.13 Interrupt Request Register Low (DMA_INTL) 618

20.3.14 Error Register Low (DMA_ERRL) 621

20.3.15 Hardware Request Status Register Low (DMA_HRSL) 624

20.3.16 Channel *n* Priority Register (DMA_DCHPRI*n*) 626

20.3.17 Channel *n* Master ID Register (DMA_DCHMID*n*) 627

20.3.18 TCD Source Address (DMA_TCD*n*_SADDR) 628

20.3.19 TCD Transfer Attributes (DMA_TCD*n*_ATTR) 629

20.3.20 TCD Signed Source Address Offset (DMA_TCD*n*_SOFF) 629

20.3.21 TCD Minor Byte Count (Minor Loop Disabled) (DMA_TCD*n*_NBYTES_MLNO) 630

20.3.22 TCD Signed Minor Loop Offset (Minor Loop Enabled and Offset Disabled) (DMA_TCD*n*_NBYTES_MLOFFNO) 631

20.3.23 TCD Signed Minor Loop Offset (Minor Loop and Offset Enabled) (DMA_TCD*n*_NBYTES_MLOFFYES) 632

20.3.24 TCD Last Source Address Adjustment (DMA_TCD*n*_SLAST) 633

20.3.25 TCD Destination Address (DMA_TCD*n*_DADDR) 634

20.3.26 TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD*n*_CITER_ELINKYES) 634

20.3.27 TCD Current Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD*n*_CITER_ELINKNO) 635
20.3.28 TCD Signed Destination Address Offset (DMA_TCD*n*_DOFF) 636
20.3.29 TCD Last Destination Address Adjustment/ Scatter Gather Address (DMA_TCD*n*_DLASTSGA) 637
20.3.30 TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD*n*_BITER_ELINKYES) 637
20.3.31 TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD*n*_BITER_ELINKNO) 638
20.3.32 TCD Control and Status (DMA_TCD*n*_CSR) 639
20.4 Functional description 641
20.4.1 eDMA microarchitecture 642
20.4.2 eDMA basic data flow 643
20.4.3 Error reporting and handling 646
20.4.4 Channel preemption 648
20.4.5 eDMA performance 648
20.5 Initialization/application information 651
20.5.1 eDMA initialization 651
20.5.2 DMA programming errors 653
20.5.3 DMA Arbitration mode considerations 654
20.5.4 DMA transfer 655
20.5.5 eDMA TCD*n* status monitoring 658
20.5.6 Channel linking 660
20.5.7 Dynamic programming 661

21 DMA channel multiplexer (DMACHMUX) 664
21.1 Introduction 664
21.1.1 Overview 664
21.1.2 Features 665
21.1.3 Modes of operation 665
21.2 External signal description 665
21.2.1 Overview 665
21.3 Memory map and register definition 665
21.3.1 Register descriptions 666
21.4 Functional description 667
21.4.1 DMA channels with periodic triggering capability 667
21.4.2 DMA channels with no triggering capability 669
21.4.3 “Always Enabled” DMA sources 669

21.5 Initialization/Application information 670
21.5.1 Reset 670
21.5.2 Enabling and configuring sources 670

22 Wakeup unit (WKPU) 675
22.1 Introduction 675
22.1.1 Overview 675
22.1.2 Features 675
22.2 Memory Map and Register Description 675
22.2.1 Memory Map 676
22.2.2 Register Descriptions 676
22.3 Functional Description 680
22.3.1 General 680
22.3.2 Non-Maskable Interrupts 680

23 Temperature sensor (TSENS) 682
23.1 Introduction 682
23.2 Features 682
23.3 Signals 682
23.4 TSENS outputs 682
23.5 Obtaining device temperature using TSENS 683
23.5.1 TSENS calibration constants 683
23.5.2 Equations for converting TSENS voltage to device temperature 683
23.6 Equations for converting CTAT and PTAT voltages into constant reference code 684

24 Clocking 685
24.1 Introduction 685
24.2 Clock generation 685
24.2.1 MC_CGM registers 686
24.3 System clock frequency limitations 687
24.3.1 JTAG frequencies 689
24.4 Default clock configuration 689
24.5 Clock sources 689
24.5.1 PLL 689
24.5.2 External oscillator (XOSC) 691

24.5.3 16 MHz internal RC oscillator (IRCOSC) . . . 693
24.6 Peripheral clocks . . . 693
24.6.1 MCAN clocking . . . 694
24.7 Clock monitoring . . . 695
24.7.1 CMU configuration . . . 695
24.7.2 PLL0 monitor . . . 697
24.7.3 External oscillator (XOSC) monitor . . . 697
24.7.4 Internal RC oscillator (IRCOSC) monitor . . . 698
24.7.5 System clock monitors . . . 698
24.8 Loss of system clock behavior . . . 698
24.8.1 Loss of PLL/XOSC clock . . . 698
24.8.2 Loss of IRCOSC clock . . . 698

25 Dual PLL Digital Interface (PLLDIG) . . . 699
25.1 Introduction . . . 699
25.2 Block Diagram . . . 699
25.3 Features . . . 699
25.4 Modes of operation . . . 699
25.4.1 Normal mode with reference, PLL0 or both PLLs enabled . . . 700
25.5 Memory map and register definition . . . 700
25.5.1 Memory map . . . 700
25.5.2 Register descriptions . . . 701
25.5.3 Register classification for safety requirements . . . 709
25.6 Functional description . . . 710
25.6.1 Input clock frequency . . . 710
25.6.2 Clock configuration . . . 710
25.6.3 Loss of lock . . . 711
25.6.4 Frequency modulation . . . 712
25.7 Initialization information . . . 713

26 Clock Monitor Unit (CMU) . . . 716
26.1 Introduction . . . 716
26.1.1 Main features . . . 716
26.2 Block diagram . . . 716
26.3 Signals . . . 716
26.4 Memory map and register definition . . . 717

26.4.1 Register descriptions . . . 718
26.5 Functional description . . . 723
26.5.1 Frequency meter . . . 723
26.5.2 CLKMN0_RMT supervisor . . . 723
26.5.3 CLKMN1 supervisor . . . 723

27 Clock Generation Module (MC_CGM) . . . 724
27.1 Introduction . . . 724
27.1.1 Overview . . . 724
27.1.2 Features . . . 725
27.2 External signal description . . . 726
27.3 Memory map and register definition . . . 726
27.3.1 Register descriptions . . . 727
27.4 Functional Description . . . 756
27.4.1 System Clock Generation . . . 756
27.4.2 Auxiliary Clock Generation . . . 762
27.4.3 Dividers Functional Description . . . 766

28 OSC Digital Interface (XOSC) . . . 767
28.1 Introduction . . . 767
28.2 Functional description . . . 767
28.2.1 Oscillator power-down control and status . . . 767
28.2.2 Oscillator startup delay . . . 767
28.2.3 Oscillator clock available interrupt . . . 768
28.2.4 Oscillator bypass mode . . . 768
28.3 Memory map and register definition . . . 768
28.3.1 Register descriptions . . . 769

29 IRCOSC digital interface . . . 770
29.1 Introduction . . . 770
29.2 Functional description . . . 770
29.3 Memory map and register definition . . . 770
29.3.1 Register descriptions . . . 771

30 Platform RAM controller (PRAM) . . . 774
30.1 Introduction . . . 774

30.2 SRAM controller memory map and register definitions 775
30.2.1 Platform RAM configuration register 1 (PRCR1) 775
30.3 Functional description 776
30.3.1 Read/Write introduction 776
30.3.2 Writes 777
30.4 Initialization/application information 777
30.5 Safety considerations 778
30.5.1 Hsiao ECC algorithm 778
30.5.2 Transaction monitor 781

31 Flash memory controller (PFLASH Controller) 783
31.1 Introduction 783
31.2 Features 783
31.3 Block diagram 783
31.4 Memory map and register definitions 784
31.4.1 Overview 784
31.5 Functional description 796
31.5.1 Basic interface protocol 796
31.5.2 Access protections 797
31.5.3 Read cycles - buffer miss 797
31.5.4 Read cycles – buffer hit 797
31.5.5 Error termination 797
31.5.6 Flash error response operation 798
31.5.7 Censorship 798
31.5.8 Line read buffers and prefetch operation 798
31.5.9 Instruction/Data prefetch triggering 799
31.5.10 Per-Master prefetch triggering 799
31.5.11 Buffer allocation 799
31.5.12 Safety considerations 800
31.5.13 Array integrity considerations 802

32 Embedded Flash Memory 803
32.1 Introduction 803
32.1.1 Overview 803
32.1.2 Features 805
32.1.3 Modes of operation 805

32.2 Flash memory map and description 806
32.2.1 Flash array memory map 807
32.3 Register memory maps and descriptions 809
32.3.1 Register memory maps 810
32.3.2 User register descriptions 811
32.4 Functional description 843
32.4.1 Reset 843
32.4.2 Power-down mode (Disable mode) 843
32.4.3 Low-power mode (Sleep mode) 844
32.4.4 Read mode 845
32.4.5 Modify mode 847
32.4.6 Protection strategy 858

33 Decorated Storage Memory Controller (DSMC) 860
33.1 Introduction 860
33.2 Decorated Stores: st[b,h,w]d{cb}x rS,rB,rA 862
33.2.1 Bit Field Insert (BFINS) into an 8, 16 or 32-bit Memory Container . . . 862
33.2.2 Compare-and-Store (CAST) 864
33.2.3 Logical AND (AND) 864
33.2.4 Logical OR (OR) 865
33.2.5 Logical Exclusive-OR (XOR) 865
33.3 Decorated Loads: l[b,h,w]d{cb}x rT,rB,rA 865
33.3.1 Simple Load (SLD) 866
33.3.2 Registers-and-Memory Exchange (SWAP) 866
33.3.3 Load-and-Set-1(Bit) (LAS1) 867
33.4 DSMC Instantiations 867

34 Successive Approximation Register Analog-to-Digital Converter (SARADC) 868
34.1 Introduction 868
34.2 Overview 868
34.3 Feature description 870
34.3.1 Main features 870
34.4 Functional description 870
34.4.1 Normal channel conversion 871
34.4.2 Injected channel conversion 872

34.4.3 Abort conversion . . . 873
34.4.4 Analog conversion timings . . . 873
34.4.5 CrossTriggering Unit (CTU) interface . . . 875
34.4.6 Test channel connection with internal analog channel . . . 877
34.4.7 Programmable analog watchdog . . . 877
34.4.8 DMA functionality . . . 879
34.4.9 Interrupts . . . 879
34.4.10 Power down mode . . . 880
34.4.11 Stop Mode . . . 880
34.5 Memory map and registers . . . 880
34.5.1 Register descriptions . . . 883
34.6 Start of conversion pulse delay . . . 913
34.7 Initialization Information . . . 915

35 Enhanced Cross Trigger Unit (ECTU) . . . 917
35.1 Introduction . . . 917
35.2 Main features . . . 917
35.3 Block diagram . . . 917
35.4 Functional description . . . 918
35.4.1 Trigger generator subunit (TGS) . . . 918
35.4.2 Digital filter . . . 920
35.4.3 Scheduler subunit (SU) . . . 921
35.4.4 CTU FIFO . . . 922
35.5 CTU interrupt and DMA requests . . . 923
35.5.1 DMA support . . . 923
35.5.2 CTU faults and errors . . . 924
35.6 ADC commands list . . . 926
35.6.1 ADC commands list format . . . 926
35.6.2 ADC results . . . 930
35.6.3 ADC-CTU interface timing . . . 930
35.7 Low power mode . . . 931
35.8 Safety features . . . 932
35.8.1 Conversion time check . . . 932
35.8.2 CTU reset . . . 932
35.8.3 Motor function iMonitor . . . 933
35.9 CTU error flags examples . . . 935

35.9.1 Invalid command error . . . 935
35.9.2 Maximum concurrent conversion error . . . 935
35.10 Register description . . . 936
35.10.1 Synchronization . . . 936
35.10.2 Double buffered registers . . . 936
35.10.3 TGS counter precision . . . 938
35.10.4 Memory map . . . 940
35.10.5 Trigger generator subunit input selection register (TGSISR) . . . 943
35.10.6 Trigger generator subunit control register (TGSCR) . . . 947
35.10.7 Trigger compare register n (TCRn) . . . 948
35.10.8 TGS counter compare register (TGSCCR) . . . 948
35.10.9 TGS counter reload register (TGSCRR) . . . 949
35.10.10 Commands list control register 1 (CLCR1) . . . 949
35.10.11 Commands list control register 2 (CLCR2) . . . 950
35.10.12 Commands list control register 3 (CLCR3) . . . 950
35.10.13 Commands list control register 4 (CLCR4) . . . 951
35.10.14 Trigger handler control register 1 (THCR1) . . . 952
35.10.15 Trigger handler control register 2 (THCR2) . . . 954
35.10.16 Trigger handler control register 3 (THCR3) . . . 956
35.10.17 Trigger handler control register 4 (THCR4) . . . 958
35.10.18 Commands list register n (CLRn) . . . 960
35.10.19 Control register (CR) . . . 961
35.10.20 Control register 1 FIFO (FCR1) . . . 962
35.10.21 Control register 2 FIFO (FCR2) . . . 964
35.10.22 THRESHOLD register 1 (TH1) . . . 967
35.10.23 THRESHOLD register 2 (TH2) . . . 967
35.10.24 THRESHOLD register 3 (TH3) . . . 968
35.10.25 THRESHOLD register 4 (TH4) . . . 969
35.10.26 Status register 1 (ST1) . . . 970
35.10.27 Status register 2 (ST2) . . . 972
35.10.28 FIFO right aligned register n (FRn) . . . 974
35.10.29 FIFO left aligned register n (FLn) . . . 975
35.10.30 Cross triggering unit error flag register (CTUEFR) . . . 976
35.10.31 Cross triggering unit interrupt flag register (CTUIFR) . . . 978
35.10.32 Cross triggering unit interrupt/DMA register (CTUIR) . . . 979
35.10.33 Control on-time register (COTR) . . . 982
35.10.34 Cross triggering unit control register (CTUCR) . . . 982

35.10.35 Cross triggering unit digital filter (CTU_FILTER) 984
35.10.36 Cross triggering unit configuration status register 1 (CTU_CFGSR1) . 985
35.10.37 Cross triggering unit configuration status register 2 (CTU_CFGSR2) . 986
35.10.38 Cross triggering unit expected value n (CTU_EXPECTEDn) 989
35.10.39 Cross triggering unit counter range (CTU_CNT_RANGEn) 990
35.10.40 CTU expected CRC register (CTU_EXP_CRC_VAL) 990
35.10.41 CTU computed CRC register (CTU_COMP_CRC_VAL) 991
35.10.42 CTU Mask CRC computation register (CTU_MASK_CRC_VAL) 991

36 Motor Control Pulse Width Modulator Module (FlexPWM) 993

36.1 Introduction 993
36.1.1 Overview 993
36.1.2 Features 994
36.1.3 Modes of Operation 994
36.1.4 Block Diagrams 996
36.2 External Signal Descriptions 997
36.2.1 PWMA[n] and PWMB[n] - External PWM Pair 997
36.2.2 PWMX[n] - Auxiliary PWM signal 997
36.2.3 FAULT[n] - Fault inputs 998
36.2.4 EXT_SYNC - External Synchronization Signal 998
36.2.5 EXT_FORCE - External Output Force Signal 998
36.2.6 EXTA[n] and EXTB[n] - Alternate PWM Control Signals 998
36.2.7 OUT_TRIG0[n] and OUT_TRIG1[n] - Output Triggers 998
36.2.8 EXT_CLK - External Clock Signal 998
36.2.9 EXT_SWITCH - External Switch Signal 998
36.3 Memory Map and Registers 998
36.3.1 Module Memory Map 999
36.3.2 Counter register n (CNTn) 1000
36.3.3 Initial Count register n (INITn) 1001
36.3.4 Control 2 register n (CTRL2n) 1002
36.3.5 Control 1 register n (CTRL1n) 1004
36.3.6 Value 0 register n (VAL0n) 1007
36.3.7 Value 1 register n (VAL1n) 1008
36.3.8 Value 2 register n (VAL2n) 1008
36.3.9 Value 3 register n (VAL3n) 1009
36.3.10 Value 4 register n (VAL4n) 1009
36.3.11 Value 5 register n (VAL5n) 1010

36.3.12 Output Control register n (OCTRLn) 1010
36.3.13 Status register n (STSn) 1012
36.3.14 Interrupt Enable register n (INTENn) 1013
36.3.15 DMA Enable register n (DMAENn) 1014
36.3.16 Output Trigger Control register n (TCTRLn) 1016
36.3.17 Fault Disable Mapping register n (DISMAPn) 1016
36.3.18 Deadtime Count 0 register n (DTCNT0n) 1017
36.3.19 Deadtime Count 1 register n (DTCNT1n) 1017
36.3.20 Capture Control A register n (CAPTCTRLAn) 1018
36.3.21 Capture Compare A register n (CAPTCMPAn) 1020
36.3.22 Capture Control B register n (CAPTCTRLBn) 1020
36.3.23 Capture Compare B register n (CAPTCMPBn) 1022
36.3.24 Capture Control X register n (CAPTCTRLXn) 1022
36.3.25 Capture Compare X register n (CAPTCMPXn) 1024
36.3.26 Capture Value 0 register n (CVAL0n) 1025
36.3.27 Capture Value 0 Cycle register n (CVAL0CYCn) 1025
36.3.28 Capture Value 1 register n (CVAL1n) 1026
36.3.29 Capture Value 1 Cycle register n (CVAL1CYCn) 1026
36.3.30 Capture Value 2 register n (CVAL2n) 1027
36.3.31 Capture Value 2 Cycle register n (CVAL2CYCn) 1027
36.3.32 Capture Value 3 register n (CVAL3n) 1028
36.3.33 Capture Value 3 Cycle register n (CVAL3CYCn) 1028
36.3.34 Capture Value 4 register n (CVAL4n) 1029
36.3.35 Capture Value 4 Cycle register n (CVAL4CYCn) 1029
36.3.36 Capture Value 5 register n (CVAL5n) 1030
36.3.37 Capture Value 5 Cycle register n (CVAL5CYCn) 1030
36.3.38 Configuration Registers 1030
36.3.39 Fault Channel Registers 1036
36.4 Functional Description 1039
36.4.1 Block Diagram 1039
36.4.2 PWM Capabilities 1039
36.4.3 Functional Details 1049
36.4.4 PWM Generator Loading 1066
36.5 Resets 1069
36.6 Clocks 1069
36.7 Interrupts 1069

36.8 DMA 1071

37 PWM Synchronizer 1073

37.1 Introduction 1073
37.2 Main Features 1074
37.3 Memory map and register description 1074
37.3.1 Memory map 1074
37.3.2 Register descriptions 1075
37.4 Programming Sequence 1084
37.5 Functional description 1084

38 Enhanced Motor Control Timer (eTimer) 1088

38.1 Introduction 1088
38.1.1 Overview 1088
38.1.2 Features 1089
38.1.3 Module block diagram 1090
38.1.4 Channel block diagram 1091
38.2 External signal descriptions 1091
38.2.1 TIO[*n*:0] - Timer Input/Outputs 1091
38.2.2 TAI[*n*:0] - Timer Auxiliary Inputs 1091
38.3 Functional Description 1092
38.3.1 General 1092
38.3.2 Counting Modes 1092
38.3.3 Other Features 1100
38.4 Memory Map and Registers 1101
38.4.1 Overview 1101
38.4.2 Module Memory Map 1101
38.4.3 Register Descriptions 1102
38.4.4 Timer Channel Registers 1102
38.4.5 Watchdog Timer Registers 1117
38.4.6 Configuration Registers 1118
38.5 Resets 1122
38.6 Clocks 1123
38.7 Interrupts 1123
38.8 DMA 1123

39 System Timer Module (STM) 1125
39.1 Introduction 1125
39.1.1 Overview 1125
39.1.2 Features 1125
39.1.3 Modes of operation 1125
39.2 External signal description 1125
39.3 Memory map and register definition 1125
39.3.1 Memory map 1125
39.3.2 Register descriptions 1126
39.4 Functional description 1129

40 Software Watchdog Timer (SWT) 1130
40.1 Introduction 1130
40.1.1 Overview 1130
40.1.2 Features 1130
40.1.3 Modes of operation 1130
40.2 External signal description 1130
40.3 Memory map and register definition 1130
40.3.1 Register descriptions 1131
40.4 Functional description 1136
40.4.1 Introduction 1136
40.4.2 Configuration locking 1137
40.4.3 Unlock sequence 1137
40.4.4 Servicing operations 1137
40.4.5 Time-out 1138
40.4.6 Initialization 1138

41 Periodic Interrupt Timer (PIT) 1139
41.1 Introduction 1139
41.1.1 Block diagram 1139
41.1.2 Features 1140
41.2 Signal description 1140
41.2.1 Memory map/register description 1140
41.3 Functional description 1143
41.3.1 General 1144
41.3.2 Interrupts 1145

41.3.3 Chained timers 1145
41.4 Initialization and application information 1145

42 CAN Subsystem 1147
42.1 Introduction 1147
42.2 Features 1148
42.3 Modular CAN (M_CAN) cores 1149
42.3.1 Features 1150
42.3.2 Block diagram 1151
42.3.3 Dual clock sources 1152
42.3.4 Dual interrupt lines 1152
42.3.5 Memory map and register descriptions 1152
42.3.6 Message RAM 1200
42.3.7 M_CAN functional description 1210
42.3.8 Timestamp generation 1220
42.3.9 Timeout counter 1220
42.3.10 Rx handling 1221
42.3.11 Dedicated Rx Buffers 1227
42.3.12 Debug on CAN Support 1228
42.3.13 Tx handling 1230
42.3.14 FIFO acknowledge handling 1234
42.3.15 Clock Calibration on CAN Unit (CCCU) 1234
42.4 CAN RAM arbiter 1246
42.4.1 Features 1247
42.4.2 Functional overview using examples 1247
42.5 SRAM interface and memory organization 1247
42.5.1 ECC controller 1247
42.6 CAN nodes 0 and 2 I/Os sharing 1248
42.7 External signal description 1249
42.8 Shared memory map 1249

43 Deserial Serial Peripheral Interface (DSPI) 1252
43.1 Introduction 1252
43.1.1 Block diagram 1252
43.1.2 Features 1253
43.1.3 SPI configuration 1254

43.1.4 Modes of operation 1255
43.2 DSPI signal description 1256
43.2.1 PCS0/SS — Peripheral Chip Select 0 / Slave Select 1256
43.2.2 PCS[1:3] — Peripheral Chip Selects 1 – 3 1257
43.2.3 PCS4/MTRIG — Peripheral Chip Select 4 / Master Trigger 1257
43.2.4 PCS5/PCSS — Peripheral Chip Select 5 / Peripheral Chip Select Strobe 1257
43.2.5 PCS[6:7] — Peripheral Chip Selects 6 – 7 1257
43.2.6 SIN — Serial Input 1257
43.2.7 SOUT — Serial Output 1257
43.2.8 SCK — Serial Clock 1257
43.2.9 Memory map/register definition 1257
43.3 Register classification for safety 1283
43.4 Functional description 1283
43.4.1 Start and Stop of DSPI transfers 1284
43.4.2 Serial Peripheral Interface (SPI) configuration 1285
43.4.3 DSPI baud rate and clock delay generation 1288
43.4.4 Transfer formats 1291
43.4.5 Continuous Serial Communications Clock 1300
43.4.6 Slave mode operation constraints 1301
43.4.7 Parity generation and check 1302
43.4.8 Interrupts/DMA requests 1302
43.4.9 Power saving features 1305
43.5 Initialization/application information 1306
43.5.1 Managing DSPI queues 1306
43.5.2 Switching master and slave mode 1306
43.5.3 Initializing DSPI in Master/Slave Modes 1307
43.5.4 Baud rate settings 1307
43.5.5 Delay settings 1308
43.5.6 Calculation of FIFO pointer addresses 1309

44 LINFlexD 1312
44.1 Introduction 1312
44.2 Main features 1314
44.2.1 LIN mode features 1314
44.2.2 UART mode features 1315
44.2.3 Standard features 1315

44.3 Functional description 1315
44.3.1 LIN protocol 1315
44.3.2 LINFlexD features 1317
44.3.3 Timer 1333
44.3.4 UART mode 1334
44.3.5 DMA interface 1339
44.4 Memory map and register description 1357
44.4.1 Memory map 1357
44.4.2 Register description 1359
44.5 Programming considerations 1392
44.5.1 Master node 1392
44.5.2 Slave node 1394
44.5.3 Extended frames 1398
44.5.4 Timeout 1398
44.5.5 UART mode 1399
44.5.6 Interrupts 1399

45 SENT Receiver (SRX) 1401
45.1 Introduction 1401
45.1.1 Features 1402
45.1.2 Modes of operation 1403
45.1.3 Block diagram 1405
45.1.4 Design overview 1405
45.2 External signal description 1407
45.2.1 SENT_RX [(CH-1):0] 1407
45.3 Memory map and register definition 1407
45.3.1 Memory map 1407
45.3.2 Register descriptions 1409
45.3.3 Register Classification for Safety 1440
45.4 Functional description 1440
45.4.1 Initialization sequence 1440
45.4.2 DMA read logic 1441
45.4.3 Message reading via interrupts 1447
45.4.4 Overflow behavior 1448
45.4.5 Adjustment for variation in sensor (Tx) clock 1448
45.4.6 Input programmable filter 1449

45.4.7 Receiver diagnostics 1450
45.4.8 Time stamp logic 1454
45.4.9 Bus Idle Diagnostic 1455
45.5 Clocks and resets 1455
45.5.1 Clocking strategy 1455
45.5.2 System bus clock requirements 1456
45.5.3 High frequency receiver clock (protocol clock) requirements 1456

46 FlexRay Communication Controller (FlexRay) 1458
46.1 Introduction 1458
46.1.1 Reference 1458
46.1.2 Color coding 1458
46.1.3 Overview 1458
46.1.4 Features 1460
46.1.5 Modes of operation 1461
46.2 External Signal Description 1463
46.2.1 Detailed signal description 1464
46.3 Controller Host Interface clocking 1465
46.4 Protocol engine clocking 1465
46.4.1 Oscillator clocking 1465
46.4.2 PLL clocking 1465
46.5 Memory map and register description 1465
46.5.1 Memory map 1466
46.5.2 Register descriptions 1471
46.6 Functional description 1562
46.6.1 Message buffer concept 1562
46.6.2 Physical message buffer 1562
46.6.3 Message Buffer Types 1563
46.6.4 FlexRay memory area layout 1570
46.6.5 Physical message buffer description 1573
46.6.6 Individual message buffer functional description 1583
46.6.7 Individual message buffer search 1601
46.6.8 Individual message buffer reconfiguration 1604
46.6.9 Receive FIFOs 1604
46.6.10 Channel device modes 1610
46.6.11 External clock synchronization 1612

46.6.12 Sync frame ID and sync frame deviation tables 1613
46.6.13 MTS generation 1616
46.6.14 Key slot transmission 1617
46.6.15 Sync frame filtering 1617
46.6.16 Strobe signal support 1618
46.6.17 Timer support 1619
46.6.18 Slot status monitoring 1620
46.6.19 System bus access 1624
46.6.20 Interrupt support 1625
46.6.21 Lower bit rate support 1629
46.6.22 PE data memory (PE DRAM) 1630
46.6.23 CHI lookup-table memory (CHI LRAM) 1631
46.6.24 Memory content error detection 1632
46.6.25 Memory error injection 1636
46.6.26 StopWatch function 1638
46.7 Application information 1639
46.7.1 Module configuration 1639
46.7.2 Initialization sequence 1640
46.7.3 Memory error injection out of POC:default config 1642
46.7.4 Shutdown sequence 1642
46.7.5 Number of usable message buffers 1642
46.7.6 Protocol Control Command Execution 1644
46.7.7 Message buffer search on simple message buffer configuration 1645

47 Reset Generation Module (MC_RGM) 1648
47.1 Introduction 1648
47.1.1 Overview 1648
47.1.2 Features 1649
47.1.3 Reset sources 1650
47.2 External signal description 1651
47.3 Memory map and register definition 1651
47.3.1 Memory map 1651
47.3.2 Register descriptions 1653
47.4 Functional description 1691
47.4.1 Reset state machine 1691
47.4.2 ‘Destructive’ resets 1694
47.4.3 ‘Functional’ resets 1694

47.4.4 Alternate event generation 1695
47.4.5 ‘Functional’ reset escalation 1695
47.4.6 ‘Destructive’ reset escalation 1696
47.4.7 Individual peripheral resets 1696

48 Boot Assist Flash (BAF) 1697
48.1 Introduction 1697
48.2 Memory map 1697
48.2.1 BAF image header 1697
48.3 Functional description 1698
48.3.1 Boot modes 1698
48.3.2 Device initialization 1698
48.3.3 Flow of control 1699
48.3.4 Initialization of BAF DCF clients 1700
48.3.5 Optionally perform a serial boot 1705
48.3.6 Serial boot configuration 1710
48.4 Resources accessed by BAF code execution 1713

49 System Status and Configuration Module (SSCM) 1714
49.1 Introduction 1714
49.1.1 Overview 1714
49.1.2 Features 1714
49.1.3 Modes of operation 1714
49.2 Memory map and register definition 1714
49.2.1 Register descriptions 1715
49.3 Functional description 1719
49.3.1 DCF mechanism 1719
49.3.2 Boot mode functionality 1721
49.3.3 BAF configuration 1721
49.3.4 User code boot sector search 1722
49.3.5 Life Cycle 1723
49.4 Initialization and application information 1724
49.4.1 Reset 1724
49.5 Additional safety measures 1724
49.5.1 Spurious reset protection 1724

50 Power management controller digital interface (PMC_Dig) 1725

50.1 Introduction 1725

50.2 Main features 1727

50.2.1 Standard features 1727

50.3 IPS bus interface 1727

50.4 Memory Map and Registers 1727

50.5 Register description 1730

50.5.1 Event Pending Register (EPR_LV0) 1730

50.5.2 Reset Event Enable Register (REE_LV0) 1731

50.5.3 Reset Event Select Register (RES_LV0) 1732

50.5.4 Interrupt Enable (IE_LV0) 1732

50.5.5 FCCU Event Enable Register (FEE_LV0) 1733

50.5.6 Event Pending Register (EPR_LV1) 1734

50.5.7 Reset Event Enable Register (REE_LV1) 1734

50.5.8 Reset Event Select Register (RES_LV1) 1735

50.5.9 Interrupt Enable (IE_LV1) 1736

50.5.10 FCCU Event Enable Register (FEE_LV1) 1737

50.5.11 Event Pending Register (EPR_HV0) 1738

50.5.12 Reset Event Enable Register (REE_HV0) 1739

50.5.13 Reset Event Select Register (RES_HV0) 1741

50.5.14 Interrupt Enable (IE_HV0) 1742

50.5.15 FCCU Event Enable Register (FEE_HV0) 1743

50.5.16 Event Pending Register (EPR_HV1) 1745

50.5.17 Reset Event Enable Register (REE_HV1) 1746

50.5.18 Reset Event Select Register (RES_HV1) 1748

50.5.19 Interrupt Enable Register (IE_HV1) 1749

50.5.20 FCCU Event Enable Register (FEE_HV1) 1751

50.5.21 Supply Gauge Status Register (*GR_S*) 1752

50.5.22 Pending Gauge Status Register (GR_P) 1755

50.5.23 Interrupt Enable Pending Register (IE_G) 1756

50.5.24 Voltage Selection of IO (VSIO) 1758

50.5.25 IPS Regulator Control PMOS NMOS register (IREGU_CTL_PN) . . . 1759

50.5.26 Event Pending Register (EPR_TD) 1760

50.5.27 Reset Event Enable Register (REE_TD) 1762

50.5.28 Reset Event Select Register (RES_TD) 1763

50.5.29 Temperature Sensor Configuration register (CTL_TD) 1765

50.5.30 Temperature Sensor FCCU Event Enable Register (FEE_TD) 1766
50.5.31 IPS Regulator Monitor Register (IREGU_MON) 1767
50.5.32 ESR0 ConFiGuration Register (ESR0_CFG) 1768
50.5.33 User BIST Flags_IO10 Register (BIST_FLAGS_IO10) 1768
50.5.34 User BIST Flags Register (BIST_FLAGS) 1769
50.5.35 User BIST Control Register (BIST_CTRL) 1770
50.5.36 User BIST Time1 and Time0 Register (BIST_TIME10) 1771
50.5.37 User BIST Time3 and Time2 Register (BIST_TIME32) 1772
50.5.38 User BIST Time6 and Time5 Register (BIST_TIME65) 1772
50.5.39 User BIST VD Under Test Monitor Register (BIST_DEBUG) 1773
50.6 Temperature Sensor Interface Logic 1774
50.7 FCCU Interface 1774
50.8 Power On Reset (MC_RGM Phase Gates) 1775
50.9 Flash interface (DCF client usage) 1775

51 Power Control Unit (MC_PCU) 1777
51.1 Introduction 1777
51.1.1 Overview 1777
51.1.2 Features 1778
51.2 External signal description 1778
51.3 Memory map and register definition 1778
51.3.1 Memory map 1778
51.3.2 Register descriptions 1778

52 Mode Entry Module (MC_ME) 1780
52.1 Introduction 1780
52.1.1 Overview 1780
52.1.2 Features 1782
52.1.3 Modes of operation 1782
52.2 External signal description 1783
52.3 Memory map and register definition 1783
52.3.1 Memory map 1784
52.3.2 Register description 1787
52.4 Functional description 1825
52.4.1 Mode Transition Request 1825
52.4.2 Modes details 1826

52.4.3 Mode Transition Process 1830
52.4.4 Protection of mode configuration registers 1838
52.4.5 Mode Transition Interrupts 1838
52.4.6 Peripheral Clock Gating 1840
52.4.7 Application example 1840

53 e200z420n3 Core Debug Support 1842

53.1 Overview 1842
53.1.1 Software Debug Facilities 1842
53.1.2 Additional Debug Facilities 1843
53.1.3 Hardware Debug Facilities 1843
53.1.4 Sharing Debug Resources by Software/Hardware 1844
53.2 Software Debug Events and Exceptions 1845
53.2.1 Instruction Address Compare Event 1846
53.2.2 Data Address Compare Event 1847
53.2.3 Linked Instruction Address Compare and Data Address Compare Events 1854
53.2.4 Trap Debug Event 1855
53.2.5 Branch Taken Debug Event 1855
53.2.6 Instruction Complete Debug Event 1855
53.2.7 Interrupt Taken Debug Event 1856
53.2.8 Critical Interrupt Taken Debug Event 1856
53.2.9 Return Debug Event 1856
53.2.10 Critical Return Debug Event 1856
53.2.11 External Debug Event 1857
53.2.12 Unconditional Debug Event 1857
53.2.13 Performance Monitor Interrupt Debug Event 1857
53.3 Debug Registers 1857
53.3.1 Debug Address and Value Registers 1858
53.3.2 Debug Control and Status Registers 1859
53.3.3 External Debug Resource Allocation Control Register (EDBRAC0) . 1885
53.3.4 Debug Event Select Register (DEVENT) 1892
53.3.5 Debug Data Acquisition Message Register (DDAM) 1893
53.4 Using Debug Resources for Stack Limit Checking 1894
53.5 External Debug Support 1896
53.5.1 External Debug Registers 1897
53.5.2 OnCE Introduction 1903

53.5.3 JTAG/OnCE Pins . . . 1906
53.5.4 OnCE Internal Interface Signals . . . 1906
53.5.5 OnCE Interface Signals . . . 1907
53.5.6 OnCE Controller and Serial Interface . . . 1909
53.5.7 Access to Debug Resources . . . 1916
53.5.8 Methods of Entering Debug Mode . . . 1918
53.5.9 CPU Status and Control Scan Chain Register (CPUSCR) . . . 1919
53.5.10 Reserved Registers (Reserved) . . . 1926
53.6 Watchpoint Support . . . 1926
53.7 MPU Operation During Debug . . . 1929
53.8 Cache Array Access During Debug . . . 1929
53.9 Basic Steps for Enabling, Using, and Exiting External Debug Mode . . . 1929

54 Debug and Calibration Interface (DCI) . . . 1932
54.1 Introduction . . . 1932
54.1.1 Features . . . 1932
54.1.2 Overview . . . 1932
54.1.3 Operating modes . . . 1934
54.2 External signal description . . . 1941
54.3 Register description . . . 1941
54.3.1 DCI control register (DCI_CR) . . . 1941
54.3.2 DCI EVTx pin multiplexing control register (DCI_PINCR) . . . 1942

55 Debug and Trace . . . 1943
55.1 Introduction . . . 1943
55.1.1 Features . . . 1943
55.1.2 Architecture . . . 1944
55.2 Register description . . . 1946
55.2.1 Nexus Crossbar Multimaster Trace Clients (NXMC) . . . 1946
55.2.2 Sequence Processing Unit (SPU) . . . 1946
55.3 Lock step behavior . . . 1962
55.3.1 Debugging phases . . . 1965
55.3.2 Internal register reading . . . 1966
55.3.3 Step-by-step execution . . . 1966
55.3.4 Exiting a debug session . . . 1967
55.3.5 External tool operation . . . 1967

55.4 Lock step tracing 1967
55.5 External signal description 1967
55.5.1 Interfaces 1967
55.5.2 Nexus trace enabling during reset 1968
55.6 Functionality 1976
55.6.1 JTAG Master (JTAGM) 1976
55.6.2 Nexus Crossbar Multimaster Trace Client (NXMC) 1976
55.6.3 JTAG Data Communication module (JDC) 1976
55.6.4 Nexus Port Controller (NPC) 1976
55.6.5 Sequence Processing Unit (SPU) 1977

56 JTAG Controller (JTAGC) 1978
56.1 Introduction 1978
56.1.1 Block Diagram 1978
56.1.2 Features 1979
56.1.3 Modes of operation 1979
56.2 External signal description 1980
56.2.1 TCK—Test clock input 1980
56.2.2 TDI—Test data input 1980
56.2.3 TDO—Test data output 1980
56.2.4 TMS—Test mode select 1980
56.2.5 JCOMP—JTAG compliancy 1981
56.3 Register description 1981
56.3.1 Instruction register 1981
56.3.2 Bypass register 1981
56.3.3 Device identification register 1981
56.3.4 TEST_CTRL Register 1982
56.3.5 JTAG_PASSWORD register 1983
56.3.6 DATA1 register 1983
56.3.7 DATA2 register 1984
56.3.8 Boundary scan register 1984
56.4 Functional description 1985
56.4.1 JTAGC reset configuration 1985
56.4.2 IEEE 1149.1-2001 (JTAG) Test Access Port 1985
56.4.3 TAP controller state machine 1985
56.4.4 JTAGC block instructions 1987

56.4.5 Boundary scan 1991
56.5 Initialization/Application information 1991

57 JTAG Data Communication (JDC) 1992
57.1 Introduction 1992
57.2 Overview 1992
57.3 Register definition 1993
57.3.1 Register descriptions 1993
57.3.2 Non-Memory mapped register definition 1995
57.4 Functional description 1996
57.4.1 IPS register access 1997
57.4.2 JTAG register access 1997

58 JTAG Master (JTAGM) 1998
58.1 Introduction 1998
58.1.1 Overview 1998
58.1.2 Feature description 1999
58.2 Functional description 1999
58.2.1 Module architecture 1999
58.2.2 JTAGM Ready signal 2000
58.2.3 EVTO and EVTI signals 2000
58.2.4 JTAGM configuration and status monitoring 2001
58.2.5 JTAGM to DCI serial interface 2001
58.2.6 Software interface 2001
58.3 Modes of operation 2001
58.4 Memory map and register definition 2001
58.4.1 Register descriptions 2002

59 Nexus Port Controller (NPC) 2008
59.1 Introduction 2008
59.1.1 Overview 2008
59.1.2 Features 2009
59.1.3 Modes of Operation 2009
59.2 External signal description 2010
59.2.1 Overview 2010
59.2.2 Detailed Signal Descriptions 2010

59.3 Register definition 2011
59.3.1 Register descriptions 2012
59.4 Functional description 2015
59.4.1 NPC reset configuration 2015
59.4.2 Auxiliary Output Port 2016
59.4.3 IEEE 1149.1-2001 (JTAG) TAP 2019
59.4.4 Nexus JTAG Port Sharing 2023
59.4.5 MCKO Control 2023
59.4.6 EVTO Sharing 2023
59.4.7 Nexus Reset Control 2023
59.5 Initialization/Application Information 2024
59.5.1 Accessing NPC tool-mapped registers 2024

60 Sequence Processing Unit (SPU) 2025
60.1 Introduction 2025
60.1.1 SPU block diagram 2025
60.1.2 Overview 2026
60.1.3 Features 2028
60.2 SPU actions 2029
60.3 Operation mode 2030
60.3.1 SPU enable/disable 2030
60.4 SPU module interface 2030
60.5 Register definition 2030
60.5.1 Registers description 2035
60.6 Functional description 2072
60.6.1 Input Mux unit 2072
60.6.2 Processor exception vector encoding 2074
60.6.3 State logic unit (SLU) 2075
60.6.4 Sequence formation 2076
60.6.5 Performance counters/timers unit 2079
60.6.6 Action unit 2081
60.6.7 Optional trace group 2081

61 e200z420n3 Nexus 3 Module 2082
61.1 Introduction 2082
61.1.1 General Description 2082

61.1.2 Terms and Definitions 2082
61.1.3 Feature List 2083
61.1.4 Functional Block Diagram 2085
61.2 Enabling Nexus 3 Operation 2085
61.2.1 Interaction with Low Power Modes 2086
61.3 TCODEs supported 2086
61.4 Nexus 3 Programmer's Model 2092
61.4.1 Client Select Control Register (CSC) 2094
61.4.2 Port Configuration Register - reference only (PCR) 2094
61.4.3 Nexus Development Control Register 1 (DC1) 2095
61.4.4 Nexus Development Control Registers 2-3 (DC2-3) 2097
61.4.5 Nexus Development Control Register 4 (DC4) 2098
61.4.6 Development Status Register (DS) 2100
61.4.7 Watchpoint Trigger Registers (WT, PTSTC, PTETC, DTSTC, DTETC) 2100
61.4.8 Nexus Watchpoint Mask Register (WMSK) 2110
61.4.9 Nexus Overrun Control Register (OVCR) 2112
61.4.10 Data Trace Control Register (DTC) 2113
61.4.11 Data Trace Start Address Register n (DTSAn) 2115
61.4.12 Data Trace End Address Register n (DTEAn) 2116
61.4.13 Read/Write Access Control/Status (RWCS) 2117
61.4.14 Read/Write Access Data (RWD) 2118
61.4.15 Read/Write Access Address (RWA) 2120
61.5 Nexus 3 Register Access via JTAG/OnCE 2120
61.6 Nexus 3 Register Access via Software 2121
61.7 Nexus Message Fields 2121
61.7.1 TCODE Field 2122
61.7.2 Source ID Field (SRC) 2122
61.7.3 Relative Address Field (U-ADDR) 2122
61.7.4 Full Address Field (F-ADDR) 2123
61.8 Nexus Message Queues 2123
61.8.1 Message Queue Overrun 2124
61.8.2 CPU Stall 2124
61.8.3 Message Suppression 2124
61.8.4 Nexus Message Priority 2124
61.8.5 Data Acquisition Message Priority Loss Response 2125

61.8.6 Ownership Trace Message Priority Loss Response 2126
61.8.7 Program Trace Message Priority Loss Response 2126
61.8.8 Program Trace Sync Message Priority Loss Response 2126
61.8.9 Data Trace Message Priority Loss Response 2126
61.9 Debug Status Messages 2126
61.10 Error Messages 2127
61.11 Ownership Trace 2127
61.11.1 Overview 2127
61.11.2 Ownership Trace Messaging (OTM) 2127
61.12 Program Trace 2128
61.12.1 Branch Trace Messaging Types 2129
61.12.2 BTM Message Formats 2130
61.12.3 Program Trace Message Fields 2131
61.12.4 Resource Full Messages 2132
61.12.5 Program Correlation Messages 2133
61.12.6 Program Trace Overflow Error Messages 2134
61.12.7 Program Trace Synchronization Messages 2134
61.12.8 Program Trace Direct/Indirect Branch with Sync Messages 2136
61.12.9 Enabling Program Trace 2137
61.12.10 Program Trace Timing Diagrams (2 MDO / 1 MSEO configuration) . 2138
61.13 Data Trace 2139
61.13.1 Data Trace Messaging (DTM) 2139
61.13.2 DTM Message Formats 2139
61.13.3 DTM Operation 2142
61.13.4 Data Trace Timing Diagrams(8 MDO / 2 MSEO configuration) 2144
61.14 Data Acquisition Messaging 2144
61.14.1 Data Acquisition ID Tag Field 2144
61.14.2 Data Acquisition Data Field 2144
61.14.3 Data Acquisition Trace event 2145
61.15 Watchpoint Trace Messaging 2145
61.15.1 Watchpoint Timing Diagram (2 MDO / 1 MSEO configuration) 2147
61.16 External Hardware Trigger Controls 2147
61.17 Nexus 3 Read/Write Access to Memory-Mapped Resources 2148
61.17.1 Single Write Access 2149
61.17.2 Block Write Access 2149
61.17.3 Single Read Access 2150

61.17.4 Block Read Access 2150
61.17.5 Error Handling 2151
61.17.6 Read/Write Access Error Message 2152
61.18 Nexus 3 Pin Interface 2152
61.18.1 Pins Implemented 2152
61.18.2 Pin Protocol 2154
61.19 Rules for Output Messages 2156
61.20 Auxiliary Port Arbitration 2157
61.21 Examples 2157
61.22 Electrical Characteristics 2160
61.23 IEEE 1149.1 (JTAG) RD/WR Sequences 2160
61.23.1 JTAG Sequence for Accessing Internal Nexus Registers 2160
61.23.2 JTAG Sequence for Read Access of Memory-Mapped Resources . . 2161
61.23.3 JTAG Sequence for Write Access of Memory-Mapped Resources . . 2161

62 Nexus Crossbar Multi-Master Client (NXMC) 2162
62.1 Introduction 2162
62.1.1 Features 2162
62.2 External signal description 2162
62.3 Supported messages and TCODEs 2163
62.4 Register description 2166
62.4.1 Register definitions 2167
62.5 Programming considerations (RESET) 2174
62.5.1 IEEE 1149.1 (JTAG) Test Access Port 2174
62.5.2 Enabling NXMC operation 2174
62.5.3 Enabling NXMC TAP controller 2174
62.5.4 NXMC register access via JTAG 2175
62.6 Functional description 2175
62.6.1 Data trace 2175
62.6.2 Watchpoint support 2180

63 Cyclic Redundancy Check (CRC) Unit 2182
63.1 Introduction 2182
63.2 Main features 2182
63.2.1 Standard features 2182

63.3 Block diagram . . . 2182
63.4 Signal description . . . 2183
63.4.1 Peripheral bus interface . . . 2183
63.5 Memory map and registers . . . 2183
63.5.1 Register description . . . 2183
63.6 Functional description . . . 2187
63.7 Use cases . . . 2189
63.7.1 Programming example . . . 2189
63.7.2 Register programming . . . 2191

64 Memory Error Management Unit (MEMU) . . . 2192
64.1 Introduction . . . 2192
64.2 Features . . . 2193
64.3 Block diagram . . . 2193
64.4 Design overview . . . 2194
64.5 External signal description . . . 2197
64.6 Memory map and register definition . . . 2197
64.6.1 Memory map . . . 2197
64.6.2 Register descriptions . . . 2199
64.7 Functional description . . . 2213
64.7.1 Initializing MEMU . . . 2213
64.7.2 Reading the reporting table . . . 2214
64.7.3 Handling overflows (Multiple error reporting) . . . 2215

65 Fault Collection and Control Unit (FCCU) . . . 2216
65.1 Introduction . . . 2216
65.2 Features . . . 2216
65.2.1 Standard features . . . 2217
65.3 Block diagram . . . 2217
65.4 Signal description . . . 2218
65.4.1 Pinout . . . 2218
65.4.2 IPS bus interface . . . 2219
65.5 Functional description . . . 2219
65.5.1 Definitions . . . 2219
65.5.2 FSM description . . . 2220

65.5.3 Reset interface . . . 2221
65.5.4 Fault priority scheme and nesting . . . 2222
65.5.5 Fault recovery . . . 2223
65.5.6 NMI/WKPU interface . . . 2225
65.5.7 FAULT interface . . . 2226
65.5.8 STCU interface . . . 2227
65.5.9 EOUT interface . . . 2228
65.5.10 Error signal flow for RF case . . . 2234
65.6 Register description . . . 2236
65.6.1 FCCU Control Register (FCCU_CTRL) . . . 2239
65.6.2 FCCU CTRL Key Register (FCCU_CTRLK) . . . 2241
65.6.3 FCCU Configuration Register (FCCU_CFG) . . . 2242
65.6.4 FCCU RF Configuration Register n (FCCU_RF_CFGn) . . . 2245
65.6.5 FCCU RFS Configuration Register n (FCCU_RFS_CFGn) . . . 2246
65.6.6 FCCU RF Status register n (FCCU_RF_Sn) . . . 2247
65.6.7 FCCU RF Key register (FCCU_RFK) . . . 2249
65.6.8 FCCU RF Enable Register n (FCCU_RF_En) . . . 2249
65.6.9 FCCU RF Timeout Enable Register n (FCCU_RF_TOEn) . . . 2250
65.6.10 FCCU RF Timeout Register (FCCU_RF_TO) . . . 2251
65.6.11 FCCU CFG Timeout Register (FCCU_CFG_TO) . . . 2252
65.6.12 FCCU IO Control Register (FCCU_EINOUT) . . . 2253
65.6.13 FCCU Status Register (FCCU_STAT) . . . 2255
65.6.14 FCCU NA Freeze Status Register (N2AF_STATUS) . . . 2256
65.6.15 FCCU AF Freeze Status Register (A2FF_STATUS) . . . 2257
65.6.16 FCCU NF Freeze Status Register (N2FF_STATUS) . . . 2258
65.6.17 FCCU FA Freeze Status Register (F2A_STATUS) . . . 2259
65.6.18 FCCU RF Fake Register (FCCU_RFF) . . . 2260
65.6.19 FCCU IRQ Status Register (FCCU_IRQ_STAT) . . . 2261
65.6.20 FCCU IRQ Enable Register (FCCU_IRQ_EN) . . . 2262
65.6.21 FCCU XTMR Register (FCCU_XTMR) . . . 2263
65.6.22 FCCU MCS Register (FCCU_MCS) . . . 2264
65.6.23 FCCU Transient Register (FCCU_TRANS_LOCK) . . . 2266
65.6.24 FCCU Permanent Lock Register (FCCU_PERMNT_LOCK) . . . 2267
65.6.25 FCCU Delta T Register (FCCU_DELTA_T) . . . 2267
65.6.26 FCCU IRQ Alarm Enable register n (FCCU_IRQ_ALARM_ENn) . . . 2268
65.6.27 FCCU NMI Enable registers n (FCCU_NMI_ENn) . . . 2269

65.6.28 FCCU EOUT Signaling Enable registers n (FCCU_EOUT_SIG_ENn) 2270
65.7 FCCU Output Supervision Unit 2271
65.8 Use cases and limitations 2273
65.8.1 Misconfigurations 2273
65.8.2 Recommendations to configure FCCU 2274

66 Self-Test Control Unit (STCU2) 2275
66.1 Introduction 2275
66.2 Main features 2276
66.3 Block diagram 2277
66.3.1 IPS bus interface 2278
66.4 Functional description 2279
66.4.1 FSM description 2279
66.4.2 Reset management 2279
66.4.3 LBIST and MBIST scheduling 2280
66.4.4 ABORT management 2280
66.4.5 PLL interface 2280
66.4.6 FCCU interface 2281
66.4.7 Watchdogs 2281
66.4.8 CRC 2282
66.4.9 SSCM interface 2282
66.5 Register description 2282
66.5.1 STCU2 Run Register (STCU_RUN) 2284
66.5.2 STCU2 Run Software Register (STCU_RUNSW) 2285
66.5.3 STCU2 SK Code Register (STCU_SKC) 2285
66.5.4 STCU2 Configuration Register (STCU_CFG) 2287
66.5.5 STCU2 PLL Configuration Register (STCU_PLL_CFG) 2289
66.5.6 STCU2 Watchdog Register Granularity (STCU_WDG) 2290
66.5.7 STCU2 CRC Expected Status Register (STCU_CRCE) 2291
66.5.8 STCU2 CRC Read Status Register (STCU_CRCR) 2292
66.5.9 STCU2 Error Register (STCU_ERR_STAT) 2293
66.5.10 STCU2 Error FM Register (STCU_ERR_FM) 2295
66.5.11 STCU2 Offline LBIST Status Register (STCU_LBS) 2296
66.5.12 STCU2 Offline LBIST End Flag Register (STCU_LBE) 2297
66.5.13 STCU2 Online LBIST Status Register (STCU_LBSSW) 2298
66.5.14 STCU2 Online LBIST End Flag Register (STCU_LBESW) 2298

66.5.15 STCU2 LBIST Unrecoverable FM Register (STCU_LBUFM) 2299
66.5.16 STCU2 Offline MBIST Status Low Register (STCU_MBSL) 2300
66.5.17 STCU2 Offline MBIST Status Medium Register (STCU_MBSM) . . . 2300
66.5.18 STCU2 Offline MBIST Status High Register (STCU_MBSH) 2301
66.5.19 STCU2 Offline MBIST End Flag Low Register (STCU_MBEL) 2302
66.5.20 STCU2 Offline MBIST End Flag Medium Register (STCU_MBEM) . 2302
66.5.21 STCU2 Offline MBIST End Flag High Register (STCU_MBEH) 2303
66.5.22 STCU2 Online MBIST Status Low Register (STCU_MBSLSW) 2304
66.5.23 STCU2 Online MBIST Status Medium Register (STCU_MBSMSW) . 2304
66.5.24 STCU2 Online MBIST Status High Register (STCU_MBSHSW) . . . 2305
66.5.25 STCU2 Online MBIST End Flag Low Register (STCU_MBELSW) 2306
66.5.26 STCU2 Online MBIST End Flag Medium Register (STCU_MBEMSW) 2306
66.5.27 STCU2 Online MBIST End Flag High Register (STCU_MBEHSW) 2307
66.5.28 STCU2 MBIST Unrecoverable FM Low Register (STCU_MBUFML) 2308
66.5.29 STCU2 MBIST Unrecoverable FM Medium Register (STCU_MBUFMM) 2309
66.5.30 STCU2 MBIST Unrecoverable FM High Register (STCU_MBUFMH) 2310
66.5.31 STCU2 LBIST Control n Register (STCU_LB_CTRLn) 2311
66.5.32 STCU2 LBIST PC Stop n Register (STCU_LB_PCSn) 2313
66.5.33 STCU2 LBIST PRPG Low n Register (STCU_LB_PRPGLn) 2314
66.5.34 STCU2 LBIST PRPG High n Register (STCU_LB_PRPGHn) 2315
66.5.35 STCU2 Offline LBIST MISR Expected Low n Register (STCU_LB_MISRELn) 2315
66.5.36 STCU2 Offline LBIST MISR Expected High n Register (STCU_LB_MISREHn) 2316
66.5.37 STCU2 Offline LBIST MISR Read Low n Register (STCU_LB_MISRRLn) 2317
66.5.38 STCU2 Offline LBIST MISR Read High n Register (STCU_LB_MISRRHn) 2317
66.5.39 STCU2 Online LBIST MISR Expected Low n Register (STCU_LB_MISRELSWn) 2318
66.5.40 STCU2 Online LBIST MISR Expected High n Register (STCU_LB_MISREHSWn) 2319

66.5.41 STCU2 Online LBIST MISR Read Low n Register (STCU_LB_MISRRLSWn) 2320
66.5.42 STCU2 Online LBIST MISR Read High n Register (STCU_LB_MISRRHSWn) 2321
66.5.43 STCU2 MBIST Control k Register (STCU_MB_CTRLk) 2322
66.6 Register map 2323
66.7 Use cases and limitations 2327
66.7.1 Offline self-test sequence 2327
66.7.2 Online self-test Sequence 2328
66.7.3 Bypass USER Mode 2330
66.7.4 Design implementation information 2330

67 Register Protection (REG_PROT) 2331
67.1 Introduction 2331
67.1.1 Overview 2331
67.1.2 Features 2331
67.1.3 Modes of operation 2332
67.2 External signal description 2332
67.3 Memory map and register definition 2332
67.3.1 Memory map 2333
67.3.2 Register descriptions 2334
67.4 Functional description 2336
67.4.1 General 2337
67.4.2 Change lock settings 2337
67.4.3 Access errors 2340
67.5 Initialization/application information 2341
67.5.1 Reset 2341

68 Password and Device Security Module (PASS) 2342
68.1 Introduction 2342
68.1.1 Overview 2342
68.1.2 Features 2342
68.1.3 Modes of operation 2342
68.2 Memory map and register definition 2343
68.2.1 Register descriptions 2344
68.3 DCF clients 2353
68.3.1 Censorship 2354

68.4 Functional description 2354
68.4.1 Censorship 2354
68.4.2 Debug Interface Access 2354
68.4.3 Flash Memory Read Protection 2355
68.4.4 UTEST protection 2356
68.4.5 Retest Password 2356
68.5 Initialization/application information 2357
68.5.1 Reset 2357
68.5.2 Setting lock bits in a password group 2357
68.5.3 Censoring and uncensoring the device 2358

Appendix A Acronyms and abbreviations 2359

Revision history 2363

1 Preface

1.1 Register conventions

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

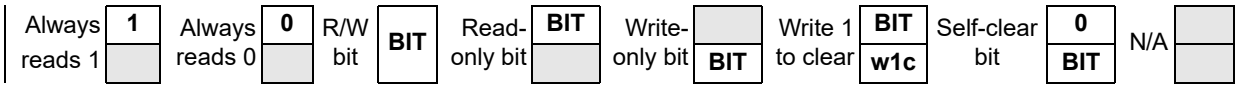


Figure 1. Key to register fields

Table 1. Register conventions

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

1.2 Acronyms and abbreviations

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

2 Introduction

2.1 SPC574Sx microcontroller

The SPC574Sx microcontroller is a member of the SPC57S family. SPC574Sx provides significant improvement over the first member of the SPC57S family in terms of performance and memory while maintaining compatibility with SPC570S50

The SPC574Sx device implements two e200z4d processor cores:

- Core_0—Main core
- Core_0s—Checker (safety) core

2.1.1 Target applications

This device is primarily targeted at the following automotive safety critical applications:

- EPS (Electric Power Steering)
- ESC (Electronic Stability Control)
- Dual 3phase BLDC FOC motor control
- Hybrid drive and DC-DC con-/inverters
- High-end airbag applications

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

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

2.1.2 Overall architecture

The architecture comprises a computational shell. The computational shell consists of two CPUs (in lock-step for safety) and fast system memory. All components of the computational shell are connected through a crossbar switch (XBAR).

Beyond the computational shell the device architecture results from a comprehensive implementation of a new generation ASILD safety concept following the SEooC approach (Safety Element out of Context).

2.1.3 Core features

The computational shell consists of an e200z420n3 CPU operating up to 140 MHz with a second isafety checker CPU running in delayed lockstep mode. The overall core, interconnect and memory subsystem is running at same frequency for optimum performance per MHz. The e200z420n3 is compatible with the Power Architecture® VLE instruction set, that supports code size reduction and embeds a floating-point unit, 8 KB instruction cache, 4 KB data cache, 32 KB data core local memory (zero latency access), a core memory protection memory unit, end-to-end error correction as well as Nexus Class3+ real time debug module. The Power Architecture has enhancements that improve the architecture's fit in embedded applications.

2.1.4 Memory hierarchy

The SPC574Sx computational shell has two levels of memory hierarchy. At the top level is system RAM (SRAM) and flash memory. Two AIPS peripheral bridges provide access to 2 independent clusters of I/O peripherals supporting safety redundancy for I/O.

2.2 Features

On-chip modules within the SPC574Sx include the following features:

- Operating parameters
  - Fully static operation, up to 140 MHz
  - Up to -40 °C to 150 °C junction temperature operating range
- Power management features
  - HALT mode — core clocks are stopped but the PLL is configurable
  - STOP mode — all clocks are stopped including the PLLs
  - Software-controlled clock gating of peripherals
- High performance, low cost e200z420 core processor
  - 32-bit CPU core complex (e200z420n3)
  - Compliant with the Power Architecture® embedded category
  - Includes an instruction set enhancement allowing variable length encoding (VLE) for code size footprint reduction. Optional encoding of mixed 16-bit and 32-bit instructions makes it possible to achieve significant code size footprint reduction.
- Advanced and flexible supply scheme
  - Internal or External regulator mode for 1.2 V supply
  - 3.3 V +/- 5% or 5 V +/- 5%
  - Up to 2 power rails for GPIOs and 2 power rails for analog/input pins enabling supply redundancy concept.
  - The 4 power rails can be supplied in independent way (depending on selected package and device configuration)
- Designed with EMI reduction techniques
  - Internal phase-locked loop
  - Frequency modulation of system clock frequency
  - On-chip regulator
  - Controlled I/O slew rate
- Advanced microcontroller bus architecture (AMBA) crossbar switch (XBAR) providing concurrent access to peripherals, Flash memory and SRAM
  - 4 master ports: FlexRay, DMA, CPU instruction bus and CPU data bus
  - 5 slave ports: Flash Controller, TCM Back-door (Port to local Data RAM), RAM controller, PBRIDGE0 and PBRIDGE1
- 32-bit internal address bus, 32-bit internal data bus
- ECC (Error Correction Code) Flash memory with Flash controller
  - Up to 1.5 MB Code Flash—single module with prefetch buffer and 256-bit data access port
  - 64 KB Data Flash—single module with prefetch buffer and 256-bit data access port
- Up to 96 KB ECC SRAM with RAM controller (in addition to 32 KB core local data RAM)
- 16 KB dedicated OTP Flash for embedded boot code
  - Boot Assisted Flash (BAF)
  - Supports internal Flash programming via a serial link (UART and MCAN)

- System timers:
  - 1 x STM (AUTOSAR®) (4 compare channels)
    1 x PIT (8 channels)
    4 x eTimers (6 channels each)
  - System watchdog timer (SWT)
    32-bit timer
    Programmable selection of system or oscillator clock for timer operation
    Programmable selection of reset or interrupt on an initial time-out
    Enabled out of reset
- Safety and integrity features:
  - Clock Monitor Unit (CMU) for safe oscillator/PLL control using internal RC oscillator reference
  - Watchdog with time window for reload
  - Memory Protection Unit (MPU): 8 regions with 32-bit granularity
  - Register protected accesses to critical peripherals
- Interrupt controller (INTC) with dedicated interrupt source channels, including software interrupts and 32 priority levels
- 12-bit analog-to-digital converter (ADC) with a conversion time of 1.5 µs at maximum operating frequency
  - 16 high-precision channels for each group of 4 ADCs
- 4 general purpose eTimer units, with 6 channels each
- 4 FlexPWM units (2 units with 4 channels each and 2 units with 2 channels each)
- Up to 2 Local Interconnect Network (LIN) controller modules capable of autonomous message handling (master), autonomous header handling (slave mode), and UART support. Compliant with LIN protocol rev. 2.1
- 4 DSPI (Deserial Serial Peripheral Interface) modules for full-duplex, synchronous, communications with external devices
- 3 MCAN (with CAN-FD support)
- Frequency-modulated phase-locked loop (FMPLL)
- Configurable general purpose pins supporting input and output operations:80 GPIO + 32 GPI/ADC input only (eTQFP144), 62 GPIO + 18 GPI/ADC input only (eTQFP100 - default bonding option)" and 64 GPIO + 16 GPI/ADC input only (eTQFP100 - bonding option upon demand)"
- Nexus development interface (NDI) per IEEE-ISTO 5001-2003 Class 3
- Device/board boundary scan testing supported with per Joint Test Action Group (JTAG) of IEEE (IEEE 1149.1)
- SPC574Sx family members are offered in the following package types:
  - 144-pin eTQFP, 0.5 mm pitch, 20 mm × 20 mm outline
  - 100-pin eTQFP, 0.5 mm pitch, 14 mm × 14 mm outline

Note: One of four sources can be used as the system clock for this device:

- *External crystal oscillator 4–40 MHz (XOSC)*
- *Internal RC oscillator 16 MHz (IRCOSC)*
- *Primary PLL*
- *Secondary PLL*

2.3 Feature summary

Table 2 lists a summary of major features for the SPC574Sx device. The feature column represents a combination of module names and capabilities of certain modules. A detailed description of the functionality provided by each on-chip module is given later in this document.

Table 2. SPC574Sx device feature summary (superset configuration)

Feature		Description
Processor	Main Core	e200z420n
	Checker core	e200z419
	Local data RAM	32 KB Data
	VLE	Yes
	Main processor frequency	140 MHz
	Instruction cache	8 KB
	Data cache	4 KB
Interrupt controller (including interrupt controller checker)		1
Software watchdog timer		1
System timers		1 AUTOSAR® STM 1 PIT with eight 16-bit channels
DMA (including DMA checker)		1
DMA channels		32
SMPU		Yes (8 regions)
System SRAM (in addition to core local data RAM)		96 KB
Code Flash memory		1.5 MB
Data Flash memory (EEPROM)		64 KB
UTEST Flash memory		16 KB
Boot assist Flash (BAF)		16 KB
CRC		1
LINFlexD		2
DSPI		4
MCAN		3
FlexRay		1 (128 MB)
SENT		2 x 3 channels
FlexPWM		2 x 4 channels (for motor control) + 2 x 2 channels (for SWG emulation via PWM)
eTimer		4 x 6 channels
ADC (SAR)		8
CTU (Cross Triggering Unit)		2

Table 2. SPC574Sx device feature summary (superset configuration) (continued)

Feature	Description
Temperature sensor	2
Self-test control unit (memory and logic BIST)	1
FCCU	1
MEMU	1
RCOSC	1
XOSC	1
PIT	1 x 8 channels
STM	1
PLL	Dual PLL with FM
Nexus	3(1)
Sequence Processing Unit (SPU)	1
External power supplies	– Single supply mode: 3.3 V or 5 V – Dual supply mode(QFP144): 3.3 V or 5 V for IOs 1.2 V for core supply
Junction temperature	-40 °C to 150 °C
Packages	eTQFP100
	eTQFP144

1. Including trace for the crossbar masters (data & instruction trace on core and data trace on eDMA). 4 MDO pin Nexus trace port for QFP100, 12 MDO pin Nexus trace port for QFP144.

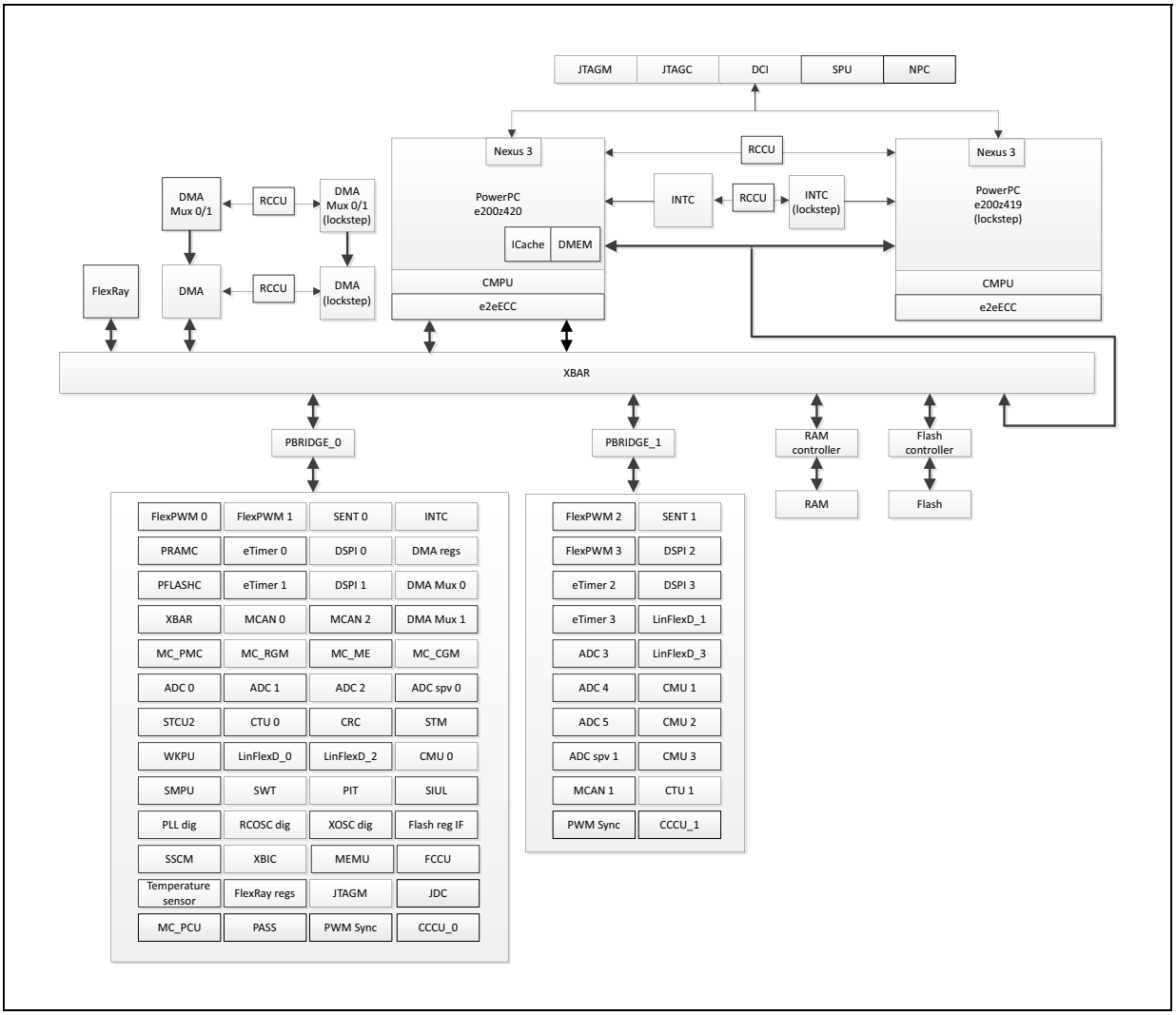
2.4 Packages

The SPC574Sx is available in three packages: eTQFP100, eTQFP144.

2.5 Block diagram

Figure 2 shows the top-level block diagram.

Figure 2. Block Diagram



3 System architecture

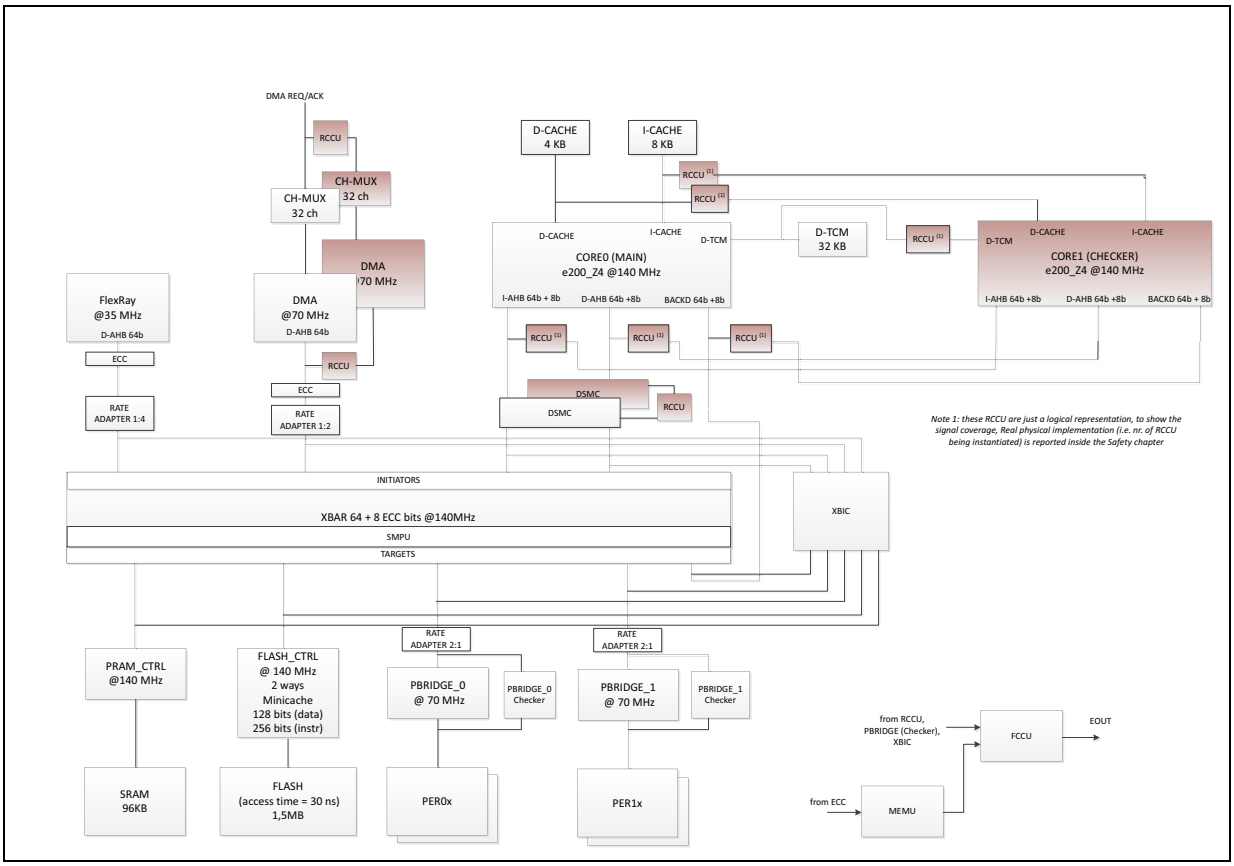
3.1 Platform Architecture

The platform architecture is organized as shown in *Figure 3*.

The brown blocks belong to the safety architecture .

Full platform including system RAM operates at speed (140 MHz) except the DMA, PBRIDGE and Flexray with fixed prescaled 1:4 ratio of the CPU frequency.

Figure 3. Platform block diagram



The platform is organized as a multi-master system, which shares a set of slave resources, accessible through a XBAR interconnection system.

3.1.1 XBAR Masters

Following IP acts as XBAR system master:

- CPU
- DMA
- FlexRay

Table 3 shows the XBAR master port assignment and the corresponding AHB master ID

Table 3. XBAR physical master port assignment

Master port number	Master	AHB Master ID
0	e200z420 CPU IBUS	0 or 8 (1)
1	e200z420 CPU DBUS	0 or 8
2	DMA	3
3	FlexRay	5

1. HMASTER = 0 when the CPU access in Normal Mode;
HMASTER = 8 when used the CPU access in Debug Mode

3.1.1.1 XBAR Slaves

Following IP are connected as XBAR system slaves:

- PBRIDGE_0
- PBRIDGE_1
- Flash Controller
- RAM Controller
- TCM Back-door

Table 4 shows the XBAR slave port assignment.

Table 4. XBAR slave port assignment

Slave port number	IP
0	Flash Controller Port0
1	CPU TCM backdoor
2	RAM Controller
3	PBRIDGE_0
4	PBRIDGE_1
5..7	Reserved

3.1.1.2 Safety architecture

Several IP and architecture solutions exist within the platform, aimed to target ASIL-D ISO26262 compliance. Following IPs are instantiated, as part of the general safety architecture :

- SMPU
- RCCU
- XBIC

3.1.1.3 Additional platform IPs

Following IPs are also part of the platform:

- Interrupt controller
- System watchdog
- DSMC

3.1.2 Functionality

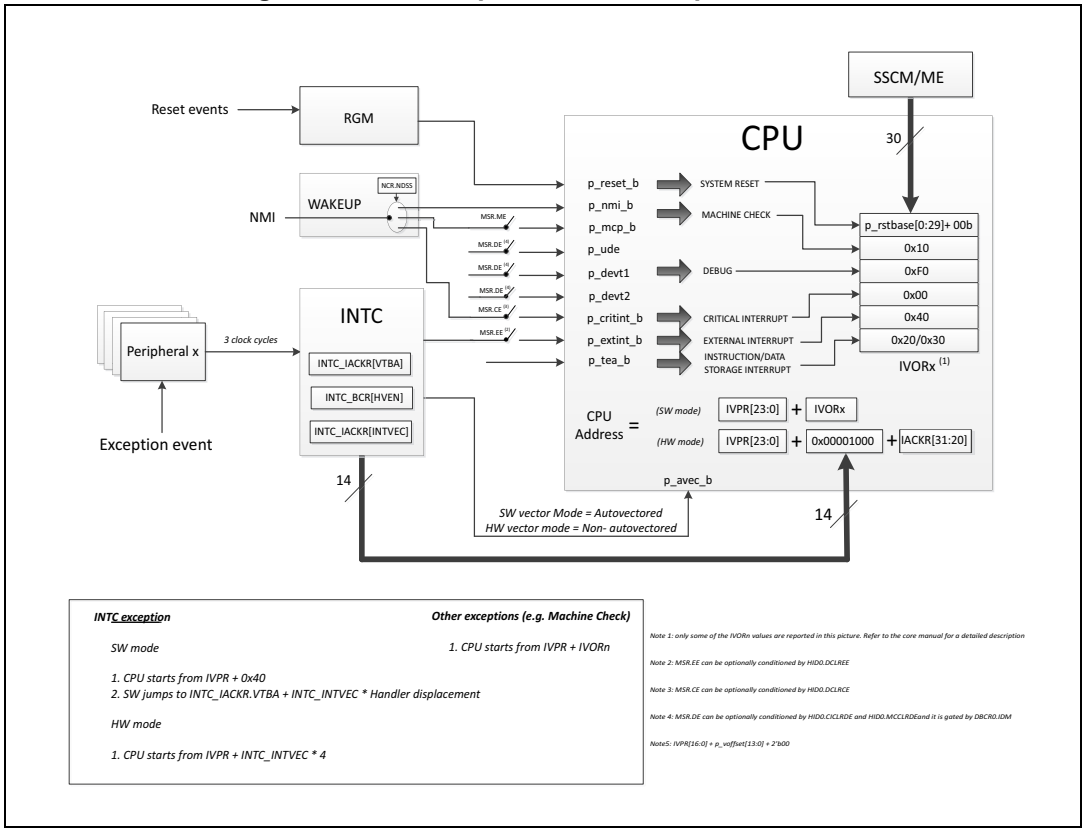
3.1.2.1 CPU exception system

Figure 4 shows how the CPU exception system mechanism and how the interrupts triggered by the interrupt controller are propagated to the CPU and handled in SW.

The CPU can serve different exceptions, triggered by CPU-internal events (for example an invalid instruction opcode) or CPU-external events (CPU input signals). For each event type, a SW exception handler can be assigned. The space allocated for each exception is 16-bytes memory space. The memory address assignment of each slot is hard-wired, starting form a base address, which is defined by the content of the CPU IVPR register. Modifying this registers, all the exception handlers can be remapped to a different memory location. The HW-wired offsets for each exception handlers can be read form the read-only IVORn. The reset exception is actually handled in a different way: the CPU restarts from the address provided by the SSCM (at the reset) or the ME (upon a run-time mode change) to the CPU. All the interrupts coming from the INTC are assigned to the “External Interrupt” CPU exception (IVOR4). When the INTC triggers the CPU External Interrupt exception signal, the CPU restarts from the address specified by the IVPR register, displaced by the HW-wired content of the IVOR4 register.

The INTC interrupt can be recognized by reading the INTC_IACKR register, which contains the interrupt handler base address and the number of the interrupt channel, which caused the interrupt triggering.

Figure 4. CPU Exceptions vs. Interrupt Controller



3.1.3 Performances

Table 5 includes the wait states number for each transaction of the different initiator versus the related target:

Table 5. Platform performances

Initiator	Target	Access	WS (master IF) with Slave Parked	WS (master IF) w/o Slave parked	Notes
DMA	SRAM	READ 8/16/32/64	1	2	1 WS is the slave is not parked on the Master
		WRITE 64	1	2	1 WS is the slave is not parked on the Master
		WRITE 8/16/32	2	3	1 WS is the slave is not parked on the Master 1 WS for ECC re-computation @ 64 bits -> RMW
	DTCM	READ 8/16/32/64	2	3	1 WS taken by rate adaptor when passing htrans to Core + 1 WS Core Slave introduce to provide hready to Rate adaptor
		WRITE 64/32	2	3	—
		WRITE 8/16	3	4	1 WS for ecc re-computation
Z4 D-AHB	SRAM	READ 8/16/32/64	0	1	—
		WRITE 64	0	1	—
		WRITE 8/16/32	1	2	1 WS for ECC re-computation @ 64 bits -> RMW
		WRAP4 READ 8/16/32/64	0,0,0,0	1,0,0,0	—
		WRAP4 WRITE 64	0,0,0,0	1,0,0,0	—
Z4	DTCM	READ 8/16/32/64	0	—	direct (no arbitration).
		WRITE 64/32	0	—	direct (no arbitration)
		WRITE 8/16	2	—	direct (no arbitration) 1 WS for read 1 WS for ECC re-computation @ 64 bits and write

Table 5. Platform performances (continued)

Initiator	Target	Access	WS (master IF) with Slave Parked	WS (master IF) w/o Slave parked	Notes
Z4 D-AHB	DTCM	READ 8/16/32/64	0	—	remapping + direct
		WRITE 64/32	0	—	remapping + direct
		WRITE 8/16	2	—	remapping + direct 1 WS for read 1 WS for ECC re-computation @ 64 bits and write
	FLASH	READ 32/64	5	6	minicache miss (pflash_ctl.pfcr1.RWSC = 4) access time = 30 ns
		READ 32/64	0	1	minicache hit (minicache= 256 bits) access time = 30 ns
		WRAP4 READ 64	5,0,0,0	6,0,0,0	minicache miss (pflash_ctl.pfcr1.RWSC = 4) access time = 30 ns (With First Miss, other 3 subsequent accesses will be hit)
		WRAP4 READ 32	5,0,0,0	6,0,0,0	minicache miss (pflash_ctl.pfcr1.RWSC = 4) access time = 30 ns (With First Miss, other 3 subsequent accesses will be hit)
		WRAP4 READ 32/64	0,0,0,0	1,0,0,0	minicache hit (minicache= 256 bits) access time = 30 ns
Z4 I-AHB	FLASH	READ 32/64	5	—	minicache miss (pflash_ctl.pfcr1.RWSC = 4) access time = 30 ns
		READ 32/64	0	—	minicache hit (minicache = 256 bits) access time = 30 ns
		WRAP4 READ 32/64	5,0,0,0	—	minicache miss (pflash_ctl.pfcr1.RWSC = 4) access time = 30 ns (With First Miss, other 3 subsequent accesses will be hit)
		WRAP4 READ 32/64	0,0,0,0	—	"minicache hit (minicache= 256 bits) access time = 30 ns"

Table 5. Platform performances (continued)

Initiator	Target	Access	WS (master IF) with Slave Parked	WS (master IF) w/o Slave parked	Notes
Z4 D-AHB	PBRIDGE	READ 8/16/32	7	8	—
		WRITE 8/16/32	7	8	—

3.2 Embedded memories

The embedded memory architecture for SPC574Sx devices is a significant departure from previous-generation devices. Features include:

- Onboard SRAM, including system SRAM, local data memory for processor core
- Onboard system error correction code (ECC) flash memory
- End-to-end ECC (e2eECC) error detection and correction
- Embedded memories in the peripherals[(a)]:
  - CAN
  - eDMA
  - Flexray

3.2.1 SRAM

3.2.1.1 System memory

SPC574Sx devices include 96 KB of general-purpose on-chip ECC SRAM.

- See *Chapter 30: Platform RAM controller (PRAM)* for details on configuring the PRAM controller.
- See *Table 15: Memory map (XBAR slave ports)* in *Chapter 5: Memory map* for the SPC574Sx SRAM address range.

3.2.1.2 Processor core local memory (D-MEM)

SPC574Sx devices include local data SRAM (D-MEM) for processor core to provide enhanced performance.

Processor core has 32 KB of D-MEM.

Each area of D-MEM is accessible by bus mastering peripheral devices.

3.2.1.3 Processor core data cache memory

Data cache memory size is 4 KB.

3.2.1.4 Processor core instruction memory

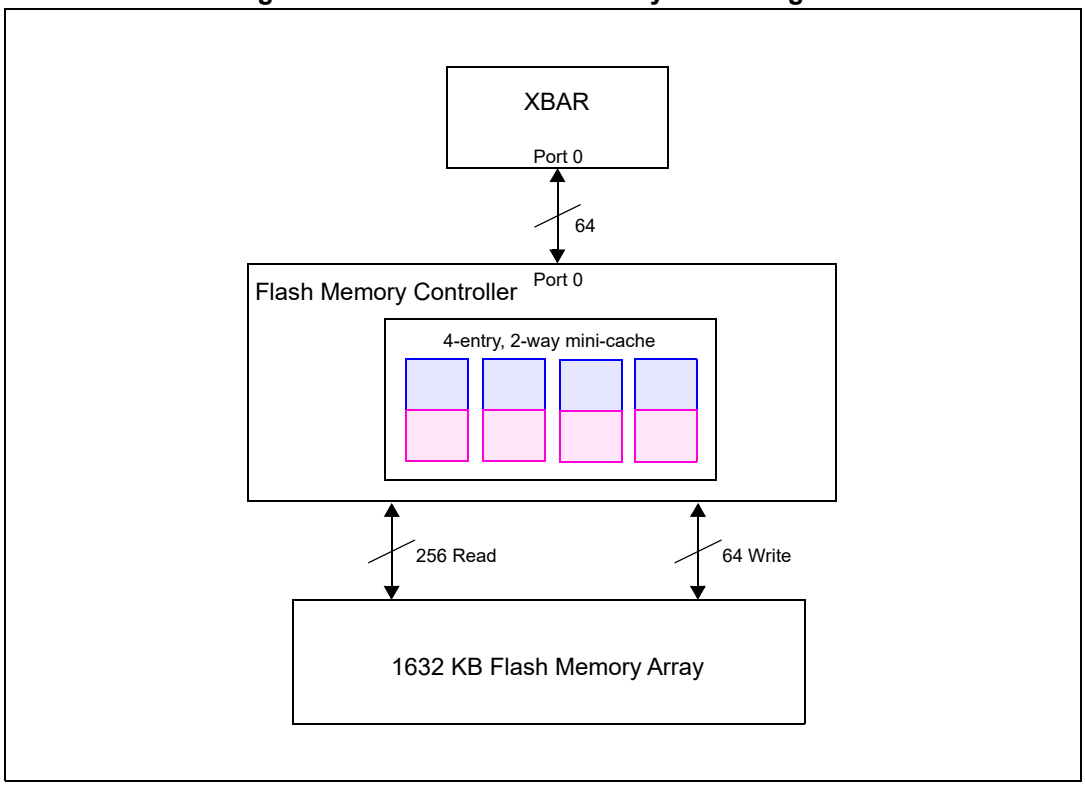
Instruction cache memory size is 8 KB.

a. Refer to the respective module chapter for details.

3.2.2 Flash memory

Flash memory on SPC574Sx devices consists of a flash memory controller and a flash memory array module. The flash controller provides flash configuration and control functions and manages the interface between the flash memory array and the device crossbar switch. *Figure 5* shows the memory architecture.

Figure 5. SPC574Sx flash memory block diagram



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

3.2.2.1 Flash memory controller

The SPC574Sx flash memory controller acts as an interface between the system bus and the flash memory array.

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

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

See *Chapter 31: Flash memory controller (PFLASH Controller)*, for details.

3.2.2.2 Flash memory array

The first partition in the array includes 1536 KB of Code Flash and a 32 KB one-time programmable (OTP) region comprising UTEST (16 KB) and BAF (16 KB), included to support systems that require non-volatile memory for security features or system initialization information. The second 64-KB partition is EE-Emulated Data Flash.

Reads of the embedded flash memory return a 256-bit page of data that may be buffered in the mini-cache in the flash memory controller. Programming of the flash may be done by double word (64 bits). Flash memory is erased on a block by block basis.

3.2.2.2.1 Features

- Flash segmentation
  - 4x 16 KB, 2x 32 KB, 2x 64 KB, and 5x 256 KB blocks for code
  - 4x 16 KB EEPROM emulation blocks for data
  - Support for reading-while-writing when the accesses are to different partitions
- Flash memory protection: write protection and OTP function available for each block
- Test and initialization data stored in a non-volatile 16 KB OTP UTEST and BAF block
- ECC with double bit detection, single bit correction
- Erase suspend, program suspend, and erase-suspended program all supported

3.2.2.2.2 Flash organization

The embedded flash memory consists of address spaces in groupings:

- 16 KB block in low address space, 32 KB block in low address space and 64 KB block in low address space
- 16 KB block in high address space
- 256 KB block in 256 KB address space

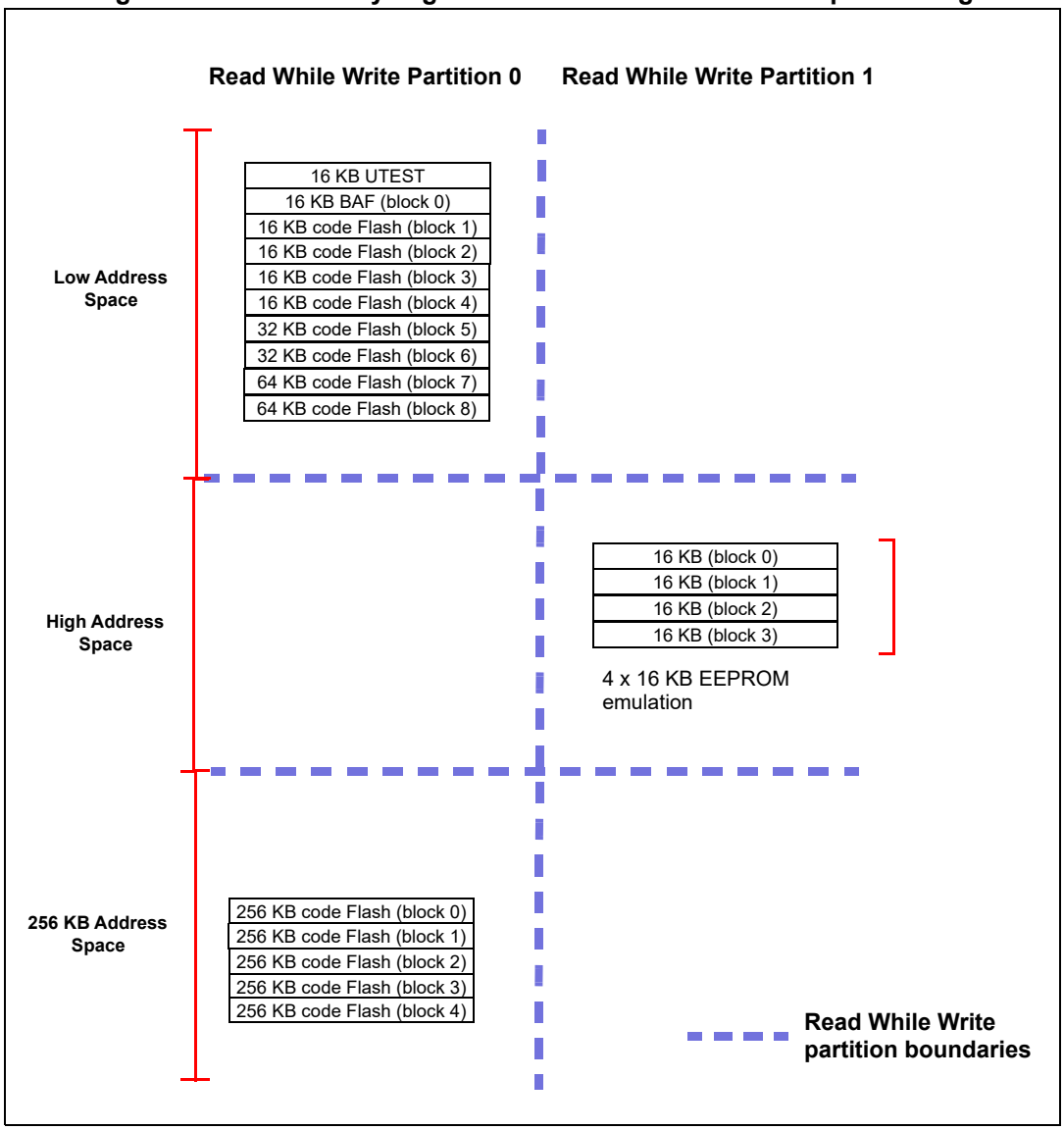
Address space group (low- or high-address space) determines which registers/fields are used to select blocks for modify operations or lock them from modify operations. Each address space is independent.

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

- For program operations, only the address specified by an interlock write determines the partition being written (block locking and block select registers do not determine the RWW partitions being written).
- For erase operations, only blocks that are selected and unlocked determine the RWW partitions being written.

Figure 6 shows the flash block segmentation and read-while-write partitioning for SPC574Sx devices.

Figure 6. Flash memory segmentation and read-while-write partitioning



Note: See Table 16: Flash memory map in Chapter 5 for the SPC574Sx flash memory map.

3.2.2.2.3 UTEST memory space

Devices in the SPC574Sx family contain a 16 KB area of OTP flash memory for storing test information and device configuration data. See *Table 19: UTEST memory map in Chapter 5* for the SPC574Sx UTEST flash memory map.

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

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

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

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

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

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

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

Traditional ECC	End-to-End ECC (e2eECC)
Bus master initiates data write	Bus master initiates data write and generates ECC checkbits based on 29-bit address and 64-bit data fields for the computational shell or 32-bit data field for the peripheral shell
Data write transfer routed from bus master to appropriate bus slave	Data write transfer (including checkbits) routed from bus master to appropriate bus slave
For a bus memory slave, generate the ECC checkbits based on the data value and store data + checkbits into the memory	For a bus memory slave, store data and checkbits into the memory
Bus master initiates data read of previously written memory location	Bus master initiates data read of previously written memory location

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

Traditional ECC	End-to-End ECC (e2eECC)
Data read transfer routed from bus master to appropriate bus slave	Data read transfer routed from bus master to appropriate bus slave
For a bus memory slave, the memory array is accessed, the controller performs the ECC checkbit decode and syndrome generation, performs any needed single-bit correction and drives the read data onto the system bus interconnect	For a bus memory slave, the memory array is accessed, and the controller passes the read data and associated checkbits onto the system bus interconnection
The bus master captures the read data and continues	The bus master captures the read data and associated checkbits, performs the ECC checkbit decode and syndrome generation, performs any needed single-bit correction and continues

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

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

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

Errors that occur within the system interconnect are typically manifested as an incorrect address, incorrect write data, or incorrect read data to be presented to the slave or back to the master. Errors occurring within the storage typically manifest themselves eventually in corrupted read data or checkbits being returned to a requester. While not all possible errors can be detected this way, this approach provides a substantial improvement versus traditional methods. Additional on-line diagnostics, or additional hardware, or both should be able to catch a majority of the errors not covered directly by the e2eECC scheme.

3.2.4 Security features

The SPC574Sx architecture uses a combination of device life cycle monitoring and password protection to limit the conditions under which debugging interfaces can access certain areas of flash memory.

3.3 Motor control Architecture

The device offers the capability to control 2 synchronized three-phases motor. *Figure 7* shows the motor control architecture, which is composed by the following elements:

- SARADC
  - 8 ADC, divided into two groups, each one comprising 3 ADC + 1 supervisor ADC
  - 6+5+5 functional channels (fast mode) per ADC group [b]
  - 16 supervision channels (fast mode) per ADC group
  - User test channels (slow mode) for self-test capabilities (VREFH, VREFH/3, 2*VREFH/3, TSENS, VREFL, VSS_ADC, LV HP, LV VSS).
  - 3.3 V or 5 V ADC supply (dedicated IO ring) per ADC group
  - Resolution: 12 bit
  - Conversion time (fast) = 1.5 µs at12 MHz
  - Conversion time (slow) > 2 µs at 12 MHz
  - Leakage current to ADC input = 300 nA (shared channels)
  - For each ADC: 4 configurable analog watchdogs for each ADC
- CTU (cross triggering unit)
  - 2 CTU, one per each ADC group
  - Up to 4 ADC can be controlled by each single CTU
  - 8 sequential triggers based on trigger CMP registers (double buffered)
  - Command list (24 descriptors double buffered) linked to the sequential trigger
  - Max 32 ADC channels per unit
  - Single/dual conversion
  - Max 8 FIFOs for data buffering
  - Max 8 + 1 DMA channels for data transfer and control
- PWM
  - One dedicated FlexPWM unit per each ADC group
  - Up to 3+3 independent or complementary outputs for each FlexPWM unit
  - 1 channel dedicated for CTU control per each PWM unit
  - 16 bits counter registers
  - 120 MHz clock (1-128 pre-scaler)
  - Duty cycle 0-100%
  - Independent control of both edges
  - Independent dead time
  - Center/phase shifted/edge aligned PWM
  - Safe mode/fault protection (fault inputs)
  - External synchronization (timer, external trigger) [c]

b. 6+6+6 channels available as bonding option on QFP100 only

c. EXT SYNC signals can be synchronized or mutually delayed

Figure 7. Motor control architecture

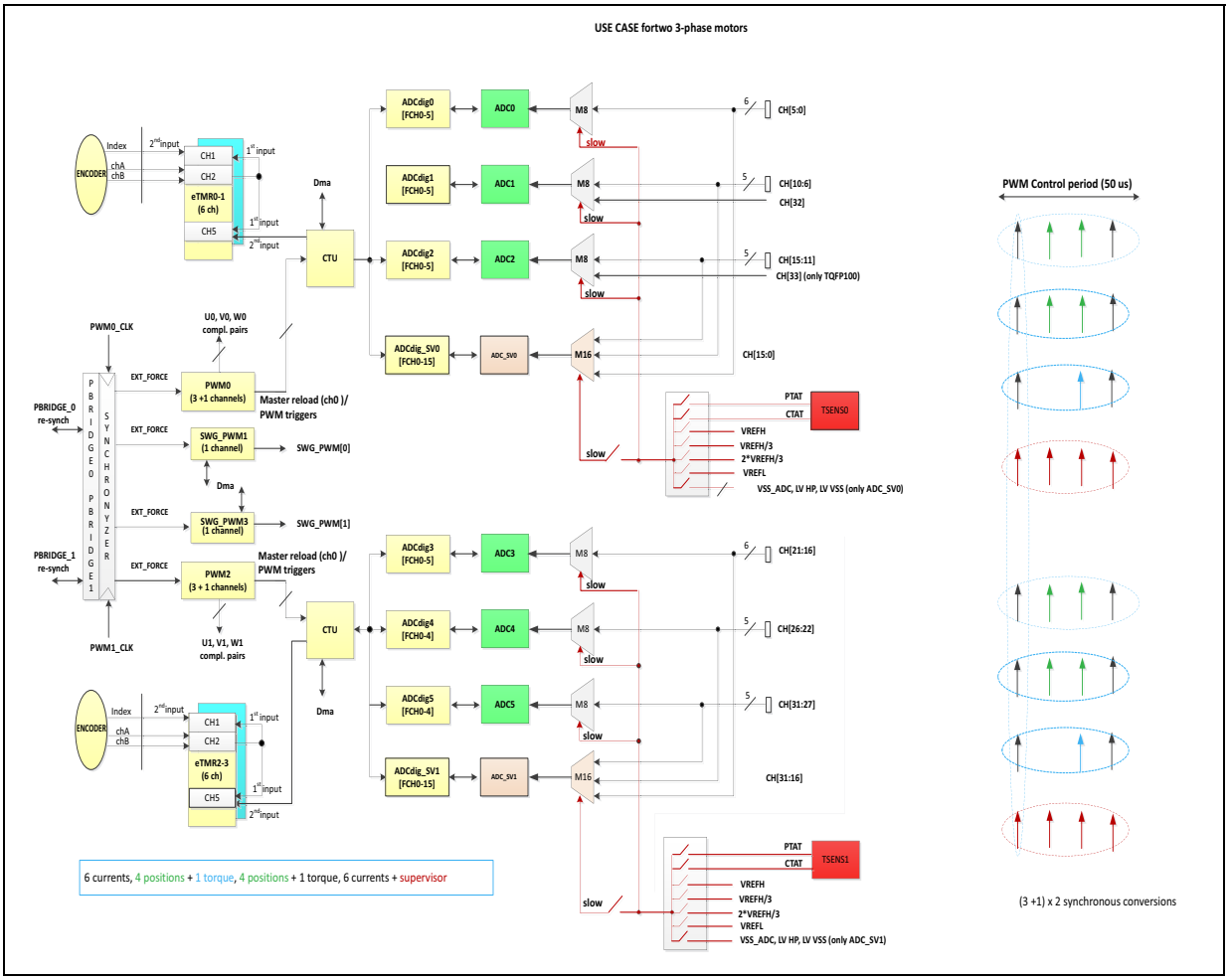


Figure 7 shows the motor control architecture.

The motor control phases are generated by two PWM. The PWM are synchronized (in phase or delayed in phase) by an external synchronizer.

The two ADC clusters, composed of 3 ADC plus 1 supervisor each, are controlled by two CTU units. Up to 4 conversions can be executed in parallel for each ADC cluster. The CTU can schedule a number of sequential conversions for each ADC and retrieve the converted data into separate FIFOs.

The PWM generates an external trigger, which is used to synchronize the CTU master-reload signal with the PWM output waveforms. This way, the ADC conversions are also synchronized with the motor phases. The CTU can periodically trigger a DMA to transfer the converted data (which are stored within the CTU FIFOs) into memory and store in the CTU the next commands for the next set of conversions. Refer to *Table 7* for ADC channels mapping description.

Table 7. ADC channels mapping description

ADC unit	Channel (F=func/ T=test)	Internal source	PAD QFP144	PAD TQFP100	Notes
ADC0	FCH0	—	PAD[94]	PAD[18]	R-series shorted
	FCH1	—	PAD[22]	PAD[22]	—
	FCH2	—	PAD[23]	PAD[23]	—
	FCH3	—	PAD[24]	PAD[24]	—
	FCH4	—	PAD[95]	PAD[19]	R-series shorted
	FCH5	—	—	—	—
	TCH120	TSENS0 (PTAT)	—	—	—
	TCH121	TSENS0 (CTAT)	—	—	—
	TCH124	VREFL	—	—	—
	TCH125	VREFH/3	—	—	—
	TCH126	VREFH*2/3	—	—	—
	TCH127	VREFH	—	—	—

Table 7. ADC channels mapping description (continued)

ADC unit	Channel (F=func/ T=test)	Internal source	PAD QFP144	PAD TQFP100	Notes
ADC1	FCH0	—	PAD[26]	PAD[26]	—
	FCH1	—	PAD[27]	PAD[27]	—
	FCH2	—	PAD[28]	PAD[28]	—
	FCH3	—	PAD[96]	PAD[20]	R-series shorted
	FCH4	—	PAD[29]	PAD[29]	—
	FCH5	—	—	PAD[129]	R-series shorted
	TCH120	TSENS0 (PTAT)	—	—	—
	TCH121	TSENS0 (CTAT)	—	—	—
	TCH124	VREFL	—	—	—
	TCH125	VREFH/3	—	—	—
	TCH126	VREFH*2/3	—	—	—
	TCH127	VREFH	—	—	—
ADC2	FCH0	—	PAD[30]	PAD[30]	—
	FCH1	—	PAD[31]	PAD[31]	—
	FCH2	—	PAD[97]	PAD[21]	R-series shorted
	FCH3	—	PAD[32]	PAD[32]	—
	FCH4	—	PAD[33]	PAD[33]	—
	FCH5	—	—	PAD[130]	—
	TCH120	TSENS0 (PTAT)	—	—	—
	TCH121	TSENS0 (CTAT)	—	—	—
	TCH124	VREFL	—	—	—
	TCH125	VREFH/3	—	—	—
	TCH126	VREFH*2/3	—	—	—
	TCH127	VREFH	—	—	—

Table 7. ADC channels mapping description (continued)

ADC unit	Channel (F=func/ T=test)	Internal source	PAD QFP144	PAD TQFP100	Notes
ADC_SV0	FCH0	—	PAD[94]	PAD[18]	R-series shorted
	FCH1	—	PAD[22]	PAD[22]	—
	FCH2	—	PAD[23]	PAD[23]	—
	FCH3	—	PAD[24]	PAD[24]	—
	FCH4	—	PAD[95]	PAD[19]	R-series shorted
	FCH5	—	—	—	—
	FCH6	—	PAD[26]	PAD[26]	—
	FCH7	—	PAD[27]	PAD[27]	—
	FCH8	—	PAD[28]	PAD[28]	—
	FCH9	—	PAD[96]	PAD[20]	R-series shorted
	FCH10	—	PAD[29]	PAD[29]	—
	FCH11	—	PAD[30]	PAD[30]	—
	FCH12	—	PAD[31]	PAD[31]	—
	FCH13	—	PAD[97]	PAD[21]	R-series shorted
	FCH14	—	PAD[32]	PAD[32]	—
	FCH15	—	PAD[33]	PAD[33]	—
	TCH104	LV_HP	—	—	—
	TCH105	LV_VSS	—	—	—
	TCH109	VSS_HV_ADC	—	—	—
	TCH120	TSENS0 (PTAT)	—	—	—
	TCH121	TSENS0 (CTAT)	—	—	—
	TCH124	VREFL	—	—	—
	TCH125	VREFH/3	—	—	—
	TCH126	VREFH*2/3	—	—	—
	TCH127	VREFH	—	—	—

Table 7. ADC channels mapping description (continued)

ADC unit	Channel (F=func/ T=test)	Internal source	PAD QFP144	PAD TQFP100	Notes
ADC3	FCH0	—	PAD[98]	—	—
	FCH1	—	PAD[99]	—	—
	FCH2	—	PAD[100]	—	—
	FCH3	—	PAD[101]	—	—
	FCH4	—	PAD[102]	—	—
	FCH5	—	PAD[103]	—	—
	TCH120	TSENS1 (PTAT)	—	—	—
	TCH121	TSENS1 (CTAT)	—	—	—
	TCH124	VREFL	—	—	—
	TCH125	VREFH/3	—	—	—
	TCH126	VREFH*2/3	—	—	—
	TCH127	VREFH	—	—	—
ADC4	FCH0	—	PAD[104]	—	—
	FCH1	—	—	—	—
	FCH2	—	PAD[106]	—	—
	FCH3	—	PAD[107]	—	—
	FCH4	—	PAD[108]	—	—
	TCH120	TSENS1 (PTAT)	—	—	—
	TCH121	TSENS1 (CTAT)	—	—	—
	TCH124	VREFL	—	—	—
	TCH125	VREFH/3	—	—	—
	TCH126	VREFH*2/3	—	—	—
	TCH127	VREFH	—	—	—

Table 7. ADC channels mapping description (continued)

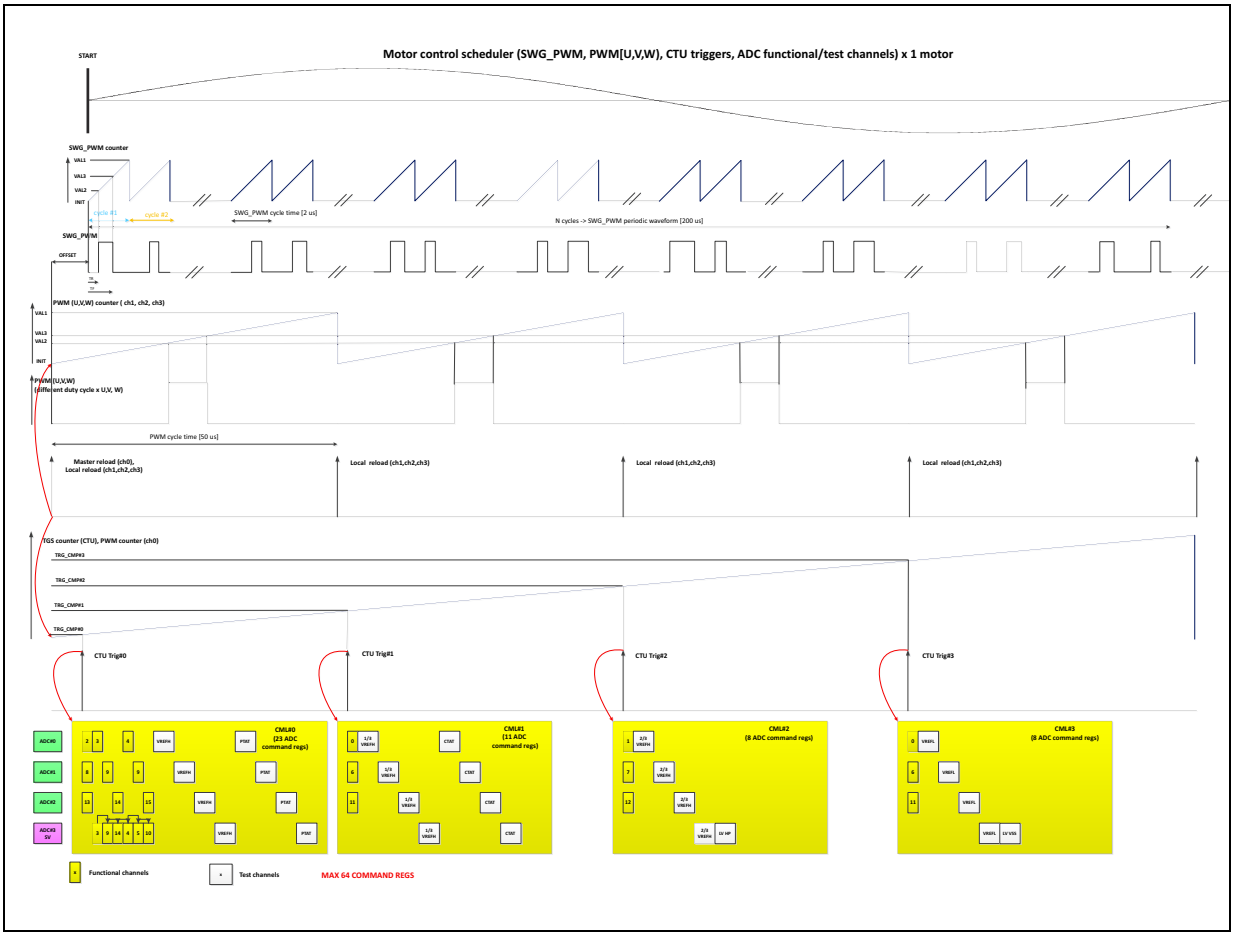
ADC unit	Channel (F=func/ T=test)	Internal source	PAD QFP144	PAD TQFP100	Notes
ADC5	FCH0	—	PAD[109]	—	—
	FCH1	—	PAD[110]	—	—
	FCH2	—	PAD[111]	—	—
	FCH3	—	PAD[112]	—	—
	FCH4	—	PAD[113]	—	—
	TCH120	TSENS1 (PTAT)	—	—	—
	TCH121	TSENS1 (CTAT)	—	—	—
	TCH124	VREFL	—	—	—
	TCH125	VREFH/3	—	—	—
	TCH126	VREFH*2/3	—	—	—
	TCH127	VREFH	—	—	—

Table 7. ADC channels mapping description (continued)

ADC unit	Channel (F=func/ T=test)	Internal source	PAD QFP144	PAD TQFP100	Notes
ADC_SV1	FCH0	—	PAD[98]	—	—
	FCH1	—	PAD[99]	—	—
	FCH2	—	PAD[100]	—	—
	FCH3	—	PAD[101]	—	—
	FCH4	—	PAD[102]	—	—
	FCH5	—	PAD[103]	—	—
	FCH6	—	PAD[104]	—	—
	FCH7	—	—	—	—
	FCH8	—	PAD[106]	—	—
	FCH9	—	PAD[107]	—	—
	FCH10	—	PAD[108]	—	—
	FCH11	—	PAD[109]	—	—
	FCH12	—	PAD[110]	—	—
	FCH13	—	PAD[111]	—	—
	FCH14	—	PAD[112]	—	—
	FCH15	—	PAD[113]	—	—
	TCH104	LV_HP	—	—	—
	TCH105	LV_VSS	—	—	—
	TCH109	VSS_HV_ADC	—	—	—
	TCH120	TSENS1 (PTAT)	—	—	—
	TCH121	TSENS1 (CTAT)	—	—	—
	TCH124	VREFL	—	—	—
	TCH125	VREFH/3	—	—	—
	TCH126	VREFH*2/3	—	—	—
	TCH127	VREFH	—	—	—

Figure 8 shows an example of a motor control waveforms and conversions.

Figure 8. Example of motor control scheduler



The ADC power supply and references are separated on different pins).

The torque sensors of each motor is controlled by two DSPI.

The DSPI are mapped on different PBRIDGE (*Table 17: Peripheral bridge 1 memory map* and *Table 18: Peripheral bridge 0 memory map*) and the I/O pins can be allocated to specific AF so that they do not interfere (see *Table 13: Peripheral multiplexing table description* for a list of the AF available for each pin).

Safety features of the ECTU (pending patent approval) are:

1. Trigger selection (ADC unit, timer unit) and timing miss
2. ADC channel miss
3. ADC FIFO channel miss
4. ADC ack and timing miss
5. CTU error

which are occurring in application mode based on the programmed configuration of the scheduler. The reaction is a FCCU trigger.

4 Signal description

4.1 Pin descriptions

The following sections provide signal descriptions and related information about the functionality and configuration of the SPC574Sx devices.

4.1.1 Pin startup and reset states

Table 8 provides startup state and reset state information for device pins.

Table 8. Pin startup and reset states

Pin	Startup state(1)	State during reset	State after reset
I/O pins(2)	High-impedance	High-impedance	High-impedance
Analog Inputs(3)	High-impedance	High-impedance	High-impedance
F0 (ERROR0)	High-impedance	Input, weak pullup	Input, weak pullup
F1 (ERROR1)	High-impedance	Input, weak pullup	Input, weak pullup
PORST	Input, pulldown	Input, weak pulldown	Input, weak pulldown
PWMODE	Weak pulldown	Weak pulldown	Weak pulldown
TESTMODE(4)	Weak pulldown	Input, weak pulldown	Input, weak pulldown
TDI	weak pullup	Input, weak pullup	Input, weak pullup
TDO	High-impedance	High-impedance	High-impedance
TMS	weak pullup	Input, weak pullup	Input, weak pullup
TCK	High-impedance	Input, weak pulldown	Input, weak pulldown
XTAL/EXTAL(5)	High-impedance	High-impedance	High-impedance

1. Startup state is exited when the core and high-voltage supplies reach minimum levels as defined in the Power Management chapter.
2. Pins on the device that are not supply or reference are not explicitly defined in this table.
3. Pure analog inputs, on digital output function.
4. Device does not implement system configuration pin. TESTMODE is the only pin affecting device configuration. It is latched on PORST rising edge and internally pull-down.
5. Function of XTAL/EXTAL pins defined by the enable bit in the flash user test row. If enabled, the pins are analog inputs to the oscillator. If disabled, the XTAL/EXTAL pins remain in high-impedance after reset.

4.1.2 Power supply and reference voltage pins

Power supply and reference pins for the device are listed in this section.

Table 9 contains information on power supply and reference pin functions for the devices.

Table 9. Power supply and reference pins

Supply			Pin No.	
Pad	**Type**	**Description**	**eTQFP100**	**eTQFP144**
VDD_LV	Power	Low voltage power supply for device (PLL is also powered by this pin.)	19, 52, 68	22, 23, 39, 47, 63, 79, 99, 126, 138
VDD_HV_IO0	Power	High voltage power supply for I/O 0 and internal power management	20, 51, 85, 95(1)	90, 103, 115, 142
VDD_HV_IO1	Power	High voltage power supply for I/O 1	—(2)	46, 59, 80
VDD_HV_OSC	Power	High voltage power supply unit and oscillator	55	43
VDD_HV_ADC_ TSENS_0	Power	High voltage supply for ADC and TSENS 0	39	18
VDD_HV_ADC_ TSENS_1	Power	High voltage supply for ADC and TSENS 1	—	38
VREFH_ADC_0	Reference	Voltage reference of ADC 0	37	16
VREFH_ADC_1	Reference	Voltage reference of ADC 1	—	36
VREFL_ADC_0	Reference	Ground reference of ADCs	38	17
VREFL_ADC_1	Reference	Ground reference of ADCs	—	37
VSS_HV	Ground	High voltage ground	Exposed pad	Exposed pad
VSS_LV	Ground	Low voltage ground		
VSS_HV_OSC_PMC	Ground	Ground supply for the internal power management unit and the oscillator		
VSS_HV_ADC_TSENS_0	Ground	Ground supply for ADC and TSENS	38	17
VSS_HV_ADC_TSENS_1	Ground	Ground supply for ADC and TSENS	—	37

1. Pin 95 is also used as high voltage supply of flash
2. IO rail split is not supported on eTQFP100, where only PWMODE = 1 is allowed

4.1.3 System pins

Table 10 contains information on system pin functions for the devices.

Table 10. System pins

Pad	Description	Direction	Pin No.	
			eTQFP100	eTQFP144
PORST	Power on reset with Schmitt trigger characteristics and noise filter. PORST is active low.	Bidirectional	66	61
PWMODE	Power Mode Connected to VSS_HV_IO on the board - select external reg for digital voltage (1.2V) Connected to VDD_HV_IO0 on board - select internal reg for digital voltage (1.2V) It has an internal pull up.	Input only	—	40
TESTMODE	Pin for testing purpose only. An internal pull-down is implemented on the TESTMODE pin to prevent the device from entering TESTMODE. It is recommended to connect the TESTMODE pin to $V_{SS_HV_IO}$ on the board. The value of the TESTMODE pin is latched at the negation of reset and has no affect afterward. The device will not exit reset with the TESTMODE pin asserted during power-up.	Input only	64	58
XTAL	Analog output of the oscillator amplifier circuit—to be grounded if the oscillator is used in bypass mode.	Output	54	42
EXTAL	– Analog input of the oscillator amplifier circuit when the oscillator is not in bypass mode – Analog input for the clock generator when the oscillator is in bypass mode	Input	53	41

4.1.4 Debug and trace pins

Table 11 contains information on debug and trace pin functions for the devices.

Table 11. Debug and trace pins

Signal	Description	Direction	Pin No.	
			eTQFP100	eTQFP144
TDI	Test Data Input	Input	59	51
TDO	Test Data Output	Output	61	54
TCK	Test Clock	Input	63	57
TMS	Test Mode Select	Input	60	53

4.1.5 Pin multiplexing

The SIUL module is used to multiplex the device input/output pins and the internal module input/output ports.

Note: Multi-function pins are programmable via their respective Multiplexed Signal Configuration Register (MSCR) Source Signal Select (SSS) values. The SSS value selects which source signal is connected to the associated destination.

For additional information, refer to Chapter 14: System Integration Unit Lite2 (SIUL2).

Table 12. Pin multiplexing table description

Port Pin	SIUL2 MSCR		Signal	Module	Full Signal Name	Direction	PAD strength				IO Segment	
	No	SSS Value					VeryStrong	Fast	Medium	Slow	eTQFP 100	eTQFP 144
PA[0]	0	0000	GPIO0	SIUL2-GPIO0	GPIO0	I/O	x	x	x	x	IO #0	IO #0
		0001	CS0	DSPI_0	DSPI 0 - Chip Select 0	I/O						
		0010	—	Reserved		—						
		0011	CS1	DSPI_1	DSPI 1 - Chip Select 1	O						
		0100	ETC0	ETIMER_0	Timer 0 - ch. 0 (IC/OC)	I/O						
		0101	SCK	DSPI_0	DSPI 0 - Serial Clock	I/O						
		0110-1111	—	Reserved	—	—						
	625	00001001	SCK	DSPI_0	DSPI0 Serial Clock Input	I						
	626	00000010	CS0	DSPI_0	DSPI0 Slave Select Input	I						
	688	00000101	ETC0	ETIMER_0	ETMR0 input channel 0	I						
	788	00000110	SENT0	SENT_0	SENT Receiver 0 Channel 0	I						
	816	00000010	REQ0	SIUL	interrupt request channel 0	I						
	980	00000101	FAULT0	PWM_0	FlexPWM 0 - FAULT[0] input channel	I						
PA[1]	1	0000	GPIO1	SIUL2-GPIO1	GPIO1	I/O	x	x	x	x	IO #0	IO #0
		0001	CS1	DSPI_1	DSPI 1 - Chip Select 1	O						
		0010	ETC0	ETIMER_0	Timer 0 - ch. 0 (IC/OC)	I/O						
		0011	IOMUX_EVTI_B00	INTG_DEBUG	Nexus EVTI	I						
		0100	ETC0	ETIMER_1	Timer 1 - ch. 0 (IC/OC)	I/O						
		0101	SOUT	DSPI_0	DSPI 0 - Serial Data	I/O						
		0110-1111	—	Reserved	—	—						
	624	00001001	SIN	DSPI_0	DSPI0 Serial Data Input	I						
	688	00000011	ETC0	ETIMER_0	ETMR0 input channel 0	I						
	694	00000101	ETC0	ETIMER_1	ETMR1 input channel 0	I						
	789	00000110	SENT1	SENT_0	SENT Receiver 0 Channel 1	I						
	981	00000101	FAULT1	PWM_0	FlexPWM 0 - FAULT[1] input channel	I						

Table 12. Pin multiplexing table description (continued)

Port Pin	SIUL2 MSCR		Signal	Module	Full Signal Name	Direction	PAD strength				IO Segment	
	No	SSS Value					VeryStrong	Fast	Medium	Slow	eTQFP 100	eTQFP 144
PA[2]	2	0000	GPIO2	SIUL2-GPIO2	GPIO2	I/O	x	x	x	x	IO #0	IO #0
		0001	CS1	DSPI_2	DSPI 2 - Chip Select 1	O						
		0010	CS4	DSPI_0	DSPI 0 - Chip Select 4	O						
		0011	EVTO_B0	DCI	Nexus EVTO	O						
		0100	ETC1	ETIMER_1	Timer 1 - ch. 1 (IC/OC)	I/O						
		0101	SOUT	DSPI_0	DSPI 0 - Serial Data	I/O						
		0110-1111	—	Reserved	—	—						
	624	00001010	SIN	DSPI_0	DSPI0 Serial Data Input	I						
	695	00000101	ETC1	ETIMER_1	ETMR1 input channel 1	I						
	790	00000110	SENT2	SENT_0	SENT Receiver 0 Channel 2	I						
PA[3]	3	0000	GPIO3	SIUL2-GPIO3	GPIO3	I/O	x	x	x	x	IO #0	IO #0
		0001	SCK	DSPI_0	DSPI 0 - Serial Clock	I/O						
		0010	—	Reserved	—	—						
		0011	ETC0	ETIMER_0	Timer 0 - ch. 0 (IC/OC)	I/O						
		0100	SCK	DSPI_1	DSPI 1 - Serial Clock	I/O						
		0101-1111	—	Reserved	—	—						
	625	00000010	SCK	DSPI_0	DSPI0 Serial Clock Input	I						
	628	00000110	SCK	DSPI_1	DSPI1 Serial Clock Input	I						
	688	00000100	ETC0	ETIMER_0	ETMR0 input channel 0	I						
	786	00000101	OS_TICK	FLEXRAY_0	Flexray Stop Watch Tick	I						
	817	00000010	REQ1	SIUL	interrupt request channel 1	I						

Table 12. Pin multiplexing table description (continued)

Port Pin	SIUL2 MSCR		Signal	Module	Full Signal Name	Direction	PAD strength				IO Segment	
	No	SSS Value					VeryStrong	Fast	Medium	Slow	eTQFP 100	eTQFP 144
PA[4]	4	0000	GPIO4	SIUL2-GPIO4	GPIO4	I/O	—	—	x	x	IO #0	IO #0
		0001	SOUT	DSPI_0	DSPI 0 - Serial Data	I/O						
		0010	—	Reserved	—	—						
		0011	ETC1	ETIMER_0	Timer 0 - ch. 1 (IC/OC)	I/O						
		0100	SOUT	DSPI_1	DSPI 1 - Serial Data	I/O						
		0101-0111	—	Reserved	—	—						
		1000	SOUT	DSPI_3	DSPI 3 - Serial Data	I/O						
		1001-1111	—	Reserved	—	—						
	561	00000111	RXD	LINFlexD_1	LinFlexD 1 Receive Data	I						
	624	00000001	SIN	DSPI_0	DSPI0 Serial Data Input	I						
	627	00000101	SIN	DSPI_1	DSPI1 Serial Data Input	I						
	633	00001010	SIN	DSPI_3	DSPI3 Serial Data Input	I						
	689	00000100	ETC1	ETIMER_0	ETMR0 input channel 1	I						
	944	00000010	NMI	WKPUFILTER_ANA	NMI input	I						
PA[5]	5	0000	GPIO5	SIUL2-GPIO5	GPIO5	I/O	x	x	x	x	IO #0	IO #0
		0001	TXD	LINFlexD_1	LINFLEX 1 - TX	O						
		0010	ETC1	ETIMER_0	Timer 0 - ch. 1 (IC/OC)	I/O						
		0011	MCKO	NPC_WRAPPER	Nexus MCK0	O						
		0100	ETC2	ETIMER_1	Timer 1 - ch. 2 (IC/OC)	I/O						
		0101-0111	—	Reserved	—	—						
		1000	CS2	DSPI_0	DSPI 0 - Chip Select 2	O						
		1001-1111	—	Reserved	—	—						
	689	00000011	ETC1	ETIMER_0	ETMR0 input channel 1	I						
	696	00000101	ETC2	ETIMER_1	ETMR1 input channel 2	I						
	944	00000101	NMI	WKPUFILTER_ANA	NMI input	I						
	984	00000101	EXT_SYNC	PWM_0	FlexPWM 0 - EXT_SYNC input channel	I						

Table 12. Pin multiplexing table description (continued)

Port Pin	SIUL2 MSCR		Signal	Module	Full Signal Name	Direction	PAD strength				IO Segment	
	No	SSS Value					VeryStrong	Fast	Medium	Slow	eTQFP 100	eTQFP 144
PA[6]	6	0000	GPIO6	SIUL2-GPIO6	GPIO6	I/O	x	x	x	x	IO #0	IO #0
		0001	—	Reserved	—	—						
		0010	ETC2	ETIMER_0	Timer 0 - ch. 2 (IC/OC)	I/O						
		0011	MDO_0	NPC_ WRAPPER	Nexus MDO 0	O						
		0100	ETC3	ETIMER_1	Timer 1 - ch. 3 (IC/OC)	I/O						
		0101	SCK	DSPI_0	DSPI 0 - Serial Clock	I/O						
		0110	ETC0	ETIMER_0	Timer 0 - ch. 0(IC/OC)	I/O						
		0111	SOUT	DSPI_1	DSPI 1 - Serial Data	I/O						
		1000	—	Reserved	—	—						
		1001	A0	PWM_0	FlexPWM 0 - A[0]	I/O						
		1010-1111	—	Reserved	—	—						
	561	00000001	RXD	LINFlexD_1	LinFlexD 1 Receive Data	I						
	625	00001000	SCK	DSPI_0	DSPI0 Serial Clock Input	I						
	627	00001001	SIN	DSPI_1	DSPI1 Serial Data Input	I						
	688	00001001	ETC0	ETIMER_0	ETMR0 input channel 0	I						
	690	00000011	ETC2	ETIMER_0	ETMR0 input channel 2	I						
	697	00000101	ETC3	ETIMER_1	ETMR1 input channel 3	I						
	952	00000101	A0	PWM_0	FlexPWM 0 - A[0] input channel	I/O						

Table 12. Pin multiplexing table description (continued)

Port Pin	SIUL2 MSCR		Signal	Module	Full Signal Name	Direction	PAD strength				IO Segment	
	No	SSS Value					VeryStrong	Fast	Medium	Slow	eTQFP 100	eTQFP 144
PA[7]	7	0000	GPIO7	SIUL2-GPIO7	GPIO7	I/O	x	x	x	x	IO #0	IO #0
		0001	SOUT	DSPI_0	DSPI 0 - Serial Data	I/O						
		0010	—	Reserved	—	—						
		0011	ETC2	ETIMER_0	Timer 0 - ch. 2 (IC/OC)	I/O						
		0100	SOUT	DSPI_1	DSPI 1 - Serial Data	I/O						
		0101	—	Reserved	—	—						
		0110	ETC1	ETIMER_1	Timer 1 - ch. 1(IC/OC)	I/O						
		0111-1000	—	Reserved	—	—						
		1001	B0	PWM_0	FlexPWM 0 - B[0]	I/O						
		1010-1111	—	Reserved	—	—						
	624	00000010	SIN	DSPI_0	DSPI0 Serial Data Input	I						
	627	00000110	SIN	DSPI_1	DSPI1 Serial Data Input	I						
	690	00000100	ETC2	ETIMER_0	ETMR0 input channel 2	I						
	695	00001001	ETC1	ETIMER_1	ETMR1 input channel 1	I						
	956	00000101	B0	PWM_0	FlexPWM 0 - B[0] input channel	I						
PA[8]	8	0000	GPIO8	SIUL2-GPIO8	GPIO8	I/O	—	—	x	x	IO #0	IO #0
		0001	CS1	DSPI_0	DSPI 0 - Chip Select 1	O						
		0010	—	Reserved	—	—						
		0011	TXD	LINFlexD_1	LINFLEX 1 - TX	O						
		0100	ETC1	ETIMER_0	Timer 0 - ch. 1 (IC/OC)	I/O						
		0101	—	Reserved	—	—						
		0110	CS0	DSPI_0	DSPI 0 - Chip Select 0	I/O						
		0111-1111	—	Reserved	—	—						
	626	00001000	CS0	DSPI_0	DSPI0 Slave Select Input	I						
	689	00000101	ETC1	ETIMER_0	ETMR0 input channel 1	I						
	982	00000101	FAULT2	PWM_0	FlexPWM 0 - FAULT[2] input channel	I						

Table 12. Pin multiplexing table description (continued)

Port Pin	SIUL2 MSCR		Signal	Module	Full Signal Name	Direction	PAD strength				IO Segment	
	No	SSS Value					VeryStrong	Fast	Medium	Slow	eTQFP 100	eTQFP 144
PA[9]	9	0000	GPIO9	SIUL2-GPIO9	GPIO9	I/O	—	—	x	x	IO #0	IO #0
		0001	CS2	DSPI_0	DSPI 0 - Chip Select 2	O						
		0010	CS7	DSPI_0	DSPI 0 - Chip Select 7	O						
		0011	—	Reserved	—	—						
		0100	ETC2	ETIMER_0	Timer 0 - ch. 2 (IC/OC)	I/O						
		0101	—	Reserved	—	—						
		0110	CS1	DSPI_0	DSPI 0 - Chip Select 1	O						
		0111-1111	—	Reserved	—	—						
	561	00000010	RXD	LINFlexD_1	LinFlexD 1 Receive Data	I						
	690	00000101	ETC2	ETIMER_0	ETMR0 input channel 2	I						
	983	00000101	FAULT3	PWM_0	FlexPWM 0 - FAULT[3] input channel	I						
PA[10]	10	0000	GPIO10	SIUL2-GPIO10	GPIO10	I/O	x	x	x	x	IO #0	IO #0
		0001	CS1	DSPI_3	DSPI 3 - Chip Select 1	O						
		0010	CS1	DSPI_1	DSPI 1 - Chip Select 1	O						
		0011	MDO_1	NPC_ WRAPPER	Nexus MDO 1	O						
		0100	—	Reserved	—	—						
		0101	A3	PWM_0	FlexPWM 0 - A[3]	I/O						
		0110	—	Reserved	—	—						
		0111	CS3	DSPI_0	DSPI 0 - Chip Select 3	O						
		1000	—	Reserved	—	—						
		1001	A0	PWM_1	FlexPWM 1 - A[0]	I/O						
		1010-1111	—	Reserved	—	—						
	819	00000101	REQ3	SIUL	interrupt request channel 2	I						
	955	00000101	A3	PWM_0	FlexPWM 0 - A[3] input channel	I						
	960	00000101	A0	PWM_1	FlexPWM 1 - A[0] input channel	I						

Table 12. Pin multiplexing table description (continued)

Port Pin	SIUL2 MSCR		Signal	Module	Full Signal Name	Direction	PAD strength				IO Segment	
	No	SSS Value					VeryStrong	Fast	Medium	Slow	eTQFP 100	eTQFP 144
PA[11]	11	0000	GPIO11	SIUL2-GPIO11	GPIO11	I/O	—	—	x	x	IO #0	IO #0
		0001	CS3	DSPI_0	DSPI 0 - Chip Select 3	O						
		0010	CS5	DSPI_0	DSPI 0 - Chip Select 5	O						
		0011	ETC3	ETIMER_0	Timer 0 - ch. 3 (IC/OC)	—						
		0100	—	Reserved	—	—						
		0101	B1	PWM_0	FlexPWM 0 - B[1]	I/O						
		0110	—	Reserved	—	—						
		0111	ETC3	ETIMER_3	Timer 3 - ch. 3 (IC/OC)	I/O						
		1000-1111	—	Reserved	—	—						
	691	00000100	ETC3	ETIMER_0	ETMR0 input channel 3	I						
	709	00001000	ETC3	ETIMER_3	ETMR3 input channel 3	I						
	820	00000101	REQ4	SIUL	interrupt request channel 2	I						
	957	00000101	B1	PWM_0	FlexPWM 0 - B[1] input channel	I						

Table 12. Pin multiplexing table description (continued)

Port Pin	SIUL2 MSCR		Signal	Module	Full Signal Name	Direction	PAD strength				IO Segment	
	No	SSS Value					VeryStrong	Fast	Medium	Slow	eTQFP 100	eTQFP 144
PA[12]	12	0000	GPIO12	SIUL2-GPIO12	GPIO12	I/O	x	x	x	x	IO #0	IO #0
		0001-0011	—	Reserved	—	—						
		0100	ETC3	ETIMER_0	Timer 0 - ch. 3 (IC/OC)	I/O						
		0101	B3	PWM_0	FlexPWM 0 - B[3]	I/O						
		0110	TXD	LINFlexD_0	LINFLEX 0 - TX	O						
		0111-1000	—	Reserved	—	—						
		1001	B0	PWM_1	FlexPWM 1 - B[0]	I/O						
		1010-1111	—	Reserved	—	—						
	520	00000010	M_CAN_1_RX	CANSUBSYS_1	MCAN 1 Receive Data	I						
	560	00000001	RXD	LINFlexD_0	LinFlexD 0 Receive Data	I						
	561	00000011	RXD	LINFlexD_1	LinFlexD 1 Receive Data	I						
	691	00000101	ETC3	ETIMER_0	ETMR0 input channel 3	I						
	959	00000101	B3	PWM_0	FlexPWM 0 - B[3] input channel	I						
	962	00000101	B0	PWM_1	FlexPWM 1 - B[0] input channel	I						

Table 12. Pin multiplexing table description (continued)

Port Pin	SIUL2 MSCR		Signal	Module	Full Signal Name	Direction	PAD strength				IO Segment	
	No	SSS Value					VeryStrong	Fast	Medium	Slow	eTQFP 100	eTQFP 144
PA[13]	13	0000	GPIO13	SIUL2-GPIO13	GPIO13	I/O	x	x	x	x	IO #0	IO #0
		0001	TXD	LINFlexD_0	LINFLEX 0 - TX	O						
		0010	M_CAN_1_TX	CANSUBSYS_1	MCAN 1 - TX	O						
		0011	TXD	LINFlexD_1	LINFLEX 1 - TX	O						
		0100	ETC4	ETIMER_0	Timer 0 - ch. 4 (IC/OC)	I/O						
		0101	—	Reserved	—	—						
		0110	B3	PWM_0	FlexPWM 0 - B[3]	I/O						
		0111-1000	—	Reserved	—	—						
		1001	B1	PWM_1	FlexPWM 1 - B[1]	I/O						
		1010-1111	—	Reserved	—	—						
	692	00000101	ETC4	ETIMER_0	ETMR0 input channel 4	I						
	959	00000110	B3	PWM_0	FlexPWM 0 - B[3] input channel	I						
	963	00000101	B1	PWM_1	FlexPWM 1 - B[1] input channel	I						

Table 12. Pin multiplexing table description (continued)

Port Pin	SIUL2 MSCR		Signal	Module	Full Signal Name	Direction	PAD strength				IO Segment	
	No	SSS Value					VeryStrong	Fast	Medium	Slow	eTQFP 100	eTQFP 144
PA[14]	14	0000	GPIO14	SIUL2-GPIO14	GPIO14	I/O	x	x	x	x	IO #0	IO #0
		0001	ETC4	ETIMER_0	Timer 0 - ch. 4 (IC/OC)	I/O						
		0010	CS1	DSPI_1	DSPI 1 - Chip Select 1	O						
		0011	—	Reserved	—	—						
		0100	ETC5	ETIMER_0	Timer 0 - ch. 5 (IC/OC)	I/O						
		0101	—	Reserved	—	—						
		0110	MSEO_B_1	NPC_WRAPPER	Nexus MSEO 1	O						
		0111	CS0	DSPI_0	DSPI 0 - Chip Select 0	I/O						
		1000-1111	—	Reserved	—	—						
	560	00000110	RXD	LINFlexD_0	LinFlexD 0 Receive Data	I						
	626	00001010	CS0	DSPI_0	DSPI0 Slave Select Input	I						
	632	00001001	CS0	DSPI_2	DSPI2 Slave Select Input	I						
	692	00000010	ETC4	ETIMER_0	ETMR0 input channel 4	I						
	693	00000101	ETC5	ETIMER_0	ETMR0 input channel 5	I						
	819	00000010	REQ3	SIUL	interrupt request channel 2	I						
PA[15]	15	0000	GPIO15	SIUL2-GPIO15	GPIO15	I/O	x	x	x	x	IO #0	IO #0
		0001	—	Reserved	—	—						
		0010	ETC0	ETIMER_1	Timer 1 - ch. 0 (IC/OC)	I/O						
		0011	MDO_2	NPC_WRAPPER	Nexus MDO 2	O						
		0100	ETC4	ETIMER_1	Timer 1 - ch. 4 (IC/OC)	I/O						
		0101	TXD_A	FLEXRAY_0	FlexRay 0 - TXA	O						
		0110	—	Reserved	—	—						
		0111	CS1	DSPI_0	DSPI 0 - Chip Select 1	O						
		1000-1111	—	Reserved	—	—						
	520	00000001	M_CAN_1_RX	CANSUBSYS_1	MCAN 1 Receive Data	I						
	694	00000011	ETC0	ETIMER_1	ETMR1 input channel 0	I						
	698	00000101	ETC4	ETIMER_1	ETMR1 input channel 4	I						

Table 12. Pin multiplexing table description (continued)

Port Pin	SIUL2 MSCR		Signal	Module	Full Signal Name	Direction	PAD strength				IO Segment	
	No	SSS Value					VeryStrong	Fast	Medium	Slow	eTQFP 100	eTQFP 144
PB[0]	16	0000	GPIO16	SIUL2-GPIO16	GPIO16	I/O	—	—	x	x	IO #0	IO #0
		0001	M_CAN_1_TX	CANSUBSYS_1	MCAN 1 - TX	O						
		0010	ETC1	ETIMER_1	Timer 1 - ch. 1 (IC/OC)	I/O						
		0011	MDO_3	NPC_WRAPPER	Nexus MDO 3	O						
		0100	ETC5	ETIMER_1	Timer 1 - ch. 5 (IC/OC)	I/O						
		0101-0110	—	Reserved	—	—						
		0111	CS2	DSPI_0	DSPI 0 - Chip Select 2	O						
		1000-1111	—	Reserved	—	—						
	695	00000011	ETC1	ETIMER_1	ETMR1 input channel 1	I						
	699	00000101	ETC5	ETIMER_1	ETMR1 input channel 5	I						
	784	00000101	RXD_A	FLEXRAY_0	Flexray Receive Data Channel A	I						
PB[1]	17	0000	GPIO17	SIUL2-GPIO17	GPIO17	I/O	x	x	x	x	IO #0	IO #0
		0001	ETC5	ETIMER_1	Timer 1 - ch. 5 (IC/OC)	I/O						
		0010	CS6	DSPI_0	DSPI 0 - Chip Select 6	O						
		0011	MSEO_B_0	NPC_WRAPPER	Nexus MSEO0	O						
		0100	CS0	DSPI_1	DSPI 1 - Chip Select 0	I/O						
		0101	M_CAN_0_TX	CANSUBSYS_0	MCAN 0 - TX	O						
		0110-1111	—	Reserved	—	—						
	629	00000110	CS0	DSPI_1	DSPI1 Slave Select Input	I						
	699	00000001	ETC5	ETIMER_1	ETMR1 input channel 5	I						

Table 12. Pin multiplexing table description (continued)

Port Pin	SIUL2 MSCR		Signal	Module	Full Signal Name	Direction	PAD strength				IO Segment	
	No	SSS Value					VeryStrong	Fast	Medium	Slow	eTQFP 100	eTQFP 144
PB[2]	18	0000	GPIO18	SIUL2-GPIO18	GPIO18	I	—	—	—	—	ADC #0	—
		0001-1111	—	Reserved	—	—						
	512	00000010	M_CAN_0_RX	CANSUBSYS_0	MCAN 0 Receive Data	I						
	692	00000001	ETC4	ETIMER_0	ETMR0 input channel 4	I						
	819	00000011	REQ3	SIUL	interrupt request channel 2	I						
	—	—	ANP[0](1)	SAR_ADC_SV0	analog channel	I						
	—	—	ANP[0](1)	SAR_ADC_0	analog channel	—						
PB[3]	19	0000	GPIO19	SIUL2-GPIO19	GPIO19	I	—	—	—	—	ADC #0	—
		0001-1111	—	Reserved	—	—						
	624	00000111	SIN	DSPI_0	DSPI0 Serial Data Input	I						
	688	00000001	ETC0	ETIMER_0	ETMR0 input channel 0	I						
	694	00000100	ETC0	ETIMER_1	ETMR1 input channel 0	I						
	—	—	ANP[4](1)	SAR_ADC_SV0	analog channel	I						
	—	—	ANP[4](1)	SAR_ADC_0	analog channel	—						
PB[4]	20	0000	GPIO20	SIUL2-GPIO20	GPIO20	I	—	—	—	—	ADC #0	—
		0001-1111	—	Reserved	—	—						
	627	00000111	SIN	DSPI_1	DSPI1 Serial Data Input	I						
	689	00000001	ETC1	ETIMER_0	ETMR0 input channel 1	I						
	695	00000100	ETC1	ETIMER_1	ETMR1 input channel 1	I						
	—	—	ANP[9](1)	SAR_ADC_SV0	analog channel	I						
	—	—	ANP[3](1)	SAR_ADC_1	analog channel							

Table 12. Pin multiplexing table description (continued)

Port Pin	SIUL2 MSCR		Signal	Module	Full Signal Name	Direction	PAD strength				IO Segment	
	No	SSS Value					VeryStrong	Fast	Medium	Slow	eTQFP 100	eTQFP 144
PB[5]	21	0000	GPIO21	SIUL2-GPIO21	GPIO21	I	—	—	—	—	ADC #0	—
		0001-1111	—	Reserved	—	—						
	630	00000111	SIN	DSPI_2	DSPI2 Serial Data Input	I						
	690	00000001	ETC2	ETIMER_0	ETMR0 input channel 2	I						
	696	00000100	ETC2	ETIMER_1	ETMR1 input channel 2	I						
	—	—	ANP[13](1)	SAR_ADC_SV0	analog channel	I						
	—	—	ANP[2](1)	SAR_ADC_2	analog channel							
PB[6]	22	0000	GPIO22	SIUL2-GPIO22	GPIO22	I	—	—	—	—	ADC #0	ADC #0
		0001-1111	—	Reserved	—	—						
	633	00000111	SIN	DSPI_3	DSPI3 Serial Data Input	I						
	691	00000001	ETC3	ETIMER_0	ETMR0 input channel 3	I						
	697	00000100	ETC3	ETIMER_1	ETMR1 input channel 3	I						
	984	00000110	EXT_SYNC	PWM_0	FlexPWM 0 - EXT_SYNC input channel	I						
	—	—	ANP[1]	SAR_ADC_SV0	analog channel	I						
	—	—	ANP[1]	SAR_ADC_0	analog channel							
PB[7]	23	0000	GPIO23	SIUL2-GPIO23	GPIO23	I	—	—	—	—	ADC #0	ADC #0
		0001-1111	—	Reserved	—	—						
	692	00000100	ETC4	ETIMER_0	ETMR0 input channel 4	I						
	698	00000110	ETC4	ETIMER_1	ETMR1 input channel 4	I						
	816	00000001	REQ0	SIUL	interrupt request channel 0	I						
	—	—	ANP[2]	SAR_ADC_SV0	analog channel	I						
	—	—	ANP[2]	SAR_ADC_0	analog channel							

Table 12. Pin multiplexing table description (continued)

Port Pin	SIUL2 MSCR		Signal	Module	Full Signal Name	Direction	PAD strength				IO Segment	
	No	SSS Value					VeryStrong	Fast	Medium	Slow	eTQFP 100	eTQFP 144
PB[8]	24	0000	GPIO24	SIUL2-GPIO24	GPIO24	I	—	—	—	—	ADC #0	ADC #0
		0001-1111	—	Reserved	—	—						
	520	00000101	M_CAN_1_RX	CANSUBSYS_1	MCAN 1 Receive Data	I						
	693	00000001	ETC5	ETIMER_0	ETMR0 input channel 5	I						
	820	00000010	REQ4	SIUL	interrupt request channel 2	I						
	—	—	ANP[3]	SAR_ADC_SV0	analog channel	I						
	—	—	ANP[3]	SAR_ADC_0	analog channel							
PB[9]	25	0000	GPIO25	SIUL2-GPIO25	GPIO25	I	—	—	—	—	ADC #0	ADC #0
	—	0001-1111	—	Reserved	—	—						
	560	00000010	RXD	LINFlexD_0	LinFlexD 0 Receive Data	I						
	703	00000001	ETC3	ETIMER_2	ETMR2 input channel 3	I						
	786	00000110	OS_TICK	FLEXRAY_0	Flexray Stop Watch Tick	I						
	821	00000010	REQ5	SIUL	interrupt request channel 2	I						
	—	—	ANP[5]	SAR_ADC_SV0	analog channel	I						
	—	—	ANP[5]	SAR_ADC_0	analog channel							
PB[10]	26	0000	GPIO26	SIUL2-GPIO26	GPIO26	I	—	—	—	—	ADC #0	ADC #0
	—	0001-1111	—	Reserved	—	—						
	693	00000100	ETC5	ETIMER_0	ETMR0 input channel 5	I						
	699	00000110	ETC5	ETIMER_1	ETMR1 input channel 5	I						
	767	00000011	EXT_IN15	CTU_0	CTU0 input trigger 15	I						
	817	00000001	REQ1	SIUL	interrupt request channel 1	I						
	—	—	ANP[6]	SAR_ADC_SV0	analog channel	I						
	—	—	ANP[0]	SAR_ADC_1	analog channel	—						

Table 12. Pin multiplexing table description (continued)

Port Pin	SIUL2 MSCR		Signal	Module	Full Signal Name	Direction	PAD strength				IO Segment	
	No	SSS Value					VeryStrong	Fast	Medium	Slow	eTQFP 100	eTQFP 144
PB[11]	27	0000	GPIO27	SIUL2-GPIO27	GPIO27	I	—	—	—	—	ADC #0	ADC #0
		0001-1111	—	Reserved	—	—						
	692	00000110	ETC4	ETIMER_0	ETMR0 input channel 4	I						
	698	00000100	ETC4	ETIMER_1	ETMR1 input channel 4	I						
	818	00000001	REQ2	SIUL	interrupt request channel 2	I						
	980	00000110	FAULT0	PWM_0	FlexPWM 0 - FAULT[0] input channel	I						
	—	—	ANP[7]	SAR_ADC_SV0	analog channel	I						
	—	—	ANP[1]	SAR_ADC_1	analog channel							
PB[12]	28	0000	GPIO28	SIUL2-GPIO28	GPIO28	I	—	—	—	—	ADC #0	ADC #0
		0001-1111	—	Reserved	—	—						
	561	00000100	RXD	LINFlexD_1	LinFlexD 1 Receive Data	I						
	699	00000100	ETC5	ETIMER_1	ETMR1 input channel 5	I						
	704	00000001	ETC4	ETIMER_2	ETMR2 input channel 4	I						
	981	00000110	FAULT1	PWM_0	FlexPWM 0 - FAULT[1] input channel	I						
	—	—	ANP[8]	SAR_ADC_SV0	analog channel	I						
	—	—	ANP[2]	SAR_ADC_1	analog channel							
PB[13]	29	0000	GPIO29	SIUL2-GPIO29	GPIO29	I	—	—	—	—	ADC #0	ADC #0
	—	0001-1111	—	Reserved	—	—						
	705	00000001	ETC5	ETIMER_2	ETMR2 input channel 5	I						
	706	00000100	ETC0	ETIMER_3	ETMR3 input channel 0	I						
	944	00000011	NMI	WKPUFILTER_ANA	NMI input	I						
	982	00000110	FAULT2	PWM_0	FlexPWM 0 - FAULT[2] input channel	I						
	—	—	ANP[10]	SAR_ADC_SV0	analog channel	I						
	—	—	ANP[4]	SAR_ADC_1	analog channel	—						

Table 12. Pin multiplexing table description (continued)

Port Pin	SIUL2 MSCR		Signal	Module	Full Signal Name	Direction	PAD strength				IO Segment	
	No	SSS Value					VeryStrong	Fast	Medium	Slow	eTQFP 100	eTQFP 144
PB[14]	30	0000	GPIO30	SIUL2-GPIO30	GPIO30	I	—	—	—	—	ADC #0	ADC #0
		0001-1111	—	Reserved	—	—						
	700	00000001	ETC0	ETIMER_2	ETMR2 input channel 0	I						
	701	00000101	ETC1	ETIMER_2	ETMR2 input channel 1	I						
	707	00000100	ETC1	ETIMER_3	ETMR3 input channel 1	I						
	983	00000110	FAULT3	PWM_0	FlexPWM 0 - FAULT[3] input channel	I						
	—	—	ANP[11]	SAR_ADC_SV0	analog channel	I						
	—	—	ANP[0]	SAR_ADC_2	analog channel	—						
PB[15]	31	0000	GPIO31	SIUL2-GPIO31	GPIO31	I	—	—	—	—	ADC #0	ADC #0
		0001-1111	—	Reserved	—	—						
	701	00000001	ETC1	ETIMER_2	ETMR2 input channel 1	I						
	702	00000101	ETC2	ETIMER_2	ETMR2 input channel 2	I						
	708	00000100	ETC2	ETIMER_3	ETMR3 input channel 2	I						
	—	—	ANP[12]	SAR_ADC_SV0	analog channel	I						
	—	—	ANP[1]	SAR_ADC_2	analog channel	—						
PC[0]	32	0000	GPIO32	SIUL2-GPIO32	GPIO32	I	—	—	—	—	ADC #0	ADC #0
	—	0001-1111	—	Reserved	—	—						
	694	00000001	ETC0	ETIMER_1	ETMR1 input channel 0	I						
	709	00000100	ETC3	ETIMER_3	ETMR3 input channel 3	I						
	816	00000100	REQ0	SIUL	interrupt request channel 0	I						
	—	—	ANP[14]	SAR_ADC_SV0	analog channel	I						
	—	—	ANP[3]	SAR_ADC_2	analog channel	—						

Table 12. Pin multiplexing table description (continued)

Port Pin	SIUL2 MSCR		Signal	Module	Full Signal Name	Direction	PAD strength				IO Segment	
	No	SSS Value					VeryStrong	Fast	Medium	Slow	eTQFP 100	eTQFP 144
PC[1]	33	0000	GPIO33	SIUL2-GPIO33	GPIO33	I	—	—	—	—	ADC #0	ADC #0
		0001-1111	—	Reserved	—	—						
	702	00000001	ETC2	ETIMER_2	ETMR2 input channel 2	I						
	704	00000101	ETC4	ETIMER_2	ETMR2 input channel 4	I						
	710	00000100	ETC4	ETIMER_3	ETMR3 input channel 4	I						
	—	—	ANP[15]	SAR_ADC_SV0	analog channel	I						
	—	—	ANP[4]	SAR_ADC_2	analog channel							
PC[2]	34	0000	GPIO34	SIUL2-GPIO34	GPIO34	I/O	—	—	x	x		—
		0001	ETC5	ETIMER_0	Timer 0 - ch. 5 (IC/OC)	I/O						
		0010	CS1	DSPI_2	DSPI 2 - Chip Select 1	O						
		0011-0110	—	Reserved	—	—						
		0111	ETC1	ETIMER_0	Timer 0 - ch1 (IC/OC)	I/O						
		1000-1111	—	Reserved	—	—						
	512	00000011	M_CAN_0_RX	CANSUBSYS_0	MCAN 0 Receive Data	I						
	520	00000011	M_CAN_1_RX	CANSUBSYS_1	MCAN 1 Receive Data	I						
	689	00001001	ETC1	ETIMER_0	ETMR0 input channel 1	I						
	693	00000010	ETC5	ETIMER_0	ETMR0 input channel 5	I						

Table 12. Pin multiplexing table description (continued)

Port Pin	SIUL2 MSCR		Signal	Module	Full Signal Name	Direction	PAD strength				IO Segment	
	No	SSS Value					VeryStrong	Fast	Medium	Slow	eTQFP 100	eTQFP 144
PC[3]	35	0000	GPIO35	SIUL2-GPIO35	GPIO35	I/O	—	—	x	x		—
		0001	ETC0	ETIMER_1	Timer 1 - ch. 0 (IC/OC)	I/O						
		0010	CS2	DSPI_2	DSPI 2 - Chip Select 2	O						
		0011	M_CAN_1_TX	CANSUBSYS_1	MCAN 1 - TX	O						
		0100	M_CAN_0_TX	CANSUBSYS_0	MCAN 0 - TX	O						
		0101-0110	—	Reserved	—	—						
		0111	ETC2	ETIMER_0	Timer 0 - ch2 (IC/OC)	I/O						
		1000-1111	—	Reserved	—	—						
	690	00001001	ETC2	ETIMER_0	ETMR0 input channel 2	I						
	694	00000010	ETC0	ETIMER_1	ETMR1 input channel 0	I						
PC[4]	36	0000	GPIO36	SIUL2-GPIO36	GPIO36	I/O	—	—	x	x	IO #0	—
		0001	ETC1	ETIMER_1	Timer 1 - ch. 1 (IC/OC)	I/O						
		0010	CS0	DSPI_1	DSPI 1 - Chip Select 0	I/O						
		0011-0110	—	Reserved	—	—						
		0111	ETC3	ETIMER_0	Timer 0 - ch3 (IC/OC)	I/O						
		1000	ETC5	ETIMER_0	Timer #0 ch5 (IC/OC)	I/O						
		1001-1111	—	Reserved	—	—						
	520	00000110	M_CAN_1_RX	CANSUBSYS_1	MCAN 1 Receive Data	I						
	629	00000100	CS0	DSPI_1	DSPI1 Slave Select Input	I						
	691	00001001	ETC3	ETIMER_0	ETMR0 input channel 3	I						
	693	00001000	ETC5	ETIMER_0	ETMR0 input channel 5	I						
	695	00000001	ETC1	ETIMER_1	ETMR1 input channel 1	I						
	817	00000011	REQ1	SIUL	interrupt request channel 1	I						

Table 12. Pin multiplexing table description (continued)

Port Pin	SIUL2 MSCR		Signal	Module	Full Signal Name	Direction	PAD strength				IO Segment	
	No	SSS Value					VeryStrong	Fast	Medium	Slow	eTQFP 100	eTQFP 144
PC[5]	37	0000	GPIO37	SIUL2-GPIO37	GPIO37	I/O	x	x	x	x	IO #0	—
		0001	CS0	DSPI_1	DSPI 1 - Chip Select 0	I/O						
		0010	ETC2	ETIMER_1	Timer 1 - ch. 2 (IC/OC)	I/O						
		0011	RDY_B	INTG_DEBUG	Nexus RDY	O						
		0100	M_CAN_1_TX	CANSUBSYS_1	MCAN 1 - TX	O						
		0101-1111	—	Reserved	—	—						
	629	00000010	CS0	DSPI_1	DSPI1 Slave Select Input	I						
	696	00000011	ETC2	ETIMER_1	ETMR1 input channel 2	I						
PC[6]	38	0000	GPIO38	SIUL2-GPIO38	GPIO38	I/O	—	—	x	x	IO #0	—
		0001	SOUT	DSPI_1	DSPI 1 - Serial Data	I/O						
		0010	ETC3	ETIMER_1	Timer 1 - ch. 3 (IC/OC)	I/O						
		0011	CS4	DSPI_2	DSPI 2 - Chip Select 4	O						
		0100	SOUT	DSPI_0	DSPI 0 - Serial Data	I/O						
		0101-0110	—	Reserved	—	—						
		0111	ETC0	ETIMER_1	Timer 1 - ch0 (IC/OC)	I/O						
		1000-1111	—	Reserved	—	—						
	624	00000101	SIN	DSPI_0	DSPI0 Serial Data Input	I						
	627	00000001	SIN	DSPI_1	DSPI1 Serial Data Input	I						
	694	00001010	ETC0	ETIMER_1	ETMR1 input channel 0	I						
	697	00000011	ETC3	ETIMER_1	ETMR1 input channel 3	I						

Table 12. Pin multiplexing table description (continued)

Port Pin	SIUL2 MSCR		Signal	Module	Full Signal Name	Direction	PAD strength				IO Segment	
	No	SSS Value					VeryStrong	Fast	Medium	Slow	eTQFP 100	eTQFP 144
PC[7]	39	0000	GPIO39	SIUL2-GPIO39	GPIO39	I/O	x	x	x	x	IO #0	—
		0001	ETC2	ETIMER_1	Timer 1 - ch. 2 (IC/OC)	I/O						
		0010	SOUT	DSPI_1	DSPI 1 - Serial Data	I/O						
		0011	CS5	DSPI_2	DSPI 2 - Chip Select 5	O						
		0100	CS0	DSPI_0	DSPI 0 - Chip Select 0	I/O						
		0101-0110	—	Reserved	—	—						
		0111	ETC2	ETIMER_0	Timer 0 - ch2 (IC/OC)	I/O						
		1000-1111	—	Reserved	—	—						
	626	00000110	CS0	DSPI_0	DSPI0 Slave Select Input	I						
	627	00000100	SIN	DSPI_1	DSPI1 Serial Data Input	I						
	690	00001010	ETC2	ETIMER_0	ETMR0 input channel 2	I						
	696	00000001	ETC2	ETIMER_1	ETMR1 input channel 2	I						
PC[8]	40	0000	GPIO40	SIUL2-GPIO40	GPIO40	I/O	x	x	x	x	IO #0	—
		0001	ETC3	ETIMER_1	Timer 1 - ch. 3 (IC/OC)	I/O						
		0010	SOUT	DSPI_1	DSPI 1 - Serial Data	I/O						
		0011	CS6	DSPI_2	DSPI 2 - Chip Select 6	O						
		0100	CS1	DSPI_0	DSPI 0 - Chip Select 1	O						
		0101-1111	—	Reserved	—	—						
	627	00000011	SIN	DSPI_1	DSPI1 Serial Data Input	I						
	697	00000001	ETC3	ETIMER_1	ETMR1 input channel 3	I						

Table 12. Pin multiplexing table description (continued)

Port Pin	SIUL2 MSCR		Signal	Module	Full Signal Name	Direction	PAD strength				IO Segment	
	No	SSS Value					VeryStrong	Fast	Medium	Slow	eTQFP 100	eTQFP 144
PC[9]	41	0000	GPIO41	SIUL2-GPIO41	GPIO41	I/O	x	x	x	x	IO #0	—
		0001	SOUT	DSPI_1	DSPI 1 - Serial Data	I/O						
		0010	ETC4	ETIMER_1	Timer 1 - ch. 4 (IC/OC)	I/O						
		0011	CS7	DSPI_2	DSPI 2 - Chip Select 7	O						
		0100	SOUT	DSPI_0	DSPI 0 - Serial Data	I/O						
		0101-0110	—	Reserved	—	—						
		0111	CS7	DSPI_0	DSPI 0 - Chip Select 7	O						
		1000	SCK	DSPI_3	DSPI 3 - Serial Clock	I/O						
		1001-1111	—	Reserved	—	—						
	624	00000110	SIN	DSPI_0	DSPI0 Serial Data Input	I						
	627	00000010	SIN	DSPI_1	DSPI1 Serial Data Input	I						
	634	00001000	SCK	DSPI_3	DSPI3 Serial Clock Input	I						
	698	00000011	ETC4	ETIMER_1	ETMR1 input channel 4	I						
PC[10]	42	0000	GPIO42	SIUL2-GPIO42	GPIO42	I/O	x	x	x	x	IO #0	—
		0001	SCK	DSPI_1	DSPI 1 - Serial Clock	I/O						
		0010	ETC5	ETIMER_1	Timer 1 - ch. 5 (IC/OC)	I/O						
		0011	CS4	DSPI_3	DSPI 3 - Chip Select 4	O						
		0100	SCK	DSPI_0	DSPI 0 - Serial Clock	I/O						
		0101-1111	—	Reserved	—	—						
	625	00000110	SCK	DSPI_0	DSPI0 Serial Clock Input	I						
	628	00000010	SCK	DSPI_1	DSPI1 Serial Clock Input	I						
	699	00000011	ETC5	ETIMER_1	ETMR1 input channel 5	I						

Table 12. Pin multiplexing table description (continued)

Port Pin	SIUL2 MSCR		Signal	Module	Full Signal Name	Direction	PAD strength				IO Segment	
	No	SSS Value					VeryStrong	Fast	Medium	Slow	eTQFP 100	eTQFP 144
PC[11]	43	0000	GPIO43	SIUL2-GPIO43	GPIO43	I/O	x	x	x	x	IO #0	—
		0001	ETC4	ETIMER_1	Timer 1 - ch. 4 (IC/OC)	I/O						
		0010	SCK	DSPI_1	DSPI 1 - Serial Clock	I/O						
		0011	CS5	DSPI_3	DSPI 3 - Chip Select 5	O						
		0100	CS2	DSPI_0	DSPI 0 - Chip Select 2	O						
		0101-0110	—	Reserved	—	—						
		0111	CS3	DSPI_2	DSPI 2 - Chip Select 3	O						
		1000	ETC0	ETIMER_0	Timer 0 - ch. 0 (IC/OC)	I/O						
		1001-1111	—	Reserved	—	—						
	628	00000100	SCK	DSPI_1	DSPI1 Serial Clock Input	I						
	688	00001010	ETC0	ETIMER_0	ETMR0 input channel 0	I						
	698	00000001	ETC4	ETIMER_1	ETMR1 input channel 4	I						
PC[12]	44	0000	GPIO44	SIUL2-GPIO44	GPIO44	I/O	x	x	x	x	IO #0	IO #1
		0001	ETC0	ETIMER_0	Timer 0 - ch. 0 (IC/OC)	I/O						
		0010	CS3	DSPI_1	DSPI 1 - Chip Select 3	O						
		0011	CS3	DSPI_3	DSPI 3 - Chip Select 3	O						
		0100	—	Reserved		—						
		0101	A1	PWM_1	FlexPWM 1 - A[1] (SWG PWM 0)	I/O						
		0110-1111	—	Reserved	—							
	560	00000011	RXD	LINFlexD_0	LinFlexD 0 Receive Data	I						
	688	00000010	ETC0	ETIMER_0	ETMR0 input channel 0	I						
	961	00000101	A1	PWM_1	FlexPWM 1 - A[1] input channel	I						

Table 12. Pin multiplexing table description (continued)

Port Pin	SIUL2 MSCR		Signal	Module	Full Signal Name	Direction	PAD strength				IO Segment	
	No	SSS Value					VeryStrong	Fast	Medium	Slow	eTQFP 100	eTQFP 144
PC[13]	45	0000	GPIO45	SIUL2-GPIO45	GPIO45	I/O	x	x	x	x	IO #0	IO #1
		0001	ETC1	ETIMER_0	Timer 0 - ch. 1 (IC/OC)	I/O						
		0010	CS4	DSPI_1	DSPI 1 - Chip Select 4	O						
		0011	CS4	DSPI_3	DSPI 3 - Chip Select 4	O						
		0100	TXD	LINFlexD_0	LINFLEX 0 - TX	O						
		0101-1000	—	Reserved	—	—						
		1001	X0	PWM_2	FlexPWM 2 - X[0]	I/O						
		1010-1111	—	Reserved	—	—						
	561	00000110	RXD	LINFlexD_1	LinFlexD 1 Receive Data	I						
	689	00000010	ETC1	ETIMER_0	ETMR0 input channel 1	I						
	988	00000101	X0	PWM_2	FlexPWM 2 - X[0] input channel	I						
PC[14]	46	0000	GPIO46	SIUL2-GPIO46	GPIO46	I/O	x	x	x	x	IO #0	IO #1
		0001	ETC2	ETIMER_0	Timer 0 - ch. 2 (IC/OC)	I/O						
		0010	CS5	DSPI_1	DSPI 1 - Chip Select 5	O						
		0011	CS5	DSPI_3	DSPI 3 - Chip Select 5	O						
		0100	CS3	DSPI_0	DSPI 0 - Chip Select 3	O						
		0101	TXD	LINFlexD_1	LINFLEX 1 - TX	O						
		0110-1000	—	Reserved	—	—						
		1001	X1	PWM_2	FlexPWM 2 - X[1]	I/O						
		1010-1111	—	Reserved	—	—						
	690	00000010	ETC2	ETIMER_0	ETMR0 input channel 2	I						
	989	00000101	X1	PWM_2	FlexPWM 2 - X[1] input channel	I						

Table 12. Pin multiplexing table description (continued)

Port Pin	SIUL2 MSCR		Signal	Module	Full Signal Name	Direction	PAD strength				IO Segment	
	No	SSS Value					VeryStrong	Fast	Medium	Slow	eTQFP 100	eTQFP 144
PC[15]	47	0000	GPIO47	SIUL2-GPIO47	GPIO47	I/O	—	—	x	x	IO #0	IO #1
		0001	—	Reserved		—						
		0010	CS2	DSPI_1	DSPI 1 - Chip Select 2	O						
		0011	—	Reserved		—						
		0100	ETC0	ETIMER_2	Timer 2 - ch. 0 (IC/OC)	I/O						
		0101	ESR0	MC_RGM	—	O						
		0110-1111	—	Reserved	—	—						
	700	00000110	ETC0	ETIMER_2	ETMR2 input channel 0	I						
	820	00000011	REQ4	SIUL	interrupt request channel 2	I						
	944	00000001	NMI	WKPUFILTER_ANA	NMI input	I						
PD[0]	48	0000	GPIO48	SIUL2-GPIO48	GPIO48	I/O	x	x	x	x	IO #0	IO #1
		0001	CS6	DSPI_1	DSPI 1 - Chip Select 6	O						
		0010	—	Reserved		—						
		0011	CS1	DSPI_3	DSPI 3 - Chip Select 1	O						
		0100	ETC1	ETIMER_2	Timer 2 - ch. 1 (IC/OC)	I/O						
		0101-1000	—	Reserved	—	—						
		1001	X2	PWM_2	FlexPWM 2 - X[2]	I/O						
		1010-1111	—	Reserved	—	—						
	701	00000110	ETC1	ETIMER_2	ETMR2 input channel 1	I						
	816	00000011	REQ0	SIUL	interrupt request channel 0	I						
	990	00000101	X2	PWM_2	FlexPWM 2 - X[2] input channel	I						

Table 12. Pin multiplexing table description (continued)

Port Pin	SIUL2 MSCR		Signal	Module	Full Signal Name	Direction	PAD strength				IO Segment	
	No	SSS Value					VeryStrong	Fast	Medium	Slow	eTQFP 100	eTQFP 144
PD[1]	49	0000	GPIO49	SIUL2-GPIO49	GPIO49	I/O	x	x	x	x	IO #0	IO #1
		0001	ETC3	ETIMER_0	Timer 0 - ch. 3 (IC/OC)	I/O						
		0010	CS7	DSPI_1	DSPI 1 - Chip Select 7	O						
		0011	CS2	DSPI_3	DSPI 3 - Chip Select 2	O						
		0100	CS4	DSPI_0	DSPI 0 - Chip Select 4	O						
		0101-0110	—	Reserved	—	—						
		0111	ETC3	ETIMER_2	Timer 2 - ch. 3 (IC/OC)	I/O						
		1000	SOUT	DSPI_2	DSPI2 - Serial Data	I/O						
		1001	X3	PWM_2	FlexPWM 2 - X[3]	I/O						
		1010-1111	—	Reserved	—	—						
	630	00001000	SIN	DSPI_2	DSPI2 Serial Data Input	I						
	691	00000010	ETC3	ETIMER_0	ETMR0 input channel 3	I						
	703	00001010	ETC3	ETIMER_2	ETMR2 input channel 3	I						
	991	00000101	X3	PWM_2	FlexPWM 2 - X[3] input channel	I						
PD[2]	50	0000	GPIO50	SIUL2-GPIO50	GPIO50	I/O	—	—	x	x	IO #0	IO #1
		0001	ETC0	ETIMER_2	Timer 2 - ch. 0 (IC/OC)	I/O						
		0010	CS1	DSPI_2	DSPI 2 - Chip Select 1	O						
		0011	CS6	DSPI_1	DSPI 1 - Chip Select 6	O						
		0100	ETC0	ETIMER_3	Timer 3 - ch. 0 (IC/OC)	I/O						
		0101-1000	—	Reserved	—	—						
		1001	DEBUG_0	FLEXRAY_0	Flexray - Debug[0]	O						
		1010-1111	—	Reserved	—	—						
	700	00000010	ETC0	ETIMER_2	ETMR2 input channel 0	I						
	706	00000110	ETC0	ETIMER_3	ETMR3 input channel 0	I						

Table 12. Pin multiplexing table description (continued)

Port Pin	SIUL2 MSCR		Signal	Module	Full Signal Name	Direction	PAD strength				IO Segment	
	No	SSS Value					VeryStrong	Fast	Medium	Slow	eTQFP 100	eTQFP 144
PD[3]	51	0000	GPIO51	SIUL2-GPIO51	GPIO51	I/O	—	—	x	x	IO #0	IO #1
		0001	ETC1	ETIMER_2	Timer 2 - ch. 1 (IC/OC)	I/O						
		0010	CS2	DSPI_2	DSPI 2 - Chip Select 2	O						
		0011	CS4	DSPI_1	DSPI 1 - Chip Select 4	O						
		0100	ETC1	ETIMER_3	Timer 3 - ch. 1 (IC/OC)	I/O						
		0101-1000	—	Reserved	—	—						
		1001	DEBUG_1	FLEXRAY_0	Flexray - Debug[1]	O						
		1010-1111	—	Reserved	—	—						
	701	00000010	ETC1	ETIMER_2	ETMR2 input channel 1	I						
	707	00000110	ETC1	ETIMER_3	ETMR3 input channel 1	I						
PD[4]	52	0000	GPIO52	SIUL2-GPIO52	GPIO52	I/O	—	—	x	x	IO #0	IO #1
		0001	ETC2	ETIMER_2	Timer 2 - ch. 2 (IC/OC)	I/O						
		0010	CS3	DSPI_2	DSPI 2 - Chip Select 3	O						
		0011	CS7	DSPI_1	DSPI 1 - Chip Select 7	O						
		0100	ETC2	ETIMER_3	Timer 3 - ch. 2 (IC/OC)	I/O						
		0101-1000	—	Reserved	—	—						
		1001	DEBUG_2	FLEXRAY_0	Flexray - Debug[2]	O						
		1010-1111	—	Reserved	—	—						
	702	00000010	ETC2	ETIMER_2	ETMR2 input channel 2	I						
	708	00000110	ETC2	ETIMER_3	ETMR3 input channel 2	I						

Table 12. Pin multiplexing table description (continued)

Port Pin	SIUL2 MSCR		Signal	Module	Full Signal Name	Direction	PAD strength				IO Segment	
	No	SSS Value					VeryStrong	Fast	Medium	Slow	eTQFP 100	eTQFP 144
PD[5]	53	0000	GPIO53	SIUL2-GPIO53	GPIO53	I/O	x	x	x	x	IO #0	IO #1
		0001	CS0	DSPI_2	DSPI2 - Chip Select 0	I/O						
		0010	ETC1	ETIMER_2	Timer 2 - ch. 1 (IC/OC)	I/O						
		0011	CS6	DSPI_1	DSPI 1 - Chip Select 6	O						
		0100	ETC3	ETIMER_3	Timer 3 - ch. 3 (IC/OC)	I/O						
		0101	A1	PWM_2	FlexPWM 2 - A[1]	—						
		0110-1000	—	Reserved	—	—						
		1001	A0	PWM_3	FlexPWM 3 - A[0]	I/O						
		1010-1111	—	Reserved	—	—						
	632	00000010	CS0	DSPI_2	DSPI2 Slave Select Input	I						
	701	00000011	ETC1	ETIMER_2	ETMR2 input channel 1	I						
	709	00000110	ETC3	ETIMER_3	ETMR3 input channel 3	I						
	965	00000101	A1	PWM_2	FlexPWM 2 - A[1] input channel	I						
	972	00000101	A0	PWM_3	FlexPWM 3 - A[0] input channel	I						

Table 12. Pin multiplexing table description (continued)

Port Pin	SIUL2 MSCR		Signal	Module	Full Signal Name	Direction	PAD strength				IO Segment	
	No	SSS Value					VeryStrong	Fast	Medium	Slow	eTQFP 100	eTQFP 144
PD[6]	54	0000	GPIO54	SIUL2-GPIO54	GPIO54	I/O	x	x	x	x	IO #0	IO #1
		0001	SOUT	DSPI_2	DSPI2 - Serial Data	I/O						
		0010	ETC2	ETIMER_2	Timer 2 - ch. 2 (IC/OC)	I/O						
		0011	CS5	DSPI_1	DSPI 1 - Chip Select 5	O						
		0100	CS5	DSPI_0	DSPI 0 - Chip Select 5	O						
		0101	A2	PWM_2	FlexPWM 2 - A[2]	—						
		0110-1000	—	Reserved	—	—						
		1001	B0	PWM_3	FlexPWM 3 - B[0]	I/O						
		1010-1111	—	Reserved	—	—						
	630	00000001	SIN	DSPI_2	DSPI2 Serial Data Input	I						
	702	00000011	ETC2	ETIMER_2	ETMR2 input channel 2	I						
	966	00000101	A2	PWM_2	FlexPWM 2 - A[2] input channel	I						
	974	00000101	B0	PWM_3	FlexPWM 3 - B[0] input channel	I						
PD[7]	55	0000	GPIO55	SIUL2-GPIO55	GPIO55	I/O	—	—	x	x	IO #0	IO #1
		0001	ETC0	ETIMER_3	Timer 3 - ch. 0 (IC/OC)	I/O						
		0010	—	Reserved		—						
		0011	CS2	DSPI_1	DSPI 1 - Chip Select 2	O						
		0100	—	Reserved		—						
		0101	A3	PWM_2	FlexPWM 2 - A[3]	—						
		0110-1111	—	Reserved	—	—						
	561	00000101	RXD	LINFlexD_1	LinFlexD 1 Receive Data	I						
	706	00000001	ETC0	ETIMER_3	ETMR3 input channel 0	I						
	767	00000010	EXT_IN15	CTU_0	CTU0 input trigger 15	I						
	783	00000010	EXT_IN15	CTU_1	CTU1 input trigger 15	I						
	967	00000101	A3	PWM_2	FlexPWM 2 - A[3] input channel	I						

Table 12. Pin multiplexing table description (continued)

Port Pin	SIUL2 MSCR		Signal	Module	Full Signal Name	Direction	PAD strength				IO Segment	
	No	SSS Value					VeryStrong	Fast	Medium	Slow	eTQFP 100	eTQFP 144
PD[8]	56	0000	GPIO56	SIUL2-GPIO56	GPIO56	I/O	—	—	x	x	IO #0	IO #1
		0001	ETC1	ETIMER_3	Timer 3 - ch. 1 (IC/OC)	I/O						
		0010	EXT_TGR0	CTU_0	CTU 0 - Trigger Output	O						
		0011	CS6	DSPI_1	DSPI 1 - Chip Select 6	O						
		0100	TXD	LINFlexD_1	LINFLEX 1 - TX	O						
		0101	B1	PWM_2	FlexPWM 2 - B[1]	—						
		0110-1000	—	Reserved	—	—						
		1001	DEBUG_3	FLEXRAY_0	Flexray - Debug[3]	O						
		1010-1111	—	Reserved	—	—						
	707	00000001	ETC1	ETIMER_3	ETMR3 input channel 1	I						
	969	00000101	B1	PWM_2	FlexPWM 2 - B[1] input channel	I						
PD[9]	57	0000	GPIO57	SIUL2-GPIO57	GPIO57	I/O	x	x	x	x	IO #0	IO #1
		0001	—	Reserved		—						
		0010	CS1	DSPI_2	DSPI 2 - Chip Select 1	O						
		0011	—	Reserved		—						
		0100	ETC2	ETIMER_2	Timer 2 - ch. 2 (IC/OC)	I/O						
		0101-1000	—	Reserved	—	—						
		1001	A0	PWM_2	FlexPWM 2 - A[0]	I/O						
		1010-1111	—	Reserved	—	—						
	512	00000001	M_CAN_0_RX	CANSUBSYS_0	MCAN 0 Receive Data	I						
	520	00000100	M_CAN_1_RX	CANSUBSYS_1	MCAN 1 Receive Data	I						
	702	00000110	ETC2	ETIMER_2	ETMR2 input channel 2	I						
	964	00000101	A0	PWM_2	FlexPWM 2 - A[0] input channel	I						

Table 12. Pin multiplexing table description (continued)

Port Pin	SIUL2 MSCR		Signal	Module	Full Signal Name	Direction	PAD strength				IO Segment	
	No	SSS Value					VeryStrong	Fast	Medium	Slow	eTQFP 100	eTQFP 144
PD[10]	58	0000	GPIO58	SIUL2-GPIO58	GPIO58	I/O	x	x	x	x	IO #0	IO #1
		0001	M_CAN_0_TX	CANSUBSYS_0	MCAN 0 - TX	O						
		0010	CS1	DSPI_3	DSPI 3 - Chip Select 1	O						
		0011	M_CAN_1_TX	CANSUBSYS_1	MCAN 1 - TX	O						
		0100	ETC3	ETIMER_2	Timer 2 - ch. 3 (IC/OC)	I/O						
		0101-1000	—	Reserved	—	—						
		1001	B0	PWM_2	FlexPWM 2 - B[0]	I/O						
		1010-1111	—	Reserved	—	—						
	703	00000110	ETC3	ETIMER_2	ETMR2 input channel 3	I						
	968	00000101	B0	PWM_2	FlexPWM 2 - B[0] input channel	I						
PD[11]	59	0000	GPIO59	SIUL2-GPIO59	GPIO59	I/O	x	x	x	x	IO #0	IO #1
		0001	ETC2	ETIMER_3	Timer 3 - ch. 2 (IC/OC)	I/O						
		0010	SCK	DSPI_2	DSPI 2 - Serial Clock	I/O						
		0011	CS7	DSPI_1	DSPI 1 - Chip Select 7	O						
		0100	SCK	DSPI_3	DSPI 3 - Serial Clock	I/O						
		0101-0110	—	Reserved	—	—						
		0111	SOUT	DSPI_0	DSPI 0 - Serial Data	I/O						
		1000	—	Reserved		—						
		1001	B1	PWM_3	FlexPWM 3 - B[1]	I/O						
		1010-1111	—	Reserved	—	—						
	624	00001011	SIN	DSPI_0	DSPI0 Serial Data Input	I						
	631	00000100	SCK	DSPI_2	DSPI2 Serial Clock Input	I						
	634	00000110	SCK	DSPI_3	DSPI3 Serial Clock Input	I						
	708	00000001	ETC2	ETIMER_3	ETMR3 input channel 2	I						
	975	00000101	B1	PWM_3	FlexPWM 3 - B[1] input channel	I						

Table 12. Pin multiplexing table description (continued)

Port Pin	SIUL2 MSCR		Signal	Module	Full Signal Name	Direction	PAD strength				IO Segment	
	No	SSS Value					VeryStrong	Fast	Medium	Slow	eTQFP 100	eTQFP 144
PD[12]	60	0000	GPIO60	SIUL2-GPIO60	GPIO60	I/O	x	x	x	x	IO #0	IO #1
		0001	SOUT	DSPI_2	DSPI2 - Serial Data	I/O						
		0010	ETC3	ETIMER_2	Timer 2 - ch. 3 (IC/OC)	I/O						
		0011	CS2	DSPI_2	DSPI 2 - Chip Select 2	O						
		0100	CS0	DSPI_3	DSPI 3 - Chip Select 0	I/O						
		0101	SOUT	DSPI_3	DSPI 3 - Serial Data	I/O						
		0110-1111	—	Reserved	—	—						
	630	00000010	SIN	DSPI_2	DSPI2 Serial Data Input	I						
	633	00001001	SIN	DSPI_3	DSPI3 Serial Data Input	I						
	635	00000110	CS0	DSPI_3	DSPI3 Slave Select Input	I						
	703	00000011	ETC3	ETIMER_2	ETMR2 input channel 3	I						
	992	00000101	FAULT0	PWM_2	FlexPWM 2 - FAULT[0] input channel	I						
PD[13]	61	0000	GPIO61	SIUL2-GPIO61	GPIO61	I/O	x	x	x	x	IO #0	IO #1
		0001	SCK	DSPI_2	DSPI2 - Serial Clock	I/O						
		0010	ETC4	ETIMER_2	Timer 2 - ch. 4 (IC/OC)	I/O						
		0011	CS3	DSPI_2	DSPI 2 - Chip Select 3	O						
		0100	CS1	DSPI_3	DSPI 3 - Chip Select 1	O						
		0101-0110	—	Reserved	—	—						
		0111	ETC0	ETIMER_3	Timer 3 - ch. 0 (IC/OC)	I/O						
		1000-1111	—	Reserved	—	—						
	631	00000010	SCK	DSPI_2	DSPI2 Serial Clock Input	I						
	704	00000011	ETC4	ETIMER_2	ETMR2 input channel 4	I						
	706	00001000	ETC0	ETIMER_3	ETMR3 input channel 0	I						
	792	00000110	SENT0	SENT_1	SENT Receiver 1 Channel 0	I						
	993	00000101	FAULT1	PWM_2	FlexPWM 2 - FAULT[1] input channel	I						

Table 12. Pin multiplexing table description (continued)

Port Pin	SIUL2 MSCR		Signal	Module	Full Signal Name	Direction	PAD strength				IO Segment	
	No	SSS Value					VeryStrong	Fast	Medium	Slow	eTQFP 100	eTQFP 144
PD[14]	62	0000	GPIO62	SIUL2-GPIO62	GPIO62	I/O	—	—	x	x	IO #0	IO #1
		0001	ETC3	ETIMER_2	Timer 2 - ch. 3 (IC/OC)	I/O						
		0010	SOUT	DSPI_2	DSPI 2 - Serial Data	I/O						
		0011	ETC3	ETIMER_3	Timer 3 - ch. 3 (IC/OC)	I/O						
		0100	SOUT	DSPI_3	DSPI 3 - Serial Data	I/O						
		0101-0110	—	Reserved	—	—						
		0111	CS0	DSPI_3	DSPI 3 - Chip Select 0	I/O						
		1000-1111	—	Reserved	—	—						
	630	00000011	SIN	DSPI_2	DSPI2 Serial Data Input	I						
	633	00000101	SIN	DSPI_3	DSPI3 Serial Data Input	I						
	635	00000111	CS0	DSPI_3	DSPI3 Slave Select Input	I						
	703	00000010	ETC3	ETIMER_2	ETMR2 input channel 3	I						
	709	00000101	ETC3	ETIMER_3	ETMR3 input channel 3	I						
	793	00000110	SENT1	SENT_1	SENT Receiver 1 Channel 1	I						
	994	00000101	FAULT2	PWM_2	FlexPWM 2 - FAULT[2] input channel	I						
PD[15]	63	0000	GPIO63	SIUL2-GPIO63	GPIO63	I/O	x	x	x	x	IO #0	IO #1
		0001	ETC4	ETIMER_2	Timer 2 - ch. 4 (IC/OC)	I/O						
		0010	SOUT	DSPI_2	DSPI 2 - Serial Data	I/O						
		0011	ETC4	ETIMER_3	Timer 3 - ch. 4 (IC/OC)	I/O						
		0100	SOUT	DSPI_3	DSPI 3 - Serial Data	I/O						
		0101-1111	—	Reserved	—	—						
	630	00000100	SIN	DSPI_2	DSPI2 Serial Data Input	I						
	633	00000110	SIN	DSPI_3	DSPI3 Serial Data Input	I						
	704	00000010	ETC4	ETIMER_2	ETMR2 input channel 4	I						
	710	00000101	ETC4	ETIMER_3	ETMR3 input channel 4	I						
	794	00000110	SENT2	SENT_1	SENT Receiver 1 Channel 2	I						
	995	00000101	FAULT3	PWM_2	FlexPWM 2 - FAULT[3] input channel	I						

Table 12. Pin multiplexing table description (continued)

Port Pin	SIUL2 MSCR		Signal	Module	Full Signal Name	Direction	PAD strength				IO Segment	
	No	SSS Value					VeryStrong	Fast	Medium	Slow	eTQFP 100	eTQFP 144
PE[0]	64	0000	GPIO64	SIUL2-GPIO64	GPIO64	I/O	—	—	x	x	IO #0	IO #1
		0001	ETC3	ETIMER_3	Timer 3 - ch. 3 (IC/OC)	I/O						
		0010	—	Reserved		—						
		0011	CS6	DSPI_3	DSPI 3 - Chip Select 6	O						
		0100	ETC4	ETIMER_2	Timer 2 - ch. 4 (IC/OC)	I/O						
		0101	TXEN_B	FLEXRAY_0	FlexRay 0 - TXEN_B	O						
		0110	—	Reserved		—						
		0111	CS0	DSPI_3	DSPI 3 - Chip Select 0	I/O						
		1000-1111	—	Reserved	—	—						
	635	00001000	CS0	DSPI_3	DSPI3 Slave Select Input	I						
	704	00000110	ETC4	ETIMER_2	ETMR2 input channel 4	I						
	709	00000001	ETC3	ETIMER_3	ETMR3 input channel 3	I						
	818	00000010	REQ2	SIUL	interrupt request channel 2	I						
PE[1]	65	0000	GPIO65	SIUL2-GPIO65	GPIO65	I/O	—	—	x	x	IO #0	IO #1
		0001	ETC4	ETIMER_3	Timer 3 - Ch. 4 (IC/OC)	I/O						
		0010	CS2	DSPI_3	DSPI 3 - Chip Select 2	O						
		0011	CS7	DSPI_3	DSPI 3 - Chip Select 7	O						
		0100	ETC5	ETIMER_2	Timer 2 - ch. 5 (IC/OC)	I/O						
		0101	—	Reserved		—						
		0110	MSEO_B_1	NPC_WRAPPER	Nexus MSEO 1	O						
		0111	CS1	DSPI_3	DSPI 3 - Chip Select 1	O						
		1000-1111	—	Reserved	—	—						
	705	00000110	ETC5	ETIMER_2	ETMR2 input channel 5	I						
	710	00000001	ETC4	ETIMER_3	ETMR3 input channel 4	I						
	783	00000100	EXT_IN15	CTU_1	CTU1 input trigger 15	I						

Table 12. Pin multiplexing table description (continued)

Port Pin	SIUL2 MSCR		Signal	Module	Full Signal Name	Direction	PAD strength				IO Segment	
	No	SSS Value					VeryStrong	Fast	Medium	Slow	eTQFP 100	eTQFP 144
PE[2]	66	0000	GPIO66	SIUL2-GPIO66	GPIO66	I/O	x	x	x	x	IO #0	IO #0
		0001	ETC5	ETIMER_2	Timer 2 - ch. 5 (IC/OC)	I/O						
		0010	CS0	DSPI_2	DSPI 2 - Chip Select 0	I/O						
		0011	CS3	DSPI_0	DSPI 0 - Chip Select 3	O						
		0100	CS2	DSPI_3	DSPI 3 - Chip Select 2	O						
		0101	—	Reserved		—						
		0110	MDO_11	NPC_ WRAPPER	Nexus MDO 11	O						
		0111-1000	—	Reserved	—	—						
		1001	DEBUG_0	FLEXRAY_0	Flexray - Debug[0]	O						
		1010-1111	—	Reserved	—	—						
	520	00000111	M_CAN_ 1_RX	CANSUBSY S_1	MCAN 1 Receive Data	I						
	632	00000100	CS0	DSPI_2	DSPI2 Slave Select Input	I						
	705	00000010	ETC5	ETIMER_2	ETMR2 input channel 5	I						
PE[3]	67	0000	GPIO67	SIUL2-GPIO67	GPIO67	I/O	x	x	x	x	IO #0	IO #0
		0001	MSEO_B _0	NPC_ WRAPPER	Nexus MSEO	O						
		0010	SCK	DSPI_3	DSPI 3 - Serial Clock	I/O						
		0011	CS4	DSPI_0	DSPI 0 - Chip Select 4	O						
		0100	SCK	DSPI_2	DSPI 2 - Serial Clock	I/O						
		0101	—	Reserved		—						
		0110	MDO_10	NPC_ WRAPPER	Nexus MDO 10	O						
		0111-1000	—	Reserved	—							
		1001	DEBUG_1	FLEXRAY_0	Flexray - Debug[1]	O						
		1010-1111	—	Reserved	—	—						
	631	00000110	SCK	DSPI_2	DSPI2 Serial Clock Input	I						
	634	00000100	SCK	DSPI_3	DSPI3 Serial Clock Input	I						

Table 12. Pin multiplexing table description (continued)

Port Pin	SIUL2 MSCR		Signal	Module	Full Signal Name	Direction	PAD strength				IO Segment	
	No	SSS Value					VeryStrong	Fast	Medium	Slow	eTQFP 100	eTQFP 144
PE[4]	68	0000	GPIO68	SIUL2-GPIO68	GPIO68	I/O	x	x	x	x	IO #0	IO #0
		0001	ETC5	ETIMER_3	Timer 3 - Ch. 5 (IC/OC)	—						
		0010	CS2	DSPI_2	DSPI 2 - Chip Select 2	O						
		0011	ETC4	ETIMER_2	Timer 2 - ch. 4 (IC/OC)	I/O						
		0100	CS0	DSPI_2	DSPI 2 - Chip Select 0	I/O						
		0101	—	Reserved		—						
		0110	MDO_9	NPC_ WRAPPER	Nexus MDO 9	O						
		0111-1111	—	Reserved	—	—						
	632	00000110	CS0	DSPI_2	DSPI2 Slave Select Input	I						
	704	00000100	ETC4	ETIMER_2	ETMR2 input channel 4	I						
	711	00000001	ETC5	ETIMER_3	ETMR3 input channel 5	I						
PE[5]	69	0000	GPIO69	SIUL2-GPIO69	GPIO69	I/O	x	x	x	x	IO #0	IO #0
		0001	MDO_3	NPC_ WRAPPER	Nexus MDO 3	O						
		0010	SOUT	DSPI_3	DSPI 3 - Serial Data	I/O						
		0011	CLKOUT0	MC_CGL	Ext. INT 5 and Divided Clock Output	O						
		0100	SOUT	DSPI_2	DSPI 2 - Serial Data	I/O						
		0101	—	Reserved		—						
		0110	MDO_7	NPC_ WRAPPER	Nexus MDO 7	O						
		0111	A1	PWM_0	FlexPWM 0 - A[1]	I/O						
		1000	—	Reserved		—						
		1001	DEBUG_2	FLEXRAY_0	Flexray - Debug[2]	O						
		1010-1111	—	Reserved	—	—						
	630	00000101	SIN	DSPI_2	DSPI2 Serial Data Input	I						
	633	00000011	SIN	DSPI_3	DSPI3 Serial Data Input	I						
	953	00000111	A1	PWM_0	FlexPWM 0 - A[1] input channel	I						

Table 12. Pin multiplexing table description (continued)

Port Pin	SIUL2 MSCR		Signal	Module	Full Signal Name	Direction	PAD strength				IO Segment	
	No	SSS Value					VeryStrong	Fast	Medium	Slow	eTQFP 100	eTQFP 144
PE[6]	70	0000	GPIO70	SIUL2-GPIO70	GPIO70	I/O	x	x	x	x	IO #0	IO #0
		0001	MDO_2	NPC_ WRAPPER	Nexus MDO 2	O						
		0010	SOUT	DSPI_3	DSPI 3 - Serial Data	I/O						
		0011	CS6	DSPI_0	DSPI 0 - Chip Select 6	O						
		0100	SOUT	DSPI_2	DSPI 2 - Serial Data	I/O						
		0101	—	Reserved		—						
		0110	MDO_6	NPC_ WRAPPER	Nexus MDO 6	O						
		0111	A2	PWM_0	FlexPWM 0 - A[2]	I/O						
		1000-1111	—	Reserved	—	—						
	630	00000110	SIN	DSPI_2	DSPI2 Serial Data Input	I						
	633	00000100	SIN	DSPI_3	DSPI3 Serial Data Input	I						
	954	00000111	A2	PWM_0	FlexPWM 0 - A[2] input channel	I						
PE[7]	71	0000	GPIO71	SIUL2-GPIO71	GPIO71	I/O	x	x	x	x	IO #0	IO #0
		0001	MDO_1	NPC_ WRAPPER	Nexus MDO 1	O						
		0010	CS0	DSPI_3	DSPI3 - Chip Select 0	I/O						
		0011	CS7	DSPI_0	DSPI 0 - Chip Select 7	O						
		0100	ETC4	ETIMER_3	Timer 3 - ch. 4 (IC/OC)	I/O						
		0101	—	Reserved		—						
		0110	MDO_5	NPC_ WRAPPER	Nexus MDO 5	O						
		0111	A3	PWM_0	FlexPWM 0 - A[3]	I/O						
		1000-1111	—	Reserved	—	—						
	635	00000100	CS0	DSPI_3	DSPI3 Slave Select Input	I						
	710	00000110	ETC4	ETIMER_3	ETMR3 input channel 4	I						
	955	00000111	A3	PWM_0	FlexPWM 0 - A[3] input channel	I						

Table 12. Pin multiplexing table description (continued)

Port Pin	SIUL2 MSCR		Signal	Module	Full Signal Name	Direction	PAD strength				IO Segment	
	No	SSS Value					VeryStrong	Fast	Medium	Slow	eTQFP 100	eTQFP 144
PE[8]	72	0000	GPIO72	SIUL2-GPIO72	GPIO72	I/O	x	x	x	x	IO #0	IO #0
		0001	MDO_0	NPC_ WRAPPER	Nexus MDO 0	O						
		0010	CS0	DSPI_0	DSPI0 - Chip Select 0	I/O						
		0011	—	Reserved		—						
		0100	ETC5	ETIMER_3	Timer 3 - ch. 5 (IC/OC)	I/O						
		0101	MDO_3	NPC_ WRAPPER	Nexus MDO 3	O						
		0110	—	Reserved		—						
		0111	B1	PWM_0	FlexPWM 0 - B[1]	I/O						
		1000	—	Reserved		—						
		1001	DEBUG_3	FLEXRAY_0	Flexray - Debug[3]	O						
		1010-1111	—	Reserved	—	—						
	626	00000100	CS0	DSPI_0	DSPI0 Slave Select Input	I						
	711	00000110	ETC5	ETIMER_3	ETMR3 input channel 5	I						
	819	00000100	REQ3	SIUL	interrupt request channel 2	I						
	957	00000111	B1	PWM_0	FlexPWM 0 - B[1] input channel	I						

Table 12. Pin multiplexing table description (continued)

Port Pin	SIUL2 MSCR		Signal	Module	Full Signal Name	Direction	PAD strength				IO Segment	
	No	SSS Value					VeryStrong	Fast	Medium	Slow	eTQFP 100	eTQFP 144
PE[9]	73	0000	GPIO73	SIUL2-GPIO73	GPIO73	I/O	x	x	x	x	IO #0	IO #0
		0001	SOUT	DSPI_3	DSPI3 - Serial Data	I/O						
		0010	ETC2	ETIMER_3	Timer 3 - ch. 2 (IC/OC)	I/O						
		0011	—	Reserved		—						
		0100	CS1	DSPI_2	DSPI 2 - Chip Select 1	O						
		0101	MDO_2	NPC_ WRAPPER	Nexus MDO 2	O						
		0110	—	Reserved		—						
		0111	B2	PWM_0	FlexPWM 0 - B[2]	I/O						
		1000-1111	—	Reserved	—	—						
	633	00000010	SIN	DSPI_3	DSPI3 Serial Data Input	I						
	708	00000011	ETC2	ETIMER_3	ETMR3 input channel 2	I						
	820	00000100	REQ4	SIUL	interrupt request channel 2	I						
	958	00000111	B2	PWM_0	FlexPWM 0 - B[2] input channel	I						
PE[10]	74	0000	GPIO74	SIUL2-GPIO74	GPIO74	I/O	x	x	x	x	IO #0	IO #0
		0001	SCK	DSPI_3	DSPI3 - Serial Clock	I/O						
		0010	ETC3	ETIMER_3	Timer 3 - ch. 3 (IC/OC)	I/O						
		0011	CS5	DSPI_0	DSPI 0 - Chip Select 5	—						
		0100	CS2	DSPI_2	DSPI 2 - Chip Select 2	O						
		0101	MDO_1	NPC_ WRAPPER	Nexus MDO 1	O						
		0110	—	Reserved		—						
		0111	B3	PWM_0	FlexPWM 0 - B[3]	I/O						
		1000-1111	—	Reserved	—	—						
	634	00000010	SCK	DSPI_3	DSPI3 Serial Clock Input	I						
	709	00000011	ETC3	ETIMER_3	ETMR3 input channel 3	I						
	821	00000100	REQ5	SIUL	interrupt request channel 2	I						
	959	00000111	B3	PWM_0	FlexPWM 0 - B[3] input channel	I						

Table 12. Pin multiplexing table description (continued)

Port Pin	SIUL2 MSCR		Signal	Module	Full Signal Name	Direction	PAD strength				IO Segment	
	No	SSS Value					VeryStrong	Fast	Medium	Slow	eTQFP 100	eTQFP 144
PE[11]	75	0000	GPIO75	SIUL2-GPIO75	GPIO75	I/O	x	x	x	x	IO #0	IO #0
		0001	MCKO	NPC_ WRAPPER	Nexus MCK0	O						
		0010	SCK	DSPI_0	—	I/O						
		0011	CS1	DSPI_0	DSPI 0 - Chip Select 1	O						
		0100	CS3	DSPI_1	DSPI 1 - Chip Select 3	O						
		0101	MDO_0	NPC_ WRAPPER	Nexus MDO 0	O						
		0110	—	Reserved		—						
		0111	SOUT	DSPI_2	DSPI 2 - Serial Data	I/O						
		1000	—	Reserved		—						
		1001	X0	PWM_0	FlexPWM 0 - X[0]	I/O						
		1010-1111	—	Reserved	—	—						
	625	00000100	SCK	DSPI_0	DSPI0 Serial Clock Input	I						
	630	00001001	SIN	DSPI_2	DSPI2 Serial Data Input	I						
	976	00000101	X0	PWM_0	FlexPWM 0 - X[0] input channel	I						

Table 12. Pin multiplexing table description (continued)

Port Pin	SIUL2 MSCR		Signal	Module	Full Signal Name	Direction	PAD strength				IO Segment	
	No	SSS Value					VeryStrong	Fast	Medium	Slow	eTQFP 100	eTQFP 144
PE[12]	76	0000	GPIO76	SIUL2-GPIO76	GPIO76	I/O	x	x	x	x	IO #0	IO #0
		0001	CS0	DSPI_3	DSPI3 - Chip Select 0	I/O						
		0010	ETC4	ETIMER_3	Timer 3 - ch. 4 (IC/OC)	I/O						
		0011	CS0	DSPI_2	DSPI 2 - Chip Select 0	I/O						
		0100	CS2	DSPI_1	DSPI 1 - Chip Select 2	O						
		0101	MCKO	NPC_ WRAPPER	Nexus MCK0	O						
		0110	—	Reserved		—						
		0111	A1	PWM_3	FlexPWM 3 - A[1] (SWG PWM 1)	I/O						
		1000	—	Reserved		—						
		1001	X1	PWM_0	FlexPWM 0 - X[1]	I/O						
		1010-1111	—	Reserved	—	—						
	632	00000101	CS0	DSPI_2	DSPI2 Slave Select Input	I						
	635	00000010	CS0	DSPI_3	DSPI3 Slave Select Input	I						
	710	00000011	ETC4	ETIMER_3	ETMR3 input channel 4	I						
	973	00000111	A1	PWM_3	FlexPWM 3 - A[1] input channel	I						
	977	00000101	X1	PWM_0	FlexPWM 0 - X[1] input channel	I						

Table 12. Pin multiplexing table description (continued)

Port Pin	SIUL2 MSCR		Signal	Module	Full Signal Name	Direction	PAD strength				IO Segment	
	No	SSS Value					VeryStrong	Fast	Medium	Slow	eTQFP 100	eTQFP 144
PE[13]	77	0000	GPIO77	SIUL2-GPIO77	GPIO77	I/O	x	x	x	x	IO #0	IO #0
		0001	SOUT	DSPI_3	DSPI3 - Serial Data	I/O						
		0010	ETC5	ETIMER_3	Timer 3 - ch. 5 (IC/OC)	I/O						
		0011	CS1	DSPI_2	DSPI 2 - Chip Select 1	O						
		0100	CS1	DSPI_1	DSPI 1 - Chip Select 1	O						
		0101	MSEO_B_0	NPC_WRAPPER	Nexus MSEO 0	O						
		0110	—	Reserved		—						
		0111	B1	PWM_2	FlexPWM 2 - B[1]	I/O						
		1000	—	Reserved		-						
		1001	X2	PWM_0	FlexPWM 0 - X[2]	I/O						
		1010-1111	—	Reserved	—	—						
	633	00000001	SIN	DSPI_3	DSPI3 Serial Data Input	I						
	711	00000011	ETC5	ETIMER_3	ETMR3 input channel 5	I						
	969	00000111	B1	PWM_2	FlexPWM 2 - B[1] input channel	I						
	978	00000101	X2	PWM_0	FlexPWM 0 - X[2] input channel	I						

Table 12. Pin multiplexing table description (continued)

Port Pin	SIUL2 MSCR		Signal	Module	Full Signal Name	Direction	PAD strength				IO Segment	
	No	SSS Value					VeryStrong	Fast	Medium	Slow	eTQFP 100	eTQFP 144
PE[14]	78	0000	GPIO78	SIUL2-GPIO78	GPIO78	I/O	x	x	x	x	IO #0	IO #0
		0001	EVTO_B0	DCI	Nexus EVTO	O						
		0010	SOUT	DSPI_0	DSPI 0 - Serial Data	I/O						
		0011	CS2	DSPI_0	DSPI 0 - Chip Select 2	O						
		0100	CS3	DSPI_2	DSPI 2 - Chip Select 3	O						
		0101-0110	—	Reserved	—	—						
		0111	B2	PWM_2	FlexPWM 2 - B[2]	I/O						
		1000	—	Reserved		—						
		1001	X3	PWM_0	FlexPWM 0 - X[3]	I/O						
		1010-1111	—	Reserved	—	—						
	624	00000011	SIN	DSPI_0	DSPI0 Serial Data Input	I						
	970	00000111	B2	PWM_2	FlexPWM 2 - B[2] input channel	I						
	979	00000101	X3	PWM_0	FlexPWM 0 - X[3] input channel	I						
PE[15]	79	0000	GPIO79	SIUL2-GPIO79	GPIO79	I/O	x	x	x	x	IO #0	IO #0
		0001	IOMUX_EVTI_B01	INTG_DEBUG	Nexus EVTI	I						
		0010	SOUT	DSPI_0	DSPI 0 - Serial Data	I/O						
		0011	CS3	DSPI_1	DSPI 1 - Chip Select 3	O						
		0100	CS3	DSPI_3	DSPI 3 - Chip Select 3	O						
		0101-0110	—	Reserved	—	—						
		0111	B3	PWM_2	FlexPWM 2 - B[3]	I/O						
		1000-1111	—	Reserved	—	—						
	624	00000100	SIN	DSPI_0	DSPI0 Serial Data Input	I						
	971	00000111	B3	PWM_2	FlexPWM 2 - B[3] input channel	I						

Table 12. Pin multiplexing table description (continued)

Port Pin	SIUL2 MSCR		Signal	Module	Full Signal Name	Direction	PAD strength				IO Segment	
	No	SSS Value					VeryStrong	Fast	Medium	Slow	eTQFP 100	eTQFP 144
PF[0]	80	0000	GPIO80	SIUL2-GPIO80	GPIO80	I/O	—	—	x	x	—	IO #0
		0001-0100	—	Reserved	—	—						
		0101	CLKOUT0	MC_CGL	Divided Clock Output	O						
		0110-1111	—	Reserved	—	—						
PF[1]	81	0000	GPIO81	SIUL2-GPIO81	GPIO81	I/O	x	x	x	x	—	IO #0
		0001-0100	—	Reserved	—	—						
		0101	SOUT	DSPI_1	DSPI 1 - Serial Data	I/O						
		0110	—	Reserved		—						
		0111	SCK	DSPI_0	DSPI 0 - Serial Clock	I/O						
		1000	CS1	DSPI_0	DSPI 0 - Chip Select 1	O						
		1001	CS3	DSPI_1	DSPI 1 - Chip Select 3	O						
		1010-1111	—	Reserved	—	—						
	625	00001010	SCK	DSPI_0	DSPI0 Serial Clock Input	I						
	627	00001000	SIN	DSPI_1	DSPI1 Serial Data Input	I						
PF[2]	82	0000	GPIO82	SIUL2-GPIO82	GPIO82	I/O	x	x	x	x	—	IO #0
		0001-0100	—	Reserved	—	—						
		0101	SOUT	DSPI_0	DSPI 0 - Serial Data	I/O						
		0110	ETC0	ETIMER_1	Timer 1 - ch. 0(IC/OC)	I/O						
		0111	—	Reserved		—						
		1000	CS2	DSPI_0	DSPI 0 - Chip Select 2	O						
		1001	CS3	DSPI_2	DSPI 2 - Chip Select 3	O						
		1010-1111	—	Reserved	—	—						
	624	00001000	SIN	DSPI_0	DSPI0 Serial Data Input	I						
	694	00001001	ETC0	ETIMER_1	ETMR1 input channel 0	I						

Table 12. Pin multiplexing table description (continued)

Port Pin	SIUL2 MSCR		Signal	Module	Full Signal Name	Direction	PAD strength				IO Segment	
	No	SSS Value					VeryStrong	Fast	Medium	Slow	eTQFP 100	eTQFP 144
PF[3]	83	0000	GPIO83	SIUL2-GPIO83	GPIO83	I/O	—	—	x	x	—	IO #0
		0001-0100	—	Reserved	—	—						
		0101	A1	PWM_0	FlexPWM 0 - A[1]	I/O						
		0110	—	Reserved		—						
		0111	CS2	DSPI_0	DSPI 0 - Chip Select 2	O						
		1000	SOUT	DSPI_3	DSPI 3 - Serial Data	I/O						
		1001	CLKOUT0	MC_CGL	Divided Clock Output	O						
		1010	SOUT	DSPI_2	DSPI 2 - Serial Data	I/O						
		1011-1111	—	Reserved	—	—						
	630	00001010	SIN	DSPI_2	DSPI2 Serial Data Input	I						
	633	00001011	SIN	DSPI_3	DSPI3 Serial Data Input	I						
	953	00000101	A1	PWM_0	FlexPWM 0 - A[1] input channel	I						
PF[4]	84	0000	GPIO84	SIUL2-GPIO84	GPIO84	I/O	—	—	x	x	—	IO #0
		0001-0100	—	Reserved	—	—						
		0101	A2	PWM_0	FlexPWM 0 - A[2]	I/O						
		0110	—	Reserved		—						
		0111	ETC4	ETIMER_3	Timer 3 - ch. 4(IC/OC)	I/O						
		1000	SOUT	DSPI_3	DSPI 3 - Serial Data	I/O						
		1001	CS6	DSPI_0	DSPI 0 - Chip Select 6	O						
		1010	SOUT	DSPI_2	DSPI 2 - Serial Data	I/O						
		1011-1111	—	Reserved	—	—						
	630	00001011	SIN	DSPI_2	DSPI2 Serial Data Input	I						
	633	00001100	SIN	DSPI_3	DSPI3 Serial Data Input	I						
	710	00001000	ETC4	ETIMER_3	ETMR3 input channel 4	I						
	954	00000101	A2	PWM_0	FlexPWM 0 - A[2] input channel	I						

Table 12. Pin multiplexing table description (continued)

Port Pin	SIUL2 MSCR		Signal	Module	Full Signal Name	Direction	PAD strength				IO Segment	
	No	SSS Value					VeryStrong	Fast	Medium	Slow	eTQFP 100	eTQFP 144
PF[5]	85	0000	GPIO85	SIUL2-GPIO85	GPIO85	I/O	—	—	x	x	—	IO #0
		0001-0100	—	Reserved	—	—						
		0101	B2	PWM_0	FlexPWM 0 - B[2]	I/O						
		0110	TXEN_A	FLEXRAY_0	FlexRay 0 - TXEN_A	O						
		0111	—	Reserved		—						
		1000	SCK	DSPI_3	DSPI 3 - Serial Clock	I/O						
		1001	CS4	DSPI_0	DSPI 0 - Chip Select 4	O						
		1010	SCK	DSPI_2	DSPI 2 - Serial Clock	I/O						
		1011-1111	—	Reserved	—	—						
	631	00001001	SCK	DSPI_2	DSPI2 Serial Clock Input	I						
	634	00001001	SCK	DSPI_3	DSPI3 Serial Clock Input	I						
	958	00000101	B2	PWM_0	FlexPWM 0 - B[2] input channel	I						
PF[6]	86	0000	GPIO86	SIUL2-GPIO86	GPIO86	I/O	—	—	x	x	—	IO #0
		0001-0100	—	Reserved	—	—						
		0101	TXEN_A	FLEXRAY_0	FlexRay 0 - TXEN_A	O						
		0110	B2	PWM_0	FlexPWM 0 - B[2]	I/O						
		0111-1111	—	Reserved	—	—						
	958	00000110	B2	PWM_0	FlexPWM 0 - B[2] input channel	I						

Table 12. Pin multiplexing table description (continued)

Port Pin	SIUL2 MSCR		Signal	Module	Full Signal Name	Direction	PAD strength				IO Segment	
	No	SSS Value					VeryStrong	Fast	Medium	Slow	eTQFP 100	eTQFP 144
PF[7]	87	0000	GPIO87	SIUL2-GPIO87	GPIO87	I/O	x	x	x	x	—	IO #0
		0001-0100	—	Reserved	—	—						
		0101	M_CAN_2_TX	CANSUBSYS_0	MCAN 2 - TX	O						
		0110	—	Reserved		—						
		0111	CS0	DSPI_0	DSPI 0 - Chip Select 0	I/O						
		1000	—	Reserved		—						
		1001	ETC5	ETIMER_3	Timer 3- ch. 5(IC/OC)	I/O						
		1010-1111	—	Reserved	—	—						
	626	00001011	CS0	DSPI_0	DSPI0 Slave Select Input	I						
	711	00001000	ETC5	ETIMER_3	ETMR3 input channel 5	I						
	819	00000110	REQ3	SIUL	interrupt request channel 2	I						
PF[8]	88	0000	GPIO88	SIUL2-GPIO88	GPIO88	I/O	—	—	x	x	—	IO #0
		0001-0110	—	Reserved	—	—						
		0111	CS2	DSPI_2	DSPI 2 - Chip Select 2	O						
		1000	ETC4	ETIMER_2	Timer 2 - ch. 4(IC/OC)	I/O						
		1001	ETC5	ETIMER_3	Timer 3- ch. 5(IC/OC)	I/O						
		1010	CS0	DSPI_2	DSPI 2 - Chip Select 0	I/O						
		1011-1111	—	Reserved	—	—						
	513	00000101	M_CAN_2_RX	CANSUBSYS_0	MCAN 2 Receive Data	I						
	632	00001010	CS0	DSPI_2	DSPI2 Slave Select Input	I						
	704	00001001	ETC4	ETIMER_2	ETMR2 input channel 4	I						
	711	00001001	ETC5	ETIMER_3	ETMR3 input channel 5	I						

Table 12. Pin multiplexing table description (continued)

Port Pin	SIUL2 MSCR		Signal	Module	Full Signal Name	Direction	PAD strength				IO Segment	
	No	SSS Value					VeryStrong	Fast	Medium	Slow	eTQFP 100	eTQFP 144
PF[9]	89	0000	GPIO89	SIUL2-GPIO89	GPIO89	I/O	—	—	x	x	—	IO #0
		0001-0110	—	Reserved	—	—						
		0111	CS7	DSPI_0	DSPI 0 - Chip Select 7	O						
		1000	ETC3	ETIMER_3	Timer 3 - ch. 3(IC/OC)	I/O						
		1001-1111	—	Reserved	—	—						
	512	00000101	M_CAN_0_RX	CANSUBSYS_0	MCAN 0 Receive Data	I						
	709	00001001	ETC3	ETIMER_3	ETMR3 input channel 3	I						
PF[10]	90	0000	GPIO90	SIUL2-GPIO90	GPIO90	I/O	—	—	x	x	—	IO #0
		0001-0100	—	Reserved	—	—						
		0101	ETC0	ETIMER_0	Timer 0 - ch. 0 (IC/OC)	I/O						
		0110	—	Reserved		—						
		0111	ETC4	ETIMER_1	Timer 1 - ch. 4(IC/OC)	I/O						
		1000-1111	—	Reserved	—	—						
	688	00001000	ETC0	ETIMER_0	ETMR0 input channel 0	I						
	698	00001000	ETC4	ETIMER_1	ETMR1 input channel 4	I						
PF[11]	91	0000	GPIO91	SIUL2-GPIO91	GPIO91	I/O	—	—	x	x	—	IO #0
		0001-0100	—	Reserved	—	—						
		0101	ETC1	ETIMER_0	Timer 0 - ch. 1 (IC/OC)	I/O						
		0110-1111	—	Reserved	—	—						
	689	00001000	ETC1	ETIMER_0	ETMR0 input channel 1	I						

Table 12. Pin multiplexing table description (continued)

Port Pin	SIUL2 MSCR		Signal	Module	Full Signal Name	Direction	PAD strength				IO Segment	
	No	SSS Value					VeryStrong	Fast	Medium	Slow	eTQFP 100	eTQFP 144
PF[12]	92	0000	GPIO92	SIUL2-GPIO92	GPIO92	I/O	—	—	x	x	—	IO #0
		0001-0100	—	Reserved	—	—						
		0101	ETC2	ETIMER_0	Timer 0 - ch. 2 (IC/OC)	I/O						
		0110	—	Reserved		—						
		0111	ETC5	ETIMER_1	Timer 1 - ch. 5(IC/OC)	I/O						
		1000	—	Reserved		—						
		1001	MCAN1TX_AND_MCAN2TX_O	CANSUBSYS_0	MCAN 0 - TX AND MCAN2 - TX	O						
		1010-1111	—	Reserved	—	—						
	690	00001000	ETC2	ETIMER_0	ETMR0 input channel 2	I						
	699	00001000	ETC5	ETIMER_1	ETMR1 input channel 5	I						
PF[13]	93	0000	GPIO93	SIUL2-GPIO93	GPIO93	I/O	—	—	x	x	—	IO #0
		0001-0100	—	Reserved	—	—						
		0101	ETC3	ETIMER_0	Timer 0 - ch. 3 (IC/OC)	I/O						
		0110	—	Reserved		—						
		0111	ETC3	ETIMER_2	Timer 2 - ch. 2(IC/OC)	I/O						
		1000-1111	—	Reserved	—	—						
	691	00001000	ETC3	ETIMER_0	ETMR0 input channel 3	I						
	703	00001011	ETC3	ETIMER_2	ETMR2 input channel 3	I						
PF[14]	94	0000	GPIO94	SIUL2-GPIO94	GPIO94	I/O	—	—	—	—	—	ADC #0
		0001-1111	—	Reserved	—	—						
	—	—	ANP[0]	SAR_ADC_SV0	analog channel							
	—	—	ANP[0]	SAR_ADC_0	analog channel							

Table 12. Pin multiplexing table description (continued)

Port Pin	SIUL2 MSCR		Signal	Module	Full Signal Name	Direction	PAD strength				IO Segment	
	No	SSS Value					VeryStrong	Fast	Medium	Slow	eTQFP 100	eTQFP 144
PF[15]	95	0000	GPIO95	SIUL2-GPIO95	GPIO95	I/O	—	—	—	—	—	ADC #0
		0001-1111	—	Reserved	—	—						
	—	—	ANP[4]	SAR_ADC_SV0	analog channel							
	—	—	ANP[4]	SAR_ADC_0	analog channel							
PG[0]	96	0000	GPIO96	SIUL2-GPIO96	GPIO96	I/O	—	—	—	—	—	ADC #0
		0001-1111	—	Reserved	—	—						
	—	—	ANP[9]	SAR_ADC_SV0	analog channel							
	—	—	ANP[3]	SAR_ADC_1	analog channel							
PG[1]	97	0000	GPIO97	SIUL2-GPIO97	GPIO97	I/O	—	—	—	—	—	ADC #0
		0001-1111	—	Reserved	—	—						
	—	—	ANP[13]	SAR_ADC_SV0	analog channel							
	—	—	ANP[2]	SAR_ADC_2	analog channel							
PG[2]	98	0000	GPIO98	SIUL2-GPIO98	GPIO98	I/O	—	—	—	—	—	ADC #1
		0001-1111	—	Reserved	—	—						
	992	00000110	FAULT0	PWM_2	FlexPWM 2 - FAULT[0] input channel	I						
	—	—	ANP[0]	SAR_ADC_SV1	analog channel							
	—	—	ANP[0]	SAR_ADC_3	analog channel							
PG[3]	99	0000	GPIO99	SIUL2-GPIO99	GPIO99	I/O	—	—	—	—	—	ADC #1
		0001-1111	—	Reserved	—	—						
	993	00000110	FAULT1	PWM_2	FlexPWM 2 - FAULT[1] input channel	I						
	—	—	ANP[1]	SAR_ADC_SV1	analog channel							
	—	—	ANP[1]	SAR_ADC_3	analog channel							

Table 12. Pin multiplexing table description (continued)

Port Pin	SIUL2 MSCR		Signal	Module	Full Signal Name	Direction	PAD strength				IO Segment	
	No	SSS Value					VeryStrong	Fast	Medium	Slow	eTQFP 100	eTQFP 144
PG[4]	100	0000	GPIO100	SIUL2-GPIO100	GPIO100	I/O	—	—	—	—	—	ADC #1
		0001-1111	—	Reserved	—	—						
	994	00000110	FAULT2	PWM_2	FlexPWM 2 - FAULT[2] input channel	I						
	—	—	ANP[2]	SAR_ADC_SV1	analog channel							
	—	—	ANP[2]	SAR_ADC_3	analog channel							
PG[5]	101	0000	GPIO101	SIUL2-GPIO101	GPIO101	I/O	—	—	—	—	—	ADC #1
		0001-1111	—	Reserved	—	—						
	995	00000110	FAULT3	PWM_2	FlexPWM 2 - FAULT[3] input channel	I						
	—	—	ANP[3]	SAR_ADC_SV1	analog channel							
	—	—	ANP[3]	SAR_ADC_3	analog channel							
PG[6]	102	0000	GPIO102	SIUL2-GPIO102	GPIO102	I/O	—	—	—	—	—	ADC #1
		0001-1111	—	Reserved	—	—						
	996	00000110	EXT_SYNC	PWM_2	FlexPWM 2 - EXT_SYNC input channel	I						
	—	—	ANP[4]	SAR_ADC_SV1	analog channel							
	—	—	ANP[4]	SAR_ADC_3	analog channel							
PG[7]	103	0000	GPIO103	SIUL2-GPIO103	GPIO103	I/O	—	—	—	—	—	ADC #1
	—	0001-1111	—	Reserved	—	—						
	—	—	ANP[5]	SAR_ADC_SV1	analog channel							
	—	—	ANP[5]	SAR_ADC_3	analog channel							
PG[8]	104	0000	GPIO104	SIUL2-GPIO104	GPIO104	I/O	—	—	—	—	—	ADC #1
		0001-1111	—	Reserved	—	—						
	—	—	ANP[6]	SAR_ADC_SV1	analog channel							
	—	—	ANP[0]	SAR_ADC_4	analog channel							

Table 12. Pin multiplexing table description (continued)

Port Pin	SIUL2 MSCR		Signal	Module	Full Signal Name	Direction	PAD strength				IO Segment	
	No	SSS Value					VeryStrong	Fast	Medium	Slow	eTQFP 100	eTQFP 144
PG[9]	105	0000	GPIO105	SIUL2-GPIO105	GPIO105	I/O	—	—	—	—	—	ADC #1
		0001-1111	—	Reserved	—	—						
	—	—	ANP[7]	SAR_ADC_SV1	analog channel							
	—	—	ANP[1]	SAR_ADC_4	analog channel							
PG[10]	106	0000	GPIO106	SIUL2-GPIO106	GPIO106	I/O	—	—	—	—	—	ADC #1
		0001-1111	—	Reserved	—	—						
	—	—	ANP[8]	SAR_ADC_SV1	analog channel							
	—	—	ANP[2]	SAR_ADC_4	analog channel							
PG[11]	107	0000	GPIO107	SIUL2-GPIO107	GPIO107	I/O	—	—	—	—	—	ADC #1
		0001-1111	—	Reserved	—	—						
	—	—	ANP[9]	SAR_ADC_SV1	analog channel							
	—	—	ANP[3]	SAR_ADC_4	analog channel							
PG[12]	108	0000	GPIO108	SIUL2-GPIO108	GPIO108	I/O	—	—	—	—	—	ADC #1
		0001-1111	—	Reserved	—	—						
	—	—	ANP[10]	SAR_ADC_SV1	analog channel							
	—	—	ANP[4]	SAR_ADC_4	analog channel							
PG[13]	109	0000	GPIO109	SIUL2-GPIO109	GPIO109	I/O	—	—	—	—	—	ADC #1
		0001-1111	—	Reserved	—	—						
	—	—	ANP[11]	SAR_ADC_SV1	analog channel							
	—	—	ANP[0]	SAR_ADC_5	analog channel							

Table 12. Pin multiplexing table description (continued)

Port Pin	SIUL2 MSCR		Signal	Module	Full Signal Name	Direction	PAD strength				IO Segment	
	No	SSS Value					VeryStrong	Fast	Medium	Slow	eTQFP 100	eTQFP 144
PG[14]	110	0000	GPIO110	SIUL2-GPIO110	GPIO110	I/O	—	—	—	—	—	ADC #1
		0001-1111	—	Reserved	—	—						
	—	—	ANP[12]	SAR_ADC_SV1	analog channel							
	—	—	ANP[1]	SAR_ADC_5	analog channel							
PG[15]	111	0000	GPIO111	SIUL2-GPIO111	GPIO111	I/O	—	—	—	—	—	ADC #1
		0001-1111	—	Reserved	—	—						
	783	00000011	EXT_IN15	CTU_1	CTU1 input trigger 15	I						
	—	—	ANP[13]	SAR_ADC_SV1	analog channel							
	—	—	ANP[2]	SAR_ADC_5	analog channel							
PH[0]	112	0000	GPIO112	SIUL2-GPIO112	GPIO112	I/O	—	—	—	—	—	ADC #1
		0001-1111	—	Reserved	—	—						
	—	—	ANP[14]	SAR_ADC_SV1	analog channel							
	—	—	ANP[3]	SAR_ADC_5	analog channel							
PH[1]	113	0000	GPIO113	SIUL2-GPIO113	GPIO113	I/O	—	—	—	—	—	ADC #1
		0001-1111	—	Reserved	—	—						
	—	—	ANP[15]	SAR_ADC_SV1	analog channel							
	—	—	ANP[4]	SAR_ADC_5	analog channel							

Table 12. Pin multiplexing table description (continued)

Port Pin	SIUL2 MSCR		Signal	Module	Full Signal Name	Direction	PAD strength				IO Segment	
	No	SSS Value					VeryStrong	Fast	Medium	Slow	eTQFP 100	eTQFP 144
PH[2]	114	0000	GPIO114	SIUL2-GPIO114	GPIO114	I/O	—	—	x	x	—	IO #1
		0001-0100	—	Reserved	—	—						
		0101	A1	PWM_3	FlexPWM 3 - A[1] (SWG PWM 1)	I/O						
		0110	—	Reserved		—						
		0111	ETC1	ETIMER_0	Timer 0 - ch. 1 (IC/OC)	I/O						
		1000-1111	—	Reserved	—	—						
	689	00001010	ETC1	ETIMER_0	ETMR0 input channel 1	I						
	973	00000101	A1	PWM_3	FlexPWM 3 - A[1] input channel	I						
PH[3]	115	0000	GPIO115	SIUL2-GPIO115	GPIO115	I/O	—	—	x	x	—	IO #1
		0001-0110	—	Reserved	—	—						
		0111	ETC0	ETIMER_1	Timer 1 - ch. 0 (IC/OC)	I/O						
		1000	ETC2	ETIMER_0	Timer 0 - ch. 2 (IC/OC)	I/O						
		1001-1111	—	Reserved	—	—						
	690	00001011	ETC2	ETIMER_0	ETMR0 input channel 2	I						
	694	00001011	ETC0	ETIMER_1	ETMR1 input channel 0	I						
PH[4]	116	0000	GPIO116	SIUL2-GPIO116	GPIO116	I/O	—	—	x	x	—	IO #1
		0001-0110	—	Reserved	—	—						
		0111	ETC5	ETIMER_0	Timer 0 - ch. 5 (IC/OC)	I/O						
		1000-1111	—	Reserved	—	—						
	693	00001001	ETC5	ETIMER_0	ETMR0 input channel 5	I						
	996	00000101	EXT_SYNC	PWM_2	FlexPWM 2 - EXT_SYNC input channel	I						

Table 12. Pin multiplexing table description (continued)

Port Pin	SIUL2 MSCR		Signal	Module	Full Signal Name	Direction	PAD strength				IO Segment	
	No	SSS Value					VeryStrong	Fast	Medium	Slow	eTQFP 100	eTQFP 144
PH[5]	117	0000	GPIO117	SIUL2-GPIO117	GPIO117	I/O	—	—	x	x	—	IO #1
		0001-0100	—	Reserved	—	—						
		0101	ETC3	ETIMER_2	Timer 2 - ch. 3 (IC/OC)	I/O						
		0110	B3	PWM_2	FlexPWM 2 -B[3]	I/O						
		0111	ETC0	ETIMER_0	Timer 0 - ch. 0 (IC/OC)	I/O						
		1000	CS0	DSPI_0	DSPI 0 - Chip Select 0	I/O						
		1001-1111	—	Reserved	—	—						
	626	00001001	CS0	DSPI_0	DSPI0 Slave Select Input	I						
	688	00001011	ETC0	ETIMER_0	ETMR0 input channel 0	I						
	703	00001000	ETC3	ETIMER_2	ETMR2 input channel 3	I						
	971	00000110	B3	PWM_2	FlexPWM 2 - B[3] input channel	I						
PH[6]	118	0000	GPIO118	SIUL2-GPIO118	GPIO118	I/O	—	—	x	x	—	IO #1
		0001-0100	—	Reserved	—	—						
		0101	B2	PWM_2	FlexPWM 2- B[2]	I/O						
		0110	—	Reserved		—						
		0111	ETC2	ETIMER_1	Timer 1 - ch. 2 (IC/OC)	I/O						
		1000-1111	—	Reserved	—	—						
	696	00001000	ETC2	ETIMER_1	ETMR1 input channel 2	I						
	970	00000101	B2	PWM_2	FlexPWM 2 - B[2] input channel	I						

Table 12. Pin multiplexing table description (continued)

Port Pin	SIUL2 MSCR		Signal	Module	Full Signal Name	Direction	PAD strength				IO Segment	
	No	SSS Value					VeryStrong	Fast	Medium	Slow	eTQFP 100	eTQFP 144
PH[7]	119	0000	GPIO119	SIUL2-GPIO119	GPIO119	I/O	—	—	x	x	—	IO #1
		0001-0100	—	Reserved	—	—						
		0101	B3	PWM_2	FlexPWM 2 - B[3]	I/O						
		0110	—	Reserved		—						
		0111	ETC3	ETIMER_1	Timer 1 - ch. 3 (IC/OC)	I/O						
		1000-1111	—	Reserved	—	—						
	697	00001000	ETC3	ETIMER_1	ETMR1 input channel 3	I						
	971	00000101	B3	PWM_2	FlexPWM 2 - B[3] input channel	I						
PH[8]	120	0000	GPIO120	SIUL2-GPIO120	GPIO120	I/O	—	—	x	x	—	IO #1
		0001-0100	—	Reserved	—	—						
		0101	SOUT	DSPI_3	DSPI 3 - Serial Data	I/O						
		0110	—	Reserved		—						
		0111	CS0	DSPI_1	DSPI 1 - Chip Select 0	I/O						
		1000-1111	—	Reserved	—	—						
	629	00000111	CS0	DSPI_1	DSPI1 Slave Select Input	I						
	633	00001000	SIN	DSPI_3	DSPI3 Serial Data Input	I						
PH[9]	121	0000	GPIO121	SIUL2-GPIO121	GPIO121	I/O	—	—	x	x	—	IO #1
		0001-0100	—	Reserved	—	—						
		0101	ETC3	ETIMER_2	Timer 2 - ch. 3 (IC/OC)	I/O						
		0110	CS0	DSPI_2	DSPI 2 - Chip Select 0	I/O						
		0111	ETC0	ETIMER_2	Timer 2 - ch. 0 (IC/OC)	I/O						
		1000-1111	—	Reserved	—	—						
	632	00001000	CS0	DSPI_2	DSPI2 Slave Select Input	I						
	700	00001000	ETC0	ETIMER_2	ETMR2 input channel 0	I						
	703	00001001	ETC3	ETIMER_2	ETMR2 input channel 3	I						

Table 12. Pin multiplexing table description (continued)

Port Pin	SIUL2 MSCR		Signal	Module	Full Signal Name	Direction	PAD strength				IO Segment	
	No	SSS Value					VeryStrong	Fast	Medium	Slow	eTQFP 100	eTQFP 144
PH[10]	122	0000	GPIO122	SIUL2-GPIO122	GPIO122	I/O	x	x	x	x	—	IO #1
		0001-0100	—	Reserved	—	—						
		0101	TXD_B	FLEXRAY_0	Flexray 0 - TX_B	O						
		0110	—	Reserved		—						
		0111	SOUT	DSPI_0	DSPI 0 - Serial Data	I/O						
		1000	CS2	DSPI_0	DSPI 0 - Chip Select 2	O						
		1001	CS3	DSPI_2	DSPI 2 - Chip Select 3	O						
		1010-1111	—	Reserved	—	—						
	624	00001100	SIN	DSPI_0	DSPI0 Serial Data Input	I						
PH[11]	123	0000	GPIO123	SIUL2-GPIO123	GPIO123	I/O	—	—	x	x	—	IO #1
		0001-0110	—	Reserved	—	—						
		0111	CS0	DSPI_3	DSPI 3 - Chip Select 0	I/O						
		1000	CS7	DSPI_0	DSPI 0 - Chip Select 7	O						
		1001	ETC4	ETIMER_3	Timer 3- ch. 4 (IC/OC)	I/O						
		1010-1111	—	Reserved	—	—						
	635	00001001	CS0	DSPI_3	DSPI3 Slave Select Input	I						
	710	00001001	ETC4	ETIMER_3	ETMR3 input channel 4	I						
	785	00000101	RXD_B	FLEXRAY_0	Flexray Receive Data Channel B	I						
PH[12]	124	0000	GPIO124	SIUL2-GPIO124	GPIO124	I/O	—	—	x	x	—	IO #1
		0001-0110	—	Reserved	—	—						
		0111	ETC1	ETIMER_2	Timer 2 - ch. 1 (IC/OC)	I/O						
		1000	—	Reserved		—						
		1001	EXT_TGR0	CTU_1	CTU 1 - Trigger Output	O						
		1010-1111	—	Reserved	—	—						
	701	00001000	ETC1	ETIMER_2	ETMR2 input channel 1	I						

Table 12. Pin multiplexing table description (continued)

Port Pin	SIUL2 MSCR		Signal	Module	Full Signal Name	Direction	PAD strength				IO Segment	
	No	SSS Value					VeryStrong	Fast	Medium	Slow	eTQFP 100	eTQFP 144
PH[13]	125	0000	GPIO125	SIUL2-GPIO125	GPIO125	I/O	x	x	x	x	—	IO #0
		0001-0101	—	Reserved	—	—						
		0110	MDO_8	NPC_ WRAPPER	Nexus MDO 8	O						
		0111	ETC2	ETIMER_3	Timer 3 - ch. 2 (IC/OC)	I/O						
		1000-1111	—	Reserved	—	—						
	708	00001000	ETC2	ETIMER_3	ETMR3 input channel 2	I						
PH[14]	126	0000	GPIO126	SIUL2-GPIO126	GPIO126	I/O	x	x	x	x	—	IO #0
		0001-0101	—	Reserved	—	—						
		0110	MDO_4	NPC_ WRAPPER	Nexus MDO 4	O						
		0111	ETC4	ETIMER_2	Timer 2 - ch. 4 (IC/OC)	I/O						
		1000	CS5	DSPI_0	DSPI 0 - Chip Select 5	O						
		1001-1111	—	Reserved	—	—						
	704	00001000	ETC4	ETIMER_2	ETMR2 input channel 4	I						
PH[15]	127	0000	GPIO127	SIUL2-GPIO127	GPIO127	I/O	x	x	x	x	—	IO #0
		0001-0100	—	Reserved	—	—						
		0101	MSEO_B_ 1	NPC_ WRAPPER	Nexus MSEO 1	O						
		0110-1111	—	Reserved	—	—						

Table 12. Pin multiplexing table description (continued)

Port Pin	SIUL2 MSCR		Signal	Module	Full Signal Name	Direction	PAD strength				IO Segment	
	No	SSS Value					VeryStrong	Fast	Medium	Slow	eTQFP 100	eTQFP 144
PI[0]	128	0000	GPIO128	SIUL2-GPIO128	GPIO128	I/O	x	x	x	x	—	IO #0
		0001-0100	—	Reserved	—	—						
		0101	RDY_B	INTG_DEBUG	Nexus RDY_b	O						
		0110	—	Reserved		—						
		0111	ETC0	ETIMER_3	Timer 3 - ch. 0 (IC/OC)	I/O						
		1000-1111	—	Reserved	—	—						
	706	00001001	ETC0	ETIMER_3	ETMR3 input channel 0	I						
PI[1]	129	0000	GPIO129	SIUL2-GPIO129	GPIO129	I	—	—	—	—	ADC #0	—
		0001-1111	—	Reserved	—	—						
		—	ANP[5](1)	SAR_ADC_1	analog channel							
PI[2]	130	0000	GPIO130	SIUL2-GPIO130	GPIO130	I	—	—	—	—	ADC #0	—
		0001-1111	—	Reserved	—	—						
		—	ANP[5]	SAR_ADC_2	analog channel							
PI[3]	131	0000	GPIO131	SIUL2-GPIO131	GPIO131	I	—	—	—	—	—	ADC #0
		0001-1111	—	Reserved	—	—						
		—	ANP[5]	SAR_ADC_1	analog channel							
PAD_FCCU_F0	480	0000	F0	FCCU	FCCU_F0	I/O	x	x	x	x	IO #0	IO #0
		0001-1111	—	Reserved	—	—						
PAD_FCCU_F1S	481	0000	F1	FCCU	FCCU_F1S	I/O	—	—	x	x	IO #0	—
		0001-1111	—	Reserved	—	—						
	948	00000001	F1	FCCU	Fccu fault input 1	I						
PAD_TDI	482	0000	TDI	DCI	TDI	I	—	—	x	x	IO #0	IO #1
		0001-1111	—	Reserved	—	—						
PAD_TMS	483	0000	TMS	DCI	TMS	I	—	—	x	x	IO #0	IO #1
		0001-1111	—	Reserved	—	—						
PAD_TDO	484	0000	TDO	DCI	TDO	I/O	x	x	x	x	IO #0	IO #1
		0001-1111	—	Reserved	—	—						

Table 12. Pin multiplexing table description (continued)

Port Pin	SIUL2 MSCR		Signal	Module	Full Signal Name	Direction	PAD strength				IO Segment	
	No	SSS Value					VeryStrong	Fast	Medium	Slow	eTQFP 100	eTQFP 144
PAD_TCK	485	0000	TCK	MC_CGL	TCK	I	—	—	x	x	IO #0	IO #1
		0001-1111	—	Reserved	—	—						
PAD_FCCU_F1E	500	0000	F1	FCCU	FCCU_F1E	I/O	—	—	x	x	—	IO #1
		0001-1111	—	Reserved	—	—						
	948	00000010	F1	FCCU	Fccu fault input 1	I						
PAD_JCOMP	501	0000	NTRST_PD	DCI	JCOMP	I	—	—	—	—	—	IO #1
		0001-1111	—	Reserved	—	—						
PAD_PWMODE	502	0000-1111	—	Reserved	—	—	—	—	—	—	—	XOSC
PAD_PKG_SEL	503	0000-1111	—	Reserved	—	—	—	—	—	—	IO #0	PKG_SEL

1. This is for eTQFP144 only.

Table 13. Peripheral multiplexing table description

Destination			MSCR Register	Alternate Functions	Source	
Instance	Peripherals	Functions			Peripherals	Functions
cansubsys_0	STA1_CAN SUBSYS1_IPS	M_CAN_0_RX	MSCR[512]	0000_0000	—	disable-low
				0000_0001	I/O-Pad	PD[9]
				0000_0010	I/O-Pad	PB[2]
				0000_0011	I/O-Pad	PC[2]
				0000_0100	—	Reserved
				0000_0101	I/O-Pad	PF[9]
				0000_0110-1111_1111	—	Reserved

Table 13. Peripheral multiplexing table description (continued)

Destination			MSCR Register	Alternate Functions	Source	
Instance	Peripherals	Functions			Peripherals	Functions
cansubsys_0	STA1_CAN SUBSYS1_IPS	M_CAN_2_RX	MSCR[513]	0000_0000	—	disable-low
				0000_0001-0000_0100	—	Reserved
				0000_0101	I/O-Pad	PF[8]
				0000_0110-1111_1111	—	Reserved
cansubsys_1	M_CAN	M_CAN_1_RX	MSCR[520]	0000_0000	—	disable-low
				0000_0001	I/O-Pad	PA[15]
				0000_0010	I/O-Pad	PA[12]
				0000_0011	I/O-Pad	PC[2]
				0000_0100	I/O-Pad	PD[9]
				0000_0101	I/O-Pad	PB[8]
				0000_0110	I/O-Pad	PC[4]
				0000_0111	I/O-Pad	PE[2]
				0000_1000-1111_1111	—	Reserved
linflex_0	LINFLEX3	RXD	MSCR[560]	0000_0000	—	disable-low
				0000_0001	I/O-Pad	PA[12]
				0000_0010	I/O-Pad	PB[9]
				0000_0011	I/O-Pad	PC[12]
				0000_0100-0000_0101	—	Reserved
				0000_0110	I/O-Pad	PA[14]
				0000_0111-1111_1111	—	Reserved

Table 13. Peripheral multiplexing table description (continued)

Destination			MSCR Register	Alternate Functions	Source	
Instance	Peripherals	Functions			Peripherals	Functions
linflex_1	LINFLEX3	RXD	MSCR[561]	0000_0000	—	disable-low
				0000_0001	I/O-Pad	PA[6]
				0000_0010	I/O-Pad	PA[9]
				0000_0011	I/O-Pad	PA[12]
				0000_0100	I/O-Pad	PB[12]
				0000_0101	I/O-Pad	PD[7]
				0000_0110	I/O-Pad	PC[13]
				0000_0111	I/O-Pad	PA[4]
				0000_1000-1111_1111	—	Reserved
linflex_2	LINFLEX3	RXD	MSCR[562]	0000_0000	—	disable-low, 0, 1
				0000_0001	I/O-Pad	PA[0]
				0000_0010	I/O-Pad	PA[1]
				0000_0011	I/O-Pad	PA[2]
				0000_0100	I/O-Pad	PD[2]
				0000_0101	I/O-Pad	PF[5]
				0000_0110-1111_1111	—	Reserved
linflex_3	LINFLEX3	RXD	MSCR[563]	0000_0000	—	disable-low, 0, 1
				0000_0001	I/O-Pad	PD[13]
				0000_0010	I/O-Pad	PD[14]
				0000_0011	I/O-Pad	PD[15]
				0000_0100	I/O-Pad	PD[4]
				0000_0101	I/O-Pad	PF[8]
				0000_0110-1111_1111	—	Reserved

Table 13. Peripheral multiplexing table description (continued)

Destination			MSCR Register	Alternate Functions	Source	
Instance	Peripherals	Functions			Peripherals	Functions
dspi_0	DSPI_IPI	SIN	MSCR[624]	0000_0000	—	disable-low
				0000_0001	I/O-Pad	PA[4]
				0000_0010	I/O-Pad	PA[7]
				0000_0011	I/O-Pad	PE[14]
				0000_0100	I/O-Pad	PE[15]
				0000_0101	I/O-Pad	PC[6]
				0000_0110	I/O-Pad	PC[9]
				0000_0111	I/O-Pad	PB[3]
				0000_1000	I/O-Pad	PF[2]
				0000_1001	I/O-Pad	PA[1]
				0000_1010	I/O-Pad	PA[2]
				0000_1011	I/O-Pad	PD[11]
				0000_1100	I/O-Pad	PH[10]
				0000_1101-1111_1111	—	Reserved
dspi_0	DSPI_IPI	SCK	MSCR[625]	0000_0000	—	disable-low
				0000_0001	—	Reserved
				0000_0010	I/O-Pad	PA[3]
				0000_0011	—	Reserved
				0000_0100	I/O-Pad	PE[11]
				0000_0101	—	Reserved
				0000_0110	I/O-Pad	PC[10]
				0000_0111	—	Reserved
				0000_1000	I/O-Pad	PA[6]
				0000_1001	I/O-Pad	PA[0]
				0000_1010	I/O-Pad	PF[1]
				0000_1011-1111_1111	—	Reserved

Table 13. Peripheral multiplexing table description (continued)

Destination			MSCR Register	Alternate Functions	Source	
Instance	Peripherals	Functions			Peripherals	Functions
dspi_0	DSPI_IPI	CS0	MSCR[626]	0000_0000	—	disable-low
				0000_0001	—	Reserved
				0000_0010	I/O-Pad	PA[0]
				0000_0011	—	Reserved
				0000_0100	I/O-Pad	PE[8]
				0000_0101	—	Reserved
				0000_0110	I/O-Pad	PC[7]
				0000_0111	—	Reserved
				0000_1000	I/O-Pad	PA[8]
				0000_1001	I/O-Pad	PH[5]
				0000_1010	I/O-Pad	PA[14]
				0000_1011	I/O-Pad	PF[7]
				0000_1100-1111_1111	—	Reserved
dspi_1	DSPI_IPI	SIN	MSCR[627]	0000_0000	—	disable-low
				0000_0001	I/O-Pad	PC[6]
				0000_0010	I/O-Pad	PC[9]
				0000_0011	I/O-Pad	PC[8]
				0000_0100	I/O-Pad	PC[7]
				0000_0101	I/O-Pad	PA[4]
				0000_0110	I/O-Pad	PA[7]
				0000_0111	I/O-Pad	PB[4]
				0000_1000	I/O-Pad	PF[1]
				0000_1001	I/O-Pad	PA[6]
				0000_1010-1111_1111	—	Reserved

Table 13. Peripheral multiplexing table description (continued)

Destination			MSCR Register	Alternate Functions	Source	
Instance	Peripherals	Functions			Peripherals	Functions
dspi_1	DSPI_IPI	SCK	MSCR[628]	0000_0000	—	disable-low
				0000_0001	—	Reserved
				0000_0010	I/O-Pad	PC[10]
				0000_0011	—	Reserved
				0000_0100	I/O-Pad	PC[11]
				0000_0101	—	Reserved
				0000_0110	I/O-Pad	PA[3]
				0000_0111-1111_1111	—	Reserved
dspi_1	DSPI_IPI	CS0	MSCR[629]	0000_0000	—	disable-low
				0000_0001	—	Reserved
				0000_0010	I/O-Pad	PC[5]
				0000_0011	—	Reserved
				0000_0100	I/O-Pad	PC[4]
				0000_0101	—	Reserved
				0000_0110	I/O-Pad	PB[1]
				0000_0111	I/O-Pad	PH[8]
				0000_1000-1111_1111	—	Reserved

Table 13. Peripheral multiplexing table description (continued)

Destination			MSCR Register	Alternate Functions	Source	
Instance	Peripherals	Functions			Peripherals	Functions
dspi_2	DSPI_IPI	SIN	MSCR[630]	0000_0000	—	disable-low
				0000_0001	I/O-Pad	PD[6]
				0000_0010	I/O-Pad	PD[12]
				0000_0011	I/O-Pad	PD[14]
				0000_0100	I/O-Pad	PD[15]
				0000_0101	I/O-Pad	PE[5]
				0000_0110	I/O-Pad	PE[6]
				0000_0111	I/O-Pad	PB[5]
				0000_1000	I/O-Pad	PD[1]
				0000_1001	I/O-Pad	PE[11]
				0000_1010	I/O-Pad	PF[3]
				0000_1011	I/O-Pad	PF[4]
				0000_1100-1111_1111	—	Reserved
dspi_2	DSPI_IPI	SCK	MSCR[631]	0000_0000	—	disable-low
				0000_0001	—	Reserved
				0000_0010	I/O-Pad	PD[13]
				0000_0011	—	Reserved
				0000_0100	I/O-Pad	PD[11]
				0000_0101	—	Reserved
				0000_0110	I/O-Pad	PE[3]
				0000_0111-0000_1000	—	Reserved
				0000_1001	I/O-Pad	PF[5]
				0000_1010-1111_1111	—	Reserved

Table 13. Peripheral multiplexing table description (continued)

Destination			MSCR Register	Alternate Functions	Source	
Instance	Peripherals	Functions			Peripherals	Functions
dspi_2	DSPI_IPI	CS0	MSCR[632]	0000_0000	—	disable-low
				0000_0001	—	Reserved
				0000_0010	I/O-Pad	PD[5]
				0000_0011	—	Reserved
				0000_0100	I/O-Pad	PE[2]
				0000_0101	I/O-Pad	PE[12]
				0000_0110	I/O-Pad	PE[4]
				0000_0111	—	Reserved
				0000_1000	I/O-Pad	PH[9]
				0000_1001	I/O-Pad	PA[14]
				0000_1010	I/O-Pad	PF[8]
				0000_1011-1111_1111	—	Reserved
dspi_3	DSPI_IPI	SIN	MSCR[633]	0000_0000	—	disable-low
				0000_0001	I/O-Pad	PE[13]
				0000_0010	I/O-Pad	PE[9]
				0000_0011	I/O-Pad	PE[5]
				0000_0100	I/O-Pad	PE[6]
				0000_0101	I/O-Pad	PD[14]
				0000_0110	I/O-Pad	PD[15]
				0000_0111	I/O-Pad	PB[6]
				0000_1000	I/O-Pad	PH[8]
				0000_1001	I/O-Pad	PD[12]
				0000_1010	I/O-Pad	PA[4]
				0000_1011	I/O-Pad	PF[3]
				0000_1100	I/O-Pad	PF[4]
				0000_1101-1111_1111	—	Reserved

Table 13. Peripheral multiplexing table description (continued)

Destination			MSCR Register	Alternate Functions	Source	
Instance	Peripherals	Functions			Peripherals	Functions
dspi_3	DSPI_IPI	SCK	MSCR[634]	0000_0000	—	disable-low
				0000_0001	—	Reserved
				0000_0010	I/O-Pad	PE[10]
				0000_0011	—	Reserved
				0000_0100	I/O-Pad	PE[3]
				0000_0101	—	Reserved
				0000_0110	I/O-Pad	PD[11]
				0000_0111	—	Reserved
				0000_1000	I/O-Pad	PC[9]
				0000_1001	I/O-Pad	PF[5]
				0000_1010-1111_1111	—	Reserved
dspi_3	DSPI_IPI	CS0	MSCR[635]	0000_0000	—	disable-low
				0000_0001	—	Reserved
				0000_0010	I/O-Pad	PE[12]
				0000_0011	—	Reserved
				0000_0100	I/O-Pad	PE[7]
				0000_0101	—	Reserved
				0000_0110	I/O-Pad	PD[12]
				0000_0111	I/O-Pad	PD[14]
				0000_1000	I/O-Pad	PE[0]
				0000_1001	I/O-Pad	PH[11]
				0000_1010-1111_1111	—	Reserved

Table 13. Peripheral multiplexing table description (continued)

Destination			MSCR Register	Alternate Functions	Source	
Instance	Peripherals	Functions			Peripherals	Functions
etimer_0	ETIMER	ETC0	MSCR[688]	0000_0000	—	disable-low
				0000_0001	I/O-Pad	PB[3]
				0000_0010	I/O-Pad	PC[12]
				0000_0011	I/O-Pad	PA[1]
				0000_0100	I/O-Pad	PA[3]
				0000_0101	I/O-Pad	PA[0]
				0000_0110	—	Reserved
				0000_0111	ctu_0	EXT_TGR1
				0000_1000	I/O-Pad	PF[10]
				0000_1001	I/O-Pad	PA[6]
				0000_1010	I/O-Pad	PC[11]
				0000_1011	I/O-Pad	PH[5]
				0000_1100-1111_1111	—	Reserved
etimer_0	ETIMER	ETC1	MSCR[689]	0000_0000	—	disable-low
				0000_0001	I/O-Pad	PB[4]
				0000_0010	I/O-Pad	PC[13]
				0000_0011	I/O-Pad	PA[5]
				0000_0100	I/O-Pad	PA[4]
				0000_0101	I/O-Pad	PA[8]
				0000_0110	—	Reserved
				0000_0111	ctu_0	EXT_TGR2
				0000_1000	I/O-Pad	PF[11]
				0000_1001	I/O-Pad	PC[2]
				0000_1010	I/O-Pad	PH[2]
				0000_1011-1111_1111	—	Reserved

Table 13. Peripheral multiplexing table description (continued)

Destination			MSCR Register	Alternate Functions	Source	
Instance	Peripherals	Functions			Peripherals	Functions
etimer_0	ETIMER	ETC2	MSCR[690]	0000_0000	—	disable-low
				0000_0001	I/O-Pad	PB[5]
				0000_0010	I/O-Pad	PC[14]
				0000_0011	I/O-Pad	PA[6]
				0000_0100	I/O-Pad	PA[7]
				0000_0101	I/O-Pad	PA[9]
				0000_0110	—	Reserved
				0000_0111	ctu_1	EXT_TGR1
				0000_1000	I/O-Pad	PF[12]
				0000_1001	I/O-Pad	PC[3]
				0000_1010	I/O-Pad	PC[7]
				0000_1011	I/O-Pad	PH[3]
				0000_1100-1111_1111	—	Reserved
etimer_0	ETIMER	ETC3	MSCR[691]	0000_0000	—	disable-low
				0000_0001	I/O-Pad	PB[6]
				0000_0010	I/O-Pad	PD[1]
				0000_0011	—	Reserved
				0000_0100	I/O-Pad	PA[11]
				0000_0101	I/O-Pad	PA[12]
				0000_0110	—	Reserved
				0000_0111	ctu_1	EXT_TGR2
				0000_1000	I/O-Pad	PF[13]
				0000_1001	I/O-Pad	PC[4]
				0000_1010-1111_1111	—	Reserved

Table 13. Peripheral multiplexing table description (continued)

Destination			MSCR Register	Alternate Functions	Source	
Instance	Peripherals	Functions			Peripherals	Functions
etimer_0	ETIMER	ETC4	MSCR[692]	0000_0000	—	disable-low
				0000_0001	I/O-Pad	PB[2]
				0000_0010	I/O-Pad	PA[14]
				0000_0011	—	Reserved
				0000_0100	I/O-Pad	PB[7]
				0000_0101	I/O-Pad	PA[13]
				0000_0110	I/O-Pad	PB[11]
				0000_0111-1111_1111	—	Reserved
etimer_0	ETIMER	ETC5	MSCR[693]	0000_0000	—	disable-low
				0000_0001	I/O-Pad	PB[8]
				0000_0010	I/O-Pad	PC[2]
				0000_0011	—	Reserved
				0000_0100	I/O-Pad	PB[10]
				0000_0101	I/O-Pad	PA[14]
				0000_0110-0000_0111	—	Reserved
				0000_1000	I/O-Pad	PC[4]
				0000_1001	I/O-Pad	PH[4]
				0000_1010-1111_1111	—	Reserved

Table 13. Peripheral multiplexing table description (continued)

Destination			MSCR Register	Alternate Functions	Source	
Instance	Peripherals	Functions			Peripherals	Functions
etimer_1	ETIMER	ETC0	MSCR[694]	0000_0000	—	disable-low
				0000_0001	I/O-Pad	PC[0]
				0000_0010	I/O-Pad	PC[3]
				0000_0011	I/O-Pad	PA[15]
				0000_0100	I/O-Pad	PB[3]
				0000_0101	I/O-Pad	PA[1]
				0000_0110	—	Reserved
				0000_0111	ctu_0	EXT_TGR1
				0000_1000	—	Reserved
				0000_1001	I/O-Pad	PF[2]
				0000_1010	I/O-Pad	PC[6]
				0000_1011	I/O-Pad	PH[3]
				0000_1100-1111_1111	—	Reserved
etimer_1	ETIMER	ETC1	MSCR[695]	0000_0000	—	disable-low
				0000_0001	I/O-Pad	PC[4]
				0000_0010	—	Reserved
				0000_0011	I/O-Pad	PB[0]
				0000_0100	I/O-Pad	PB[4]
				0000_0101	I/O-Pad	PA[2]
				0000_0110	—	Reserved
				0000_0111	ctu_0	EXT_TGR2
				0000_1000	—	Reserved
				0000_1001	I/O-Pad	PA[7]
				0000_1010-1111_1111	—	Reserved

Table 13. Peripheral multiplexing table description (continued)

Destination			MSCR Register	Alternate Functions	Source	
Instance	Peripherals	Functions			Peripherals	Functions
etimer_1	ETIMER	ETC2	MSCR[696]	0000_0000	—	disable-low
				0000_0001	I/O-Pad	PC[7]
				0000_0010	—	Reserved
				0000_0011	I/O-Pad	PC[5]
				0000_0100	I/O-Pad	PB[5]
				0000_0101	I/O-Pad	PA[5]
				0000_0110	—	Reserved
				0000_0111	ctu_1	EXT_TGR1
				0000_1000	I/O-Pad	PH[6]
				0000_1001-1111_1111	—	Reserved
etimer_1	ETIMER	ETC3	MSCR[697]	0000_0000	—	disable-low
				0000_0001	I/O-Pad	PC[8]
				0000_0010	—	Reserved
				0000_0011	I/O-Pad	PC[6]
				0000_0100	I/O-Pad	PB[6]
				0000_0101	I/O-Pad	PA[6]
				0000_0110	—	Reserved
				0000_0111	ctu_1	EXT_TGR2
				0000_1000	I/O-Pad	PH[7]
				0000_1001-1111_1111	—	Reserved

Table 13. Peripheral multiplexing table description (continued)

Destination			MSCR Register	Alternate Functions	Source	
Instance	Peripherals	Functions			Peripherals	Functions
etimer_1	ETIMER	ETC4	MSCR[698]	0000_0000	—	disable-low
				0000_0001	I/O-Pad	PC[11]
				0000_0010	—	Reserved
				0000_0011	I/O-Pad	PC[9]
				0000_0100	I/O-Pad	PB[11]
				0000_0101	I/O-Pad	PA[15]
				0000_0110	I/O-Pad	PB[7]
				0000_0111	—	Reserved
				0000_1000	I/O-Pad	PF[10]
				0000_1001-1111_1111	—	Reserved
etimer_1	ETIMER	ETC5	MSCR[699]	0000_0000	—	disable-low
				0000_0001	I/O-Pad	PB[1]
				0000_0010	—	Reserved
				0000_0011	I/O-Pad	PC[10]
				0000_0100	I/O-Pad	PB[12]
				0000_0101	I/O-Pad	PB[0]
				0000_0110	I/O-Pad	PB[10]
				0000_0111	—	Reserved
				0000_1000	I/O-Pad	PF[12]
				0000_1001-1111_1111	—	Reserved
etimer_2	ETIMER	ETC0	MSCR[700]	0000_0000	—	disable-low
				0000_0001	I/O-Pad	PB[14]
				0000_0010	I/O-Pad	PD[2]
				0000_0011-0000_0101	—	Reserved
				0000_0110	I/O-Pad	PC[15]
				0000_0111	ctu_0	EXT_TGR1
				0000_1000	I/O-Pad	PH[9]
				0000_1001-1111_1111	—	Reserved

Table 13. Peripheral multiplexing table description (continued)

Destination			MSCR Register	Alternate Functions	Source	
Instance	Peripherals	Functions			Peripherals	Functions
etimer_2	ETIMER	ETC1	MSCR[701]	0000_0000	—	disable-low
				0000_0001	I/O-Pad	PB[15]
				0000_0010	I/O-Pad	PD[3]
				0000_0011	I/O-Pad	PD[5]
				0000_0100	—	Reserved
				0000_0101	I/O-Pad	PB[14]
				0000_0110	I/O-Pad	PD[0]
				0000_0111	ctu_0	EXT_TGR2
				0000_1000	I/O-Pad	PH[12]
				0000_1001-1111_1111	—	Reserved
etimer_2	ETIMER	ETC2	MSCR[702]	0000_0000	—	disable-low
				0000_0001	I/O-Pad	PC[1]
				0000_0010	I/O-Pad	PD[4]
				0000_0011	I/O-Pad	PD[6]
				0000_0100	—	Reserved
				0000_0101	I/O-Pad	PB[15]
				0000_0110	I/O-Pad	PD[9]
				0000_0111	ctu_1	EXT_TGR1
				0000_1000-1111_1111	—	Reserved

Table 13. Peripheral multiplexing table description (continued)

Destination			MSCR Register	Alternate Functions	Source	
Instance	Peripherals	Functions			Peripherals	Functions
etimer_2	ETIMER	ETC3	MSCR[703]	0000_0000	—	disable-low
				0000_0001	I/O-Pad	PB[9]
				0000_0010	I/O-Pad	PD[14]
				0000_0011	I/O-Pad	PD[12]
				0000_0100-0000_0101	—	Reserved
				0000_0110	I/O-Pad	PD[10]
				0000_0111	ctu_1	EXT_TGR2
				0000_1000	I/O-Pad	PH[5]
				0000_1001	I/O-Pad	PH[9]
				0000_1010	I/O-Pad	PD[1]
				0000_1011	I/O-Pad	PF[13]
				0000_1100-1111_1111	—	Reserved
etimer_2	ETIMER	ETC4	MSCR[704]	0000_0000	—	disable-low
				0000_0001	I/O-Pad	PB[12]
				0000_0010	I/O-Pad	PD[15]
				0000_0011	I/O-Pad	PD[13]
				0000_0100	I/O-Pad	PE[4]
				0000_0101	I/O-Pad	PC[1]
				0000_0110	I/O-Pad	PE[0]
				0000_0111	—	Reserved
				0000_1000	I/O-Pad	PH[14]
				0000_1001	I/O-Pad	PF[8]
				0000_1010-1111_1111	—	Reserved

Table 13. Peripheral multiplexing table description (continued)

Destination			MSCR Register	Alternate Functions	Source	
Instance	Peripherals	Functions			Peripherals	Functions
etimer_2	ETIMER	ETC5	MSCR[705]	0000_0000	—	disable-low
				0000_0001	I/O-Pad	PB[13]
				0000_0010	I/O-Pad	PE[2]
				0000_0011-0000_0101	—	Reserved
				0000_0110	I/O-Pad	PE[1]
				0000_0111-1111_1111	—	Reserved
etimer_3	ETIMER	ETC0	MSCR[706]	0000_0000	—	disable-low
				0000_0001	I/O-Pad	PD[7]
				0000_0010-0000_0011	—	Reserved
				0000_0100	I/O-Pad	PB[13]
				0000_0101	—	Reserved
				0000_0110	I/O-Pad	PD[2]
				0000_0111	ctu_0	EXT_TGR1
				0000_1000	I/O-Pad	PD[13]
				0000_1001	I/O-Pad	PI[0]
				0000_1010-1111_1111	—	Reserved
etimer_3	ETIMER	ETC1	MSCR[707]	0000_0000	—	disable-low
				0000_0001	I/O-Pad	PD[8]
				0000_0010-0000_0011	—	Reserved
				0000_0100	I/O-Pad	PB[14]
				0000_0101	—	Reserved
				0000_0110	I/O-Pad	PD[3]
				0000_0111	ctu_0	EXT_TGR2
				0000_1000-1111_1111	—	Reserved

Table 13. Peripheral multiplexing table description (continued)

Destination			MSCR Register	Alternate Functions	Source	
Instance	Peripherals	Functions			Peripherals	Functions
etimer_3	ETIMER	ETC2	MSCR[708]	0000_0000	—	disable-low
				0000_0001	I/O-Pad	PD[11]
				0000_0010	—	Reserved
				0000_0011	I/O-Pad	PE[9]
				0000_0100	I/O-Pad	PB[15]
				0000_0101	—	Reserved
				0000_0110	I/O-Pad	PD[4]
				0000_0111	ctu_1	EXT_TGR1
				0000_1000	I/O-Pad	PH[13]
				0000_1001-1111_1111	—	Reserved
etimer_3	ETIMER	ETC3	MSCR[709]	0000_0000	—	disable-low
				0000_0001	I/O-Pad	PE[0]
				0000_0010	—	Reserved
				0000_0011	I/O-Pad	PE[10]
				0000_0100	I/O-Pad	PC[0]
				0000_0101	I/O-Pad	PD[14]
				0000_0110	I/O-Pad	PD[5]
				0000_0111	ctu_1	EXT_TGR2
				0000_1000	I/O-Pad	PA[11]
				0000_1001	I/O-Pad	PF[9]
				0000_1010-1111_1111	—	Reserved

Table 13. Peripheral multiplexing table description (continued)

Destination			MSCR Register	Alternate Functions	Source	
Instance	Peripherals	Functions			Peripherals	Functions
etimer_3	ETIMER	ETC4	MSCR[710]	0000_0000	—	disable-low
				0000_0001	I/O-Pad	PE[1]
				0000_0010	—	Reserved
				0000_0011	I/O-Pad	PE[12]
				0000_0100	I/O-Pad	PC[1]
				0000_0101	I/O-Pad	PD[15]
				0000_0110	I/O-Pad	PE[7]
				0000_0111	—	Reserved
				0000_1000	I/O-Pad	PF[4]
				0000_1001	I/O-Pad	PH[11]
				0000_1010-1111_1111	—	Reserved
etimer_3	ETIMER	ETC5	MSCR[711]	0000_0000	—	disable-low
				0000_0001	I/O-Pad	PE[4]
				0000_0010	—	Reserved
				0000_0011	I/O-Pad	PE[13]
				0000_0100-0000_0101	—	Reserved
				0000_0110	I/O-Pad	PE[8]
				0000_0111	—	Reserved
				0000_1000	I/O-Pad	PF[7]
				0000_1001	I/O-Pad	PF[8]
				0000_1010-1111_1111	—	Reserved
etimer_0	ETIMER	AUX0	MSCR[712]	0000_0000	—	disable-low
				0000_0001	ctu_0	EXT_TGR1
				0000_0010	ctu_0	EXT_TGR2
				0000_0011	ctu_1	EXT_TGR1
				0000_0100	ctu_1	EXT_TGR2
				0000_0101	etimer_1	ETC4
				0000_0110-1111_1111	—	Reserved

Table 13. Peripheral multiplexing table description (continued)

Destination			MSCR Register	Alternate Functions	Source	
Instance	Peripherals	Functions			Peripherals	Functions
etimer_0	ETIMER	AUX1	MSCR[713]	0000_0000	—	disable-low
				0000_0001	ctu_0	EXT_TGR1
				0000_0010	ctu_0	EXT_TGR2
				0000_0011	ctu_1	EXT_TGR1
				0000_0100	ctu_1	EXT_TGR2
				0000_0101	etimer_1	ETC2
				0000_0110-1111_1111	—	Reserved
etimer_1	ETIMER	AUX0	MSCR[714]	0000_0000	—	disable-low
				0000_0001	ctu_0	EXT_TGR1
				0000_0010	ctu_0	EXT_TGR2
				0000_0011	ctu_1	EXT_TGR1
				0000_0100	ctu_1	EXT_TGR2
				0000_0101	etimer_0	ETC4
				0000_0110-1111_1111	—	Reserved
etimer_1	ETIMER	AUX1	MSCR[715]	0000_0000	—	disable-low
				0000_0001	ctu_0	EXT_TGR1
				0000_0010	ctu_0	EXT_TGR2
				0000_0011	ctu_1	EXT_TGR1
				0000_0100	ctu_1	EXT_TGR2
				0000_0101	etimer_0	ETC2
				0000_0110-1111_1111	—	Reserved
etimer_2	ETIMER	AUX0	MSCR[716]	0000_0000	—	disable-low
				0000_0001	ctu_1	EXT_TGR1
				0000_0010	ctu_1	EXT_TGR2
				0000_0011	ctu_0	EXT_TGR1
				0000_0100	ctu_0	EXT_TGR2
				0000_0101	etimer_3	ETC4
				0000_0110-1111_1111	—	Reserved

Table 13. Peripheral multiplexing table description (continued)

Destination			MSCR Register	Alternate Functions	Source	
Instance	Peripherals	Functions			Peripherals	Functions
etimer_2	ETIMER	AUX1	MSCR[717]	0000_0000	—	disable-low
				0000_0001	ctu_1	EXT_TGR1
				0000_0010	ctu_1	EXT_TGR2
				0000_0011	ctu_0	EXT_TGR1
				0000_0100	ctu_0	EXT_TGR2
				0000_0101	etimer_3	ETC2
				0000_0110-1111_1111	—	Reserved
etimer_3	ETIMER	AUX0	MSCR[718]	0000_0000	—	disable-low
				0000_0001	ctu_1	EXT_TGR1
				0000_0010	ctu_1	EXT_TGR2
				0000_0011	ctu_0	EXT_TGR1
				0000_0100	ctu_0	EXT_TGR2
				0000_0101	etimer_2	ETC4
				0000_0110-1111_1111	—	Reserved
etimer_3	ETIMER	AUX1	MSCR[719]	0000_0000	—	disable-low
				0000_0001	ctu_1	EXT_TGR1
				0000_0010	ctu_1	EXT_TGR2
				0000_0011	ctu_0	EXT_TGR1
				0000_0100	ctu_0	EXT_TGR2
				0000_0101	etimer_2	ETC2
				0000_0110-1111_1111	—	Reserved
ctu_0	CTU	EXT_IN0	MSCR[752]	0000_0000	—	disable-low
				0000_0001	etimer_0	ETC0
				0000_0010	etimer_1	ETC0
				0000_0011	etimer_2	ETC0
				0000_0100	etimer_3	ETC0
				0000_0101	pwm_0	PWM_RLD
				0000_0110-1111_1111	—	Reserved

Table 13. Peripheral multiplexing table description (continued)

Destination			MSCR Register	Alternate Functions	Source	
Instance	Peripherals	Functions			Peripherals	Functions
ctu_0	CTU	EXT_IN1	MSCR[753]	0000_0000	—	disable-low
				0000_0001	etimer_0	ETC1
				0000_0010	etimer_1	ETC1
				0000_0011	etimer_2	ETC1
				0000_0100	etimer_3	ETC1
				0000_0101	pwm_0	OUT0_ TRIG0
				0000_0110-1111_1111	—	Reserved
ctu_0	CTU	EXT_IN2	MSCR[754]	0000_0000	—	disable-low
				0000_0001	etimer_0	ETC2
				0000_0010	etimer_1	ETC2
				0000_0011	etimer_2	ETC2
				0000_0100	etimer_3	ETC2
				0000_0101	pwm_0	OUT0_ TRIG1
				0000_0110-1111_1111	—	Reserved
ctu_0	CTU	EXT_IN3	MSCR[755]	0000_0000	—	disable-low
				0000_0001	etimer_0	ETC3
				0000_0010	etimer_1	ETC3
				0000_0011	etimer_2	ETC3
				0000_0100	etimer_3	ETC3
				0000_0101	pwm_0	OUT0_ TRIG2
				0000_0110-1111_1111	—	Reserved

Table 13. Peripheral multiplexing table description (continued)

Destination			MSCR Register	Alternate Functions	Source	
Instance	Peripherals	Functions			Peripherals	Functions
ctu_0	CTU	EXT_IN4	MSCR[756]	0000_0000	—	disable-low
				0000_0001	etimer_0	ETC4
				0000_0010	etimer_1	ETC4
				0000_0011	etimer_2	ETC4
				0000_0100	etimer_3	ETC4
				0000_0101	pwm_0	OUT0_ TRIG3
				0000_0110-1111_1111	—	Reserved
ctu_0	CTU	EXT_IN5	MSCR[757]	0000_0000	—	disable-low
				0000_0001	etimer_0	ETC5
				0000_0010	etimer_1	ETC5
				0000_0011	etimer_2	ETC5
				0000_0100	etimer_3	ETC5
				0000_0101	pwm_0	OUT1_ TRIG0
				0000_0110-1111_1111	—	Reserved
ctu_0	CTU	EXT_IN6	MSCR[758]	0000_0000	—	disable-low
				0000_0001	etimer_0	ETC0
				0000_0010	etimer_1	ETC0
				0000_0011	etimer_2	ETC0
				0000_0100	etimer_3	ETC0
				0000_0101	pwm_0	OUT1_ TRIG1
				0000_0110-1111_1111	—	Reserved

Table 13. Peripheral multiplexing table description (continued)

Destination			MSCR Register	Alternate Functions	Source	
Instance	Peripherals	Functions			Peripherals	Functions
ctu_0	CTU	EXT_IN7	MSCR[759]	0000_0000	—	disable-low
				0000_0001	etimer_0	ETC1
				0000_0010	etimer_1	ETC1
				0000_0011	etimer_2	ETC1
				0000_0100	etimer_3	ETC1
				0000_0101	pwm_0	OUT1_ TRIG2
				0000_0110-1111_1111	—	Reserved
ctu_0	CTU	EXT_IN8	MSCR[760]	0000_0000	—	disable-low
				0000_0001	etimer_0	ETC2
				0000_0010	etimer_1	ETC2
				0000_0011	etimer_2	ETC2
				0000_0100	etimer_3	ETC2
				0000_0101	pwm_0	OUT1_ TRIG3
				0000_0110-1111_1111	—	Reserved
ctu_0	CTU	EXT_IN9	MSCR[761]	0000_0000	—	disable-low
				0000_0001	etimer_0	ETC3
				0000_0010	etimer_1	ETC3
				0000_0011	etimer_2	ETC3
				0000_0100	etimer_3	ETC3
				0000_0101	pwm_0	X0
				0000_0110-1111_1111	—	Reserved

Table 13. Peripheral multiplexing table description (continued)

Destination			MSCR Register	Alternate Functions	Source	
Instance	Peripherals	Functions			Peripherals	Functions
ctu_0	CTU	EXT_IN10	MSCR[762]	0000_0000	—	disable-low
				0000_0001	etimer_0	ETC4
				0000_0010	etimer_1	ETC4
				0000_0011	etimer_2	ETC4
				0000_0100	etimer_3	ETC4
				0000_0101	pwm_0	X1
				0000_0110-1111_1111	—	Reserved
ctu_0	CTU	EXT_IN11	MSCR[763]	0000_0000	—	disable-low
				0000_0001	etimer_0	ETC5
				0000_0010	etimer_1	ETC5
				0000_0011	etimer_2	ETC5
				0000_0100	etimer_3	ETC5
				0000_0101	pwm_0	X2
				0000_0110-1111_1111	—	Reserved
ctu_0	CTU	EXT_IN12	MSCR[764]	0000_0000	—	disable-low
				0000_0001-0000_0100	—	Reserved
				0000_0101	pwm_0	X3
				0000_0110-1111_1111	—	Reserved
ctu_0	CTU	EXT_IN13	MSCR[765]	0000_0000	—	disable-low
				0000_0001	etimer_0	ETC2
				0000_0010	etimer_1	ETC4
				0000_0011	etimer_2	ETC2
				0000_0100	etimer_2	ETC4
				0000_0101-1111_1111	—	Reserved

Table 13. Peripheral multiplexing table description (continued)

Destination			MSCR Register	Alternate Functions	Source	
Instance	Peripherals	Functions			Peripherals	Functions
ctu_0	CTU	EXT_IN14	MSCR[766]	0000_0000	—	disable-low
				0000_0001	etimer_0	ETC4
				0000_0010	etimer_1	ETC2
				0000_0011	etimer_3	ETC4
				0000_0100	etimer_3	ETC2
				0000_0101-1111_1111	—	Reserved
ctu_0	CTU	EXT_IN15	MSCR[767]	0000_0000	—	disable-low
				0000_0001	—	Reserved
				0000_0010	I/O-Pad	PD[7]
				0000_0011	I/O-Pad	PB[10]
				0000_0100-1111_1111	—	Reserved
ctu_1	CTU	EXT_IN0	MSCR[768]	0000_0000	—	disable-low
				0000_0001	etimer_0	ETC0
				0000_0010	etimer_1	ETC0
				0000_0011	etimer_2	ETC0
				0000_0100	etimer_3	ETC0
				0000_0101	pwm_2	PWM_RLD
				0000_0110-1111_1111	—	Reserved
ctu_1	CTU	EXT_IN1	MSCR[769]	0000_0000	—	disable-low
				0000_0001	etimer_0	ETC1
				0000_0010	etimer_1	ETC1
				0000_0011	etimer_2	ETC1
				0000_0100	etimer_3	ETC1
				0000_0101	pwm_2	OUT0_TRIG0
				0000_0110-1111_1111	—	Reserved

Table 13. Peripheral multiplexing table description (continued)

Destination			MSCR Register	Alternate Functions	Source	
Instance	Peripherals	Functions			Peripherals	Functions
ctu_1	CTU	EXT_IN2	MSCR[770]	0000_0000	—	disable-low
				0000_0001	etimer_0	ETC2
				0000_0010	etimer_1	ETC2
				0000_0011	etimer_2	ETC2
				0000_0100	etimer_3	ETC2
				0000_0101	pwm_2	OUT0_ TRIG1
				0000_0110-1111_1111	—	Reserved
ctu_1	CTU	EXT_IN3	MSCR[771]	0000_0000	—	disable-low
				0000_0001	etimer_0	ETC3
				0000_0010	etimer_1	ETC3
				0000_0011	etimer_2	ETC3
				0000_0100	etimer_3	ETC3
				0000_0101	pwm_2	OUT0_ TRIG2
				0000_0110-1111_1111	—	Reserved
ctu_1	CTU	EXT_IN4	MSCR[772]	0000_0000	—	disable-low
				0000_0001	etimer_0	ETC4
				0000_0010	etimer_1	ETC4
				0000_0011	etimer_2	ETC4
				0000_0100	etimer_3	ETC4
				0000_0101	pwm_2	OUT0_ TRIG3
				0000_0110-1111_1111	—	Reserved

Table 13. Peripheral multiplexing table description (continued)

Destination			MSCR Register	Alternate Functions	Source	
Instance	Peripherals	Functions			Peripherals	Functions
ctu_1	CTU	EXT_IN5	MSCR[773]	0000_0000	—	disable-low
				0000_0001	etimer_0	ETC5
				0000_0010	etimer_1	ETC5
				0000_0011	etimer_2	ETC5
				0000_0100	etimer_3	ETC5
				0000_0101	pwm_2	OUT1_ TRIG0
				0000_0110-1111_1111	—	Reserved
ctu_1	CTU	EXT_IN6	MSCR[774]	0000_0000	—	disable-low
				0000_0001	etimer_0	ETC0
				0000_0010	etimer_1	ETC0
				0000_0011	etimer_2	ETC0
				0000_0100	etimer_3	ETC0
				0000_0101	pwm_2	OUT1_ TRIG1
				0000_0110-1111_1111	—	Reserved
ctu_1	CTU	EXT_IN7	MSCR[775]	0000_0000	—	disable-low
				0000_0001	etimer_0	ETC1
				0000_0010	etimer_1	ETC1
				0000_0011	etimer_2	ETC1
				0000_0100	etimer_3	ETC1
				0000_0101	pwm_2	OUT1_ TRIG2
				0000_0110-1111_1111	—	Reserved

Table 13. Peripheral multiplexing table description (continued)

Destination			MSCR Register	Alternate Functions	Source	
Instance	Peripherals	Functions			Peripherals	Functions
ctu_1	CTU	EXT_IN8	MSCR[776]	0000_0000	—	disable-low
				0000_0001	etimer_0	ETC2
				0000_0010	etimer_1	ETC2
				0000_0011	etimer_2	ETC2
				0000_0100	etimer_3	ETC2
				0000_0101	pwm_2	OUT1_ TRIG3
				0000_0110-1111_1111	—	Reserved
ctu_1	CTU	EXT_IN9	MSCR[777]	0000_0000	—	disable-low
				0000_0001	etimer_0	ETC3
				0000_0010	etimer_1	ETC3
				0000_0011	etimer_2	ETC3
				0000_0100	etimer_3	ETC3
				0000_0101	pwm_2	X0
				0000_0110-1111_1111	—	Reserved
ctu_1	CTU	EXT_IN10	MSCR[778]	0000_0000	—	disable-low
				0000_0001	etimer_0	ETC4
				0000_0010	etimer_1	ETC4
				0000_0011	etimer_2	ETC4
				0000_0100	etimer_3	ETC4
				0000_0101	pwm_2	X1
				0000_0110-1111_1111	—	Reserved
ctu_1	CTU	EXT_IN11	MSCR[779]	0000_0000	—	disable-low
				0000_0001	etimer_0	ETC5
				0000_0010	etimer_1	ETC5
				0000_0011	etimer_2	ETC5
				0000_0100	etimer_3	ETC5
				0000_0101	pwm_2	X2
				0000_0110-1111_1111	—	Reserved

Table 13. Peripheral multiplexing table description (continued)

Destination			MSCR Register	Alternate Functions	Source	
Instance	Peripherals	Functions			Peripherals	Functions
ctu_1	CTU	EXT_IN12	MSCR[780]	0000_0000	—	disable-low
				0000_0001-0000_0100	—	Reserved
				0000_0101	pwm_2	X3
				0000_0110-1111_1111	—	Reserved
ctu_1	CTU	EXT_IN13	MSCR[781]	0000_0000	—	disable-low
				0000_0001	etimer_0	ETC2
				0000_0010	etimer_1	ETC4
				0000_0011	etimer_2	ETC2
				0000_0100	etimer_2	ETC4
				0000_0101-1111_1111	—	Reserved
ctu_1	CTU	EXT_IN14	MSCR[782]	0000_0000	—	disable-low
				0000_0001	etimer_0	ETC4
				0000_0010	etimer_1	ETC2
				0000_0011	etimer_3	ETC4
				0000_0100	etimer_3	ETC2
				0000_0101-1111_1111	—	Reserved
ctu_1	CTU	EXT_IN15	MSCR[783]	0000_0000	—	disable-low
				0000_0001	—	Reserved
				0000_0010	I/O-Pad	PD[7]
				0000_0011	I/O-Pad	PG[15]
				0000_0100	I/O-Pad	PE[1]
				0000_0101-1111_1111	—	Reserved
flexray_0	ESYS_ FLEXRAY_2	RXD_A	MSCR[784]	0000_0000	—	disable-low
				0000_0001-0000_0100	—	Reserved
				0000_0101	I/O-Pad	PB[0]
				0000_0110-1111_1111	—	Reserved

Table 13. Peripheral multiplexing table description (continued)

Destination			MSCR Register	Alternate Functions	Source	
Instance	Peripherals	Functions			Peripherals	Functions
flexray_0	ESYS_ FLEXRAY_2	RXD_B	MSCR[785]	0000_0000	—	disable-low
				0000_0001-0000_0100	—	Reserved
				0000_0101	I/O-Pad	PH[11]
				0000_0110-1111_1111	—	Reserved
flexray_0	ESYS_ FLEXRAY_2	OS_tick	MSCR[786]	0000_0000	—	disable-low
				0000_0001	pit_rti_0	pit_trigger[0]
				0000_0010	acp_sphaero	STM0_INT0
				0000_0011-0000_0100	—	Reserved
				0000_0101	I/O-Pad	PA[3]
				0000_0110	I/O-Pad	PB[9]
				0000_0111-1111_1111	—	Reserved
sent_0	SENT_IP_ 32BIT_TOP	SENT0	MSCR[788]	0000_0000	—	disable-low
				0000_0001-0000_0101	—	Reserved
				0000_0110	I/O-Pad	PA[0]
				0000_0111-1111_1111	—	Reserved
sent_0	SENT_IP_ 32BIT_TOP	SENT1	MSCR[789]	0000_0000	—	disable-low
				0000_0001-0000_0101	—	Reserved
				0000_0110	I/O-Pad	PA[1]
				0000_0111-1111_1111	—	Reserved
sent_0	SENT_IP_ 32BIT_TOP	SENT2	MSCR[790]	0000_0000	—	disable-low
				0000_0001-0000_0101	—	Reserved
				0000_0110	I/O-Pad	PA[2]
				0000_0111-1111_1111	—	Reserved

Table 13. Peripheral multiplexing table description (continued)

Destination			MSCR Register	Alternate Functions	Source	
Instance	Peripherals	Functions			Peripherals	Functions
sent_1	SENT_IP_ 32BIT_TOP	SENT0	MSCR[792]	0000_0000	—	disable-low
				0000_0001-0000_0101	—	Reserved
				0000_0110	I/O-Pad	PD[13]
				0000_0111-1111_1111	—	Reserved
sent_1	SENT_IP_ 32BIT_TOP	SENT1	MSCR[793]	0000_0000	—	disable-low
				0000_0001-0000_0101	—	Reserved
				0000_0110	I/O-Pad	PD[14]
				0000_0111-1111_1111	—	Reserved
sent_1	SENT_IP_ 32BIT_TOP	SENT2	MSCR[794]	0000_0000	—	disable-low
				0000_0001-0000_0101	—	Reserved
				0000_0110	I/O-Pad	PD[15]
				0000_0111-1111_1111	—	Reserved
siul	SIUL	EIRQ0	MSCR[816]	0000_0000	—	disable-low
				0000_0001	I/O-Pad	PB[7]
				0000_0010	I/O-Pad	PA[0]
				0000_0011	I/O-Pad	PD[0]
				0000_0100	I/O-Pad	PC[0]
				0000_0101-1111_1111	—	Reserved
siul	SIUL	EIRQ1	MSCR[817]	0000_0000	—	disable-low
				0000_0001	I/O-Pad	PB[10]
				0000_0010	I/O-Pad	PA[3]
				0000_0011	I/O-Pad	PC[4]
				0000_0100-1111_1111	—	Reserved

Table 13. Peripheral multiplexing table description (continued)

Destination			MSCR Register	Alternate Functions	Source	
Instance	Peripherals	Functions			Peripherals	Functions
siul	SIUL	EIRQ2	MSCR[818]	0000_0000	—	disable-low
				0000_0001	I/O-Pad	PB[11]
				0000_0010	I/O-Pad	PE[0]
				0000_0011-1111_1111	—	Reserved
siul	SIUL	EIRQ3	MSCR[819]	0000_0000	—	disable-low
				0000_0001	—	Reserved
				0000_0010	I/O-Pad	PA[14]
				0000_0011	I/O-Pad	PB[2]
				0000_0100	I/O-Pad	PE[8]
				0000_0101	I/O-Pad	PA[10]
				0000_0110	I/O-Pad	PF[7]
				0000_0111-1111_1111	—	Reserved
siul	SIUL	EIRQ4	MSCR[820]	0000_0000	—	disable-low
				0000_0001	—	Reserved
				0000_0010	I/O-Pad	PB[8]
				0000_0011	I/O-Pad	PC[15]
				0000_0100	I/O-Pad	PE[9]
				0000_0101	I/O-Pad	PA[11]
				0000_0110-1111_1111	—	Reserved
siul	SIUL	EIRQ5	MSCR[821]	0000_0000	—	disable-low
				0000_0001	—	Reserved
				0000_0010	I/O-Pad	PB[9]
				0000_0011	—	Reserved
				0000_0100	I/O-Pad	PE[10]
				0000_0101-1111_1111	—	Reserved

Table 13. Peripheral multiplexing table description (continued)

Destination			MSCR Register	Alternate Functions	Source	
Instance	Peripherals	Functions			Peripherals	Functions
wkpufilter_ana	I2CFIL_M55_A1	NMI	MSCR[944]	0000_0000	—	disable-high
				0000_0001	I/O-Pad	PC[15]
				0000_0010	I/O-Pad	PA[4]
				0000_0011	I/O-Pad	PB[13]
				0000_0100	—	Reserved
				0000_0101	I/O-Pad	PA[5]
				0000_0110-1111_1111	—	Reserved
fccu	FCCU_TOP	F1	MSCR[948]	0000_0000	—	disable-high
				0000_0001	I/O-Pad	PAD_FCCU_F1S
				0000_0010	I/O-Pad	PAD_FCCU_F1E
				0000_0011-1111_1111	—	Reserved
pwm_0	FLEXPWM	A0	MSCR[952]	0000_0000	—	disable-low
				0000_0001-0000_0100	—	Reserved
				0000_0101	I/O-Pad	PA[6]
				0000_0110-1111_1111	—	Reserved
pwm_0	FLEXPWM	A1	MSCR[953]	0000_0000	—	disable-low
				0000_0001-0000_0100	—	Reserved
				0000_0101	I/O-Pad	PF[3]
				0000_0110	—	Reserved
				0000_0111	I/O-Pad	PE[5]
				0000_1000-1111_1111	—	Reserved

Table 13. Peripheral multiplexing table description (continued)

Destination			MSCR Register	Alternate Functions	Source	
Instance	Peripherals	Functions			Peripherals	Functions
pwm_0	FLEXPWM	A2	MSCR[954]	0000_0000	—	disable-low
				0000_0001-0000_0100	—	Reserved
				0000_0101	I/O-Pad	PF[4]
				0000_0110	—	Reserved
				0000_0111	I/O-Pad	PE[6]
				0000_1000-1111_1111	—	Reserved
pwm_0	FLEXPWM	A3	MSCR[955]	0000_0000	—	disable-low
				0000_0001-0000_0100	—	Reserved
				0000_0101	I/O-Pad	PA[10]
				0000_0110	—	Reserved
				0000_0111	I/O-Pad	PE[7]
				0000_1000-1111_1111	—	Reserved
pwm_0	FLEXPWM	B0	MSCR[956]	0000_0000	—	disable-low
				0000_0001-0000_0100	—	Reserved
				0000_0101	I/O-Pad	PA[7]
				0000_0110-1111_1111	—	Reserved
pwm_0	FLEXPWM	B1	MSCR[957]	0000_0000	—	disable-low
				0000_0001-0000_0100	—	Reserved
				0000_0101	I/O-Pad	PA[11]
				0000_0110	—	Reserved
				0000_0111	I/O-Pad	PE[8]
				0000_1000-1111_1111	—	Reserved

Table 13. Peripheral multiplexing table description (continued)

Destination			MSCR Register	Alternate Functions	Source	
Instance	Peripherals	Functions			Peripherals	Functions
pwm_0	FLEXPWM	B2	MSCR[958]	0000_0000	—	disable-low
				0000_0001-0000_0100	—	Reserved
				0000_0101	I/O-Pad	PF[5]
				0000_0110	I/O-Pad	PF[6]
				0000_0111	I/O-Pad	PE[9]
				0000_1000-1111_1111	—	Reserved
pwm_0	FLEXPWM	B3	MSCR[959]	0000_0000	—	disable-low
				0000_0001-0000_0100	—	Reserved
				0000_0101	I/O-Pad	PA[12]
				0000_0110	I/O-Pad	PA[13]
				0000_0111	I/O-Pad	PE[10]
				0000_1000-1111_1111	—	Reserved
pwm_1	FLEXPWM	A0	MSCR[960]	0000_0000	—	disable-low
				0000_0001-0000_0100	—	Reserved
				0000_0101	I/O-Pad	PA[10]
				0000_0110-1111_1111	—	Reserved
pwm_1	FLEXPWM	A1	MSCR[961]	0000_0000	—	disable-low
				0000_0001-0000_0100	—	Reserved
				0000_0101	I/O-Pad	PC[12]
				0000_0110-1111_1111	—	Reserved
pwm_1	FLEXPWM	B0	MSCR[962]	0000_0000	—	disable-low
				0000_0001-0000_0100	—	Reserved
				0000_0101	I/O-Pad	PA[12]
				0000_0110-1111_1111	—	Reserved

Table 13. Peripheral multiplexing table description (continued)

Destination			MSCR Register	Alternate Functions	Source	
Instance	Peripherals	Functions			Peripherals	Functions
pwm_1	FLEXPWM	B1	MSCR[963]	0000_0000	—	disable-low
				0000_0001-0000_0100	—	Reserved
				0000_0101	I/O-Pad	PA[13]
				0000_0110-1111_1111	—	Reserved
pwm_2	FLEXPWM	A0	MSCR[964]	0000_0000	—	disable-low
				0000_0001-0000_0100	—	Reserved
				0000_0101	I/O-Pad	PD[9]
				0000_0110-1111_1111	—	Reserved
pwm_2	FLEXPWM	A1	MSCR[965]	0000_0000	—	disable-low
				0000_0001-0000_0100	—	Reserved
				0000_0101	I/O-Pad	PD[5]
				0000_0110-1111_1111	—	Reserved
pwm_2	FLEXPWM	A2	MSCR[966]	0000_0000	—	disable-low
				0000_0001-0000_0100	—	Reserved
				0000_0101	I/O-Pad	PD[6]
				0000_0110-1111_1111	—	Reserved
pwm_2	FLEXPWM	A3	MSCR[967]	0000_0000	—	disable-low
				0000_0001-0000_0100	—	Reserved
				0000_0101	I/O-Pad	PD[7]
				0000_0110-1111_1111	—	Reserved

Table 13. Peripheral multiplexing table description (continued)

Destination			MSCR Register	Alternate Functions	Source	
Instance	Peripherals	Functions			Peripherals	Functions
pwm_2	FLEXPWM	B0	MSCR[968]	0000_0000	—	disable-low
				0000_0001-0000_0100	—	Reserved
				0000_0101	I/O-Pad	PD[10]
				0000_0110-1111_1111	—	Reserved
pwm_2	FLEXPWM	B1	MSCR[969]	0000_0000	—	disable-low
				0000_0001-0000_0100	—	Reserved
				0000_0101	I/O-Pad	PD[8]
				0000_0110	—	Reserved
				0000_0111	I/O-Pad	PE[13]
				0000_1000-1111_1111	—	Reserved
pwm_2	FLEXPWM	B2	MSCR[970]	0000_0000	—	disable-low
				0000_0001-0000_0100	—	Reserved
				0000_0101	I/O-Pad	PH[6]
				0000_0110	—	Reserved
				0000_0111	I/O-Pad	PE[14]
				0000_1000-1111_1111	—	Reserved
pwm_2	FLEXPWM	B3	MSCR[971]	0000_0000	—	disable-low
				0000_0001-0000_0100	—	Reserved
				0000_0101	I/O-Pad	PH[7]
				0000_0110	I/O-Pad	PH[5]
				0000_0111	I/O-Pad	PE[15]
				0000_1000-1111_1111	—	Reserved

Table 13. Peripheral multiplexing table description (continued)

Destination			MSCR Register	Alternate Functions	Source	
Instance	Peripherals	Functions			Peripherals	Functions
pwm_3	FLEXPWM	A0	MSCR[972]	0000_0000	—	disable-low
				0000_0001-0000_0100	—	Reserved
				0000_0101	I/O-Pad	PD[5]
				0000_0110-1111_1111	—	Reserved
pwm_3	FLEXPWM	A1	MSCR[973]	0000_0000	—	disable-low
				0000_0001-0000_0100	—	Reserved
				0000_0101	I/O-Pad	PH[2]
				0000_0110	—	Reserved
				0000_0111	I/O-Pad	PE[12]
				0000_1000-1111_1111	—	Reserved
pwm_3	FLEXPWM	B0	MSCR[974]	0000_0000	—	disable-low
				0000_0001-0000_0100	—	Reserved
				0000_0101	I/O-Pad	PD[6]
				0000_0110-1111_1111	—	Reserved
pwm_3	FLEXPWM	B1	MSCR[975]	0000_0000	—	disable-low
				0000_0001-0000_0100	—	Reserved
				0000_0101	I/O-Pad	PD[11]
				0000_0110-1111_1111	—	Reserved
pwm_0	FLEXPWM	X0	MSCR[976]	0000_0000	—	disable-low
				0000_0001-0000_0100	—	Reserved
				0000_0101	I/O-Pad	PE[11]
				0000_0110-1111_1111	—	Reserved

Table 13. Peripheral multiplexing table description (continued)

Destination			MSCR Register	Alternate Functions	Source	
Instance	Peripherals	Functions			Peripherals	Functions
pwm_0	FLEXPWM	X1	MSCR[977]	0000_0000	—	disable-low
				0000_0001-0000_0100	—	Reserved
				0000_0101	I/O-Pad	PE[12]
				0000_0110-1111_1111	—	Reserved
pwm_0	FLEXPWM	X2	MSCR[978]	0000_0000	—	disable-low
				0000_0001-0000_0100	—	Reserved
				0000_0101	I/O-Pad	PE[13]
				0000_0110-1111_1111	—	Reserved
pwm_0	FLEXPWM	X3	MSCR[979]	0000_0000	—	disable-low
				0000_0001-0000_0100	—	Reserved
				0000_0101	I/O-Pad	PE[14]
				0000_0110-1111_1111	—	Reserved
pwm_0	FLEXPWM	FAULT0	MSCR[980]	0000_0000	—	disable-low
				0000_0001-0000_0100	—	Reserved
				0000_0101	I/O-Pad	PA[0]
				0000_0110	I/O-Pad	PB[11]
				0000_0111-1111_1111	—	Reserved
pwm_0	FLEXPWM	FAULT1	MSCR[981]	0000_0000	—	disable-low
				0000_0001-0000_0100	—	Reserved
				0000_0101	I/O-Pad	PA[1]
				0000_0110	I/O-Pad	PB[12]
				0000_0111-1111_1111	—	Reserved

Table 13. Peripheral multiplexing table description (continued)

Destination			MSCR Register	Alternate Functions	Source	
Instance	Peripherals	Functions			Peripherals	Functions
pwm_0	FLEXPWM	FAULT2	MSCR[982]	0000_0000	—	disable-low
				0000_0001-0000_0100	—	Reserved
				0000_0101	I/O-Pad	PA[8]
				0000_0110	I/O-Pad	PB[13]
				0000_0111-1111_1111	—	Reserved
pwm_0	FLEXPWM	FAULT3	MSCR[983]	0000_0000	—	disable-low
				0000_0001-0000_0100	—	Reserved
				0000_0101	I/O-Pad	PA[9]
				0000_0110	I/O-Pad	PB[14]
				0000_0111-1111_1111	—	Reserved
pwm_0	FLEXPWM	EXT_SYNC	MSCR[984]	0000_0000	—	disable-low
				0000_0001-0000_0100	—	Reserved
				0000_0101	I/O-Pad	PA[5]
				0000_0110	I/O-Pad	PB[6]
				0000_0111-1111_1111	—	Reserved
pwm_0	FLEXPWM	EXT_FORCE	MSCR[985]	0000_0000	—	disable-low
				0000_0001-0000_0100	—	Reserved
				0000_0101	pwm_ synchr_0	EXT_ FORCE0
				0000_0110	etimer_0	ETC1
				0000_0111	etimer_2	ETC5
				0000_1000-1111_1111	—	Reserved

Table 13. Peripheral multiplexing table description (continued)

Destination			MSCR Register	Alternate Functions	Source	
Instance	Peripherals	Functions			Peripherals	Functions
pwm_0	FLEXPWM	EXT_CLK	MSCR[986]	0000_0000	—	disable-low
				0000_0001-0000_0100	—	Reserved
				0000_0101	etimer_0	ETC1
				0000_0110	etimer_1	ETC1
				0000_0111	etimer_3	ETC5
				0000_1000-1111_1111	—	Reserved
pwm_1	FLEXPWM	EXT_FORCE	MSCR[987]	0000_0000	—	disable-low
				0000_0001-0000_0100	—	Reserved
				0000_0101	pwm_ synchr_0	EXT_ FORCE1
				0000_0110	etimer_0	ETC3
				0000_0111	etimer_2	ETC3
				0000_1000-1111_1111	—	Reserved
pwm_2	FLEXPWM	X0	MSCR[988]	0000_0000	—	disable-low
				0000_0001-0000_0100	—	Reserved
				0000_0101	I/O-Pad	PC[13]
				0000_0110-1111_1111	—	Reserved
pwm_2	FLEXPWM	X1	MSCR[989]	0000_0000	—	disable-low
				0000_0001-0000_0100	—	Reserved
				0000_0101	I/O-Pad	PC[14]
				0000_0110-1111_1111	—	Reserved
pwm_2	FLEXPWM	X2	MSCR[990]	0000_0000	—	disable-low
				0000_0001-0000_0100	—	Reserved
				0000_0101	I/O-Pad	PD[0]
				0000_0110-1111_1111	—	Reserved

Table 13. Peripheral multiplexing table description (continued)

Destination			MSCR Register	Alternate Functions	Source	
Instance	Peripherals	Functions			Peripherals	Functions
pwm_2	FLEXPWM	X3	MSCR[991]	0000_0000	—	disable-low
				0000_0001-0000_0100	—	Reserved
				0000_0101	I/O-Pad	PD[1]
				0000_0110-1111_1111	—	Reserved
pwm_2	FLEXPWM	FAULT0	MSCR[992]	0000_0000	—	disable-low
				0000_0001-0000_0100	—	Reserved
				0000_0101	I/O-Pad	PD[12]
				0000_0110	I/O-Pad	PG[2]
				0000_0111-1111_1111	—	Reserved
pwm_2	FLEXPWM	FAULT1	MSCR[993]	0000_0000	—	disable-low
				0000_0001-0000_0100	—	Reserved
				0000_0101	I/O-Pad	PD[13]
				0000_0110	I/O-Pad	PG[3]
				0000_0111-1111_1111	—	Reserved
pwm_2	FLEXPWM	FAULT2	MSCR[994]	0000_0000	—	disable-low
				0000_0001-0000_0100	—	Reserved
				0000_0101	I/O-Pad	PD[14]
				0000_0110	I/O-Pad	PG[4]
				0000_0111-1111_1111	—	Reserved
pwm_2	FLEXPWM	FAULT3	MSCR[995]	0000_0000	—	disable-low
				0000_0001-0000_0100	—	Reserved
				0000_0101	I/O-Pad	PD[15]
				0000_0110	I/O-Pad	PG[5]
				0000_0111-1111_1111	—	Reserved

Table 13. Peripheral multiplexing table description (continued)

Destination			MSCR Register	Alternate Functions	Source	
Instance	Peripherals	Functions			Peripherals	Functions
pwm_2	FLEXPWM	EXT_SYNC	MSCR[996]	0000_0000	—	disable-low
				0000_0001-0000_0100	—	Reserved
				0000_0101	I/O-Pad	PH[4]
				0000_0110	I/O-Pad	PG[6]
				0000_0111-1111_1111	—	Reserved
pwm_2	FLEXPWM	EXT_FORCE	MSCR[997]	0000_0000	—	disable-low
				0000_0001-0000_0100	—	Reserved
				0000_0101	pwm_ synchr_0	EXT_ FORCE2
				0000_0110	etimer_2	ETC1
				0000_0111	etimer_0	ETC5
				0000_1000-1111_1111	—	Reserved
pwm_2	FLEXPWM	EXT_CLK	MSCR[998]	0000_0000	—	disable-low
				0000_0001-0000_0100	—	Reserved
				0000_0101	etimer_2	ETC1
				0000_0110	etimer_3	ETC1
				0000_0111	etimer_1	ETC5
				0000_1000-1111_1111	—	Reserved
pwm_3	FLEXPWM	EXT_FORCE	MSCR[999]	0000_0000	—	disable-low
				0000_0001-0000_0100	—	Reserved
				0000_0101	pwm_ synchr_0	EXT_ FORCE3
				0000_0110	etimer_2	ETC3
				0000_0111	etimer_0	ETC3
				0000_1000-1111_1111	—	Reserved

4.2 Package pads/pins

Table 14 shows the cross-mapping between pads and eTQFP100, eTQFP144 pinouts.

Table 14. Cross-mapping between pads and package pins

PAD	Port pin name	Package type and pin	
		eTQFP 100	eTQFP 144
PAD_FCCU_F0	FCCU_F0	1	110
PAD[0]	PA[0]	2	111
PAD[1]	PA[1]	3	112
PAD[2]	PA[2]	4	113
PAD[3]	PA[3]	5	114
PAD[4]	PA[4]	6	117
PAD[5]	PA[5]	7	118
PAD[6]	PA[6]	8	119
PAD[7]	PA[7]	9	121
PAD[8]	PA[8]	10	122
PAD[9]	PA[9]	11	123
PAD[10]	PA[10]	12	127
PAD[11]	PA[11]	13	128
PAD[12]	PA[12]	14	130
PAD[13]	PA[13]	15	131
PAD[14]	PA[14]	16	132
PAD[15]	PA[15]	17	134
PAD[16]	PB[0]	18	135
PAD[17]	PB[1]	21	139
PAD[18]	PB[2]	22	—
PAD[19]	PB[3]	23	—
PAD[20]	PB[4]	24	—
PAD[21]	PB[5]	25	—
PAD[22]	PB[6]	26	2
PAD[23]	PB[7]	27	3
PAD[24]	PB[8]	28	4
PAD[25]	PB[9]	—	—
PAD[26]	PB[10]	31	8
PAD[27]	PB[11]	32	9
PAD[28]	PB[12]	33	10

Table 14. Cross-mapping between pads and package pins (continued)

PAD	Port pin name	Package type and pin	
		eTQFP 100	eTQFP 144
PAD[29]	PB[13]	34	12
PAD[30]	PB[14]	35	13
PAD[31]	PB[15]	36	14
PAD[32]	PC[0]	29	6
PAD[33]	PC[1]	30	7
PAD[34]	PC[2]	—	—
PAD[35]	PC[3]	—	—
PAD[36]	PC[4]	42	—
PAD[37]	PC[5]	43	—
PAD[38]	PC[6]	44	—
PAD[39]	PC[7]	45	—
PAD[40]	PC[8]	46	—
PAD[41]	PC[9]	47	—
PAD[42]	PC[10]	48	—
PAD[43]	PC[11]	49	—
PAD_FCCU_F1S	FCCU_F1S	50	—
PAD[44]	PC[12]	56	44
PAD_FCCU_F1E	FCCU_F1E	—	45
PAD[45]	PC[13]	57	48
PAD[46]	PC[14]	58	49
PAD_TDI	TDI	59	51
PAD_TMS	TMS	60	53
PAD_TDO	TDO	61	54
PAD[47]	PC[15]	62	55
PAD_TCK	TCK	63	57
PAD_JCOMP	JCOMP	—	—
PAD[48]	PD[0]	65	60
PAD[49]	PD[1]	67	62
PAD[50]	PD[2]	69	64
PAD[51]	PD[3]	70	65
PAD[52]	PD[4]	71	67
PAD[53]	PD[5]	72	68
PAD[54]	PD[6]	73	69

Table 14. Cross-mapping between pads and package pins (continued)

PAD	Port pin name	Package type and pin	
		eTQFP 100	eTQFP 144
PAD[55]	PD[7]	74	70
PAD[56]	PD[8]	75	71
PAD[57]	PD[9]	76	74
PAD[58]	PD[10]	77	75
PAD[59]	PD[11]	78	77
PAD[60]	PD[12]	79	78
PAD[61]	PD[13]	80	81
PAD[62]	PD[14]	81	82
PAD[63]	PD[15]	82	84
PAD[64]	PE[0]	83	85
PAD[65]	PE[1]	84	88
PAD[66]	PE[2]	86	91
PAD[67]	PE[3]	87	92
PAD[68]	PE[4]	88	93
PAD[69]	PE[5]	89	95
PAD[70]	PE[6]	90	96
PAD[71]	PE[7]	91	97
PAD[72]	PE[8]	92	100
PAD[73]	PE[9]	93	101
PAD[74]	PE[10]	94	102
PAD[75]	PE[11]	96	104
PAD[76]	PE[12]	97	105
PAD[77]	PE[13]	98	106
PAD[78]	PE[14]	99	107
PAD[79]	PE[15]	100	108
PAD[80]	PF[0]	—	—
PAD[81]	PF[1]	—	116
PAD[82]	PF[2]	—	120
PAD[83]	PF[3]	—	124
PAD[84]	PF[4]	—	125
PAD[85]	PF[5]	—	129
PAD[86]	PF[6]	—	133
PAD[87]	PF[7]	—	136

Table 14. Cross-mapping between pads and package pins (continued)

PAD	Port pin name	Package type and pin	
		eTQFP 100	eTQFP 144
PAD[88]	PF[8]	—	137
PAD[89]	PF[9]	—	140
PAD[90]	PF[10]	—	141
PAD[91]	PF[11]	—	—
PAD[92]	PF[12]	—	143
PAD[93]	PF[13]	—	144
PAD[94]	PF[14]	—	1
PAD[95]	PF[15]	—	5
PAD[96]	PG[0]	—	11
PAD[97]	PG[1]	—	15
PAD[98]	PG[2]	—	19
PAD[99]	PG[3]	—	20
PAD[100]	PG[4]	—	21
PAD[101]	PG[5]	—	24
PAD[102]	PG[6]	—	25
PAD[103]	PG[7]	—	26
PAD[104]	PG[8]	—	27
PAD[105]	PG[9]	—	—
PAD[106]	PG[10]	—	30
PAD[107]	PG[11]	—	31
PAD[108]	PG[12]	—	32
PAD[109]	PG[13]	—	33
PAD[110]	PG[14]	—	34
PAD[111]	PG[15]	—	35
PAD[112]	PH[0]	—	28
PAD[113]	PH[1]	—	29
PAD[114]	PH[2]	—	50
PAD[115]	PH[3]	—	52
PAD[116]	PH[4]	—	56
PAD[117]	PH[5]	—	66
PAD[118]	PH[6]	—	72
PAD[119]	PH[7]	—	73
PAD[120]	PH[8]	—	76

Table 14. Cross-mapping between pads and package pins (continued)

PAD	Port pin name	Package type and pin	
		eTQFP 100	eTQFP 144
PAD[121]	PH[9]	—	83
PAD[122]	PH[10]	—	86
PAD[123]	PH[11]	—	87
PAD[124]	PH[12]	—	89
PAD[125]	PH[13]	—	94
PAD[126]	PH[14]	—	98
PAD[127]	PH[15]	—	—
PAD[128]	PI[0]	—	109
PAD[129]	PI[1]	40	—
PAD[130]	PI[2]	41	—
PAD[131]	PI[3]	—	—

5 Memory map

5.1 Architecture

Two system memories are included in this device:

- System RAM
- Flash memory

Both memories are interfaced to the system platform through their respective RAM and Flash controllers, which in turn are connected to the XBAR slave ports.

5.2 Memory map

Table 15. Memory map (XBAR slave ports)

Start address	End address	Description
0x0000_0000	0x003F_FFFF	Reserved
0x0040_0000	0x0113_FFFF	Flash memory controller
0x0114_0000	0x3FFF_FFFF	Reserved
0x4000_0000	0x4001_7FFF	System RAM (128 MB)
0x4001_8000	0x4FFF_FFFF	Reserved
0x5000_0000	0x5000_7FFF	Processor local DMEM (32KB)
0x5000_8000	0x5000_FFFF	DMEM mirrored(1) (2)
0x5001_0000	0xF7FF_FFFF	Reserved
0xF800_0000	0xFBFF_FFFF	Peripherals bridge 1 (see *Table 17: Peripheral bridge 1 memory map*)
0xFC00_0000	0xFFFF_FFFF	Peripherals bridge 0 (see *Table 18: Peripheral bridge 0 memory map*)

1. The DMEM window is HW-wired to be 64KB, but in reality only 32KB addresses are used, creating a mirrored version of the DMEM space.
2. In order to avoid ECC errors, it is mandatory to keep a consistent access between write and read within the same DMEM space portion. In other words, if data is written into the upper DMEM window and the same “physical” address is read back from the lower (mirrored) DMEM window, an ECC error will happen because of the different “logical” address being used. The same applies for a write executed into the lower (mirrored) DMEM window, followed by the data reading at the same “physical” address within the upper DMEM window. Whenever there is an access to the DMEM mirrored space, the address feedback alarm gets set. It will be reported to FCCU as ncf [70].

Table 16. Flash memory map

Start address	End address	Description	RWW partition ID
0x0040_0000	0x0040_3FFF	Flash 16 KB UTest block	1
0x0040_4000	0x0040_7FFF	Flash 16 KB BAF block	1
0x0040_8000	0x0040_BFFF	Flash 16 KB Test block(1)	1
0x0040_C000	0x007F_FFFF	Reserved	
0x0080_0000	0x0080_3FFF	Flash 16 KB Data Block 0	2
0x0080_4000	0x0080_7FFF	Flash 16 KB Data Block 1	2
0x0080_8000	0x0080_BFFF	Flash 16 KB Data Block 2	2
0x0080_C000	0x0080_FFFF	Flash 16 KB Data Block 3	2
0x0081_0000	0x00FB_FFFF	Reserved	
0x00FC_0000	0x00FC_3FFF	Flash 16 KB Code Block 0	1
0x00FC_4000	0x00FC_7FFF	Flash 16 KB Code Block 1	1
0x00FC_8000	0x00FC_BFFF	Flash 16 KB Code Block 2	1
0x00FC_C000	0x00FC_FFFF	Flash 16 KB Code Block 3	1
0x00FD_0000	0x00FD_7FFF	Flash 32 KB Code Block 0	1
0x00FD_8000	0x00FD_FFFF	Flash 32 KB Code Block 1	1
0x00FE_0000	0x00FE_FFFF	Flash 64 KB Code Block 0	1
0x00FF_0000	0x00FF_FFFF	Flash 64 KB Code Block 1	1
0x0100_0000	0x0103_FFFF	Flash 256 KB Code Block 0	1
0x0104_0000	0x0107_FFFF	Flash 256 KB Code Block 1	1
0x0108_0000	0x010B_FFFF	Flash 256 KB Code Block 2	1
0x010C_0000	0x010F_FFFF	Flash 256 KB Code Block 3	1
0x0110_0000	0x0113_FFFF	Flash 256 KB Code Block 4	1
0x0114_0000	0x3FFF_FFFF	Reserved	

1. The Test Flash Block is not accessible directly through the Flash controller. It can be accessed only by the SSCM moduleduring the reset. The table reports the test Flash Block address, as it were visible from the Flash controller. In reality, theSSCM module accesses the Test Flash block using the real Flash block address space, which goes from 0x0080_8000 till 0x0080_BFFF.

Table 17. Peripheral bridge 1 memory map

Start address	End address	PBRIDGE Access Control Register	Peripheral
0xF800_0000	0xF800_3FFF	PACR0	PBRIDGE_1
0xF800_4000	0xFBBF_FFFF	—	Reserved
0xFBC0_0000	0xFBC0_3FFF	OPACR255	FlexPWM Synchronizer
0xFBC0_4000	0xFBC0_7FFF	OPACR254	FlexPWM_2
0xFBC0_8000	0xFBC0_BFFF	OPACR253	FlexPWM_3
0xFBC0_C000	0xFBC1_FFFF	—	Reserved
0xFBC2_0000	0xFBC2_3FFF	OPACR247	eTimer_2
0xFBC2_4000	0xFBC2_7FFF	OPACR246	eTimer_3
0xFBC2_8000	0xFBC2_BFFF	—	Reserved
0xFBC2_C000	0xFBC2_FFFF	OPACR244	CTU_1
0xFBC3_0000	0xFBDF_FFFF	—	Reserved
0xFBE0_0000	0xFBE0_3FFF	OPACR127	SAR_ADC_12BIT_3
0xFBE0_4000	0xFBE0_7FFF	OPACR126	SAR_ADC_12BIT_4
0xFBE0_8000	0xFBE0_BFFF	OPACR125	SAR_ADC_12BIT_5
0xFBE0_0000	0xFBE0_FFFF	—	Reserved
0xFBE1_0000	0xFBE1_3FFF	OPACR123	ADC 3 seq
0xFBE1_4000	0xFBE1_7FFF	OPACR122	ADC 4 seq
0xFBE1_8000	0xFBE1_BFFF	OPACR121	ADC 5 seq
0xFBE1_C000	0xFBE3_BFFF	—	Reserved
0xFBE3_C000	0xFBE3_FFFF	OPACR112	SAR_ADC_12BIT_B1 (ADC Supervisor PBRIDGE_1)
0xFBE4_0000	0xFBE4_3FFF	OPACR111	ADC Supervisor 1 seq

Table 17. Peripheral bridge 1 memory map (continued)

Start address	End address	PBRIDGE Access Control Register	Peripheral
0xFBE4_4000	0xFBE5_BFFF	—	Reserved
0xFBE5_C000	0xFBE5_FFFF	OPACR104	SENT_1
0xFBE6_0000	0xFBE6_FFFF	—	Reserved
0xFBE7_0000	0xFBE7_3FFF	OPACR99	DSPI_2
0xFBE7_4000	0xFBE7_7FFF	OPACR98	DSPI_3
0xFBE7_8000	0xFBE8_BFFF	—	Reserved
0xFBE8_C000	0xFBE8_FFFF	OPACR92	LINFlexD_1
0xFBE9_0000	0xFBE9_3FFF	OPACR91	LINFlexD_3
0xFBE9_4000	0xFBED_3FFF	—	Reserved
0xFBED_4000	0xFBED_7FFF	OPACR74	CAN_SUB_1 Message RAM
0xFBED_8000	0xFBEE_3FFF	—	Reserved
0xFBEE_4000	0xFBEE_7FFF	OPACR70	CAN_SUB_1_M_CAN_1
0xFBEE_8000	0xFBF0_3FFF	—	Reserved
0xFBF0_4000	0xFBF0_7FFF	OPACR62	CCCU_1 (CAN_SUB_1)
0xFBF0_8000	0xFBFB_01FF	—	Reserved
0xFBFB_0200	0xFBFB_023F	OPACR19	CMU_1 (System Clock)
0xFBFB_0240	0xFBFB_027F	OPACR19	CMU_2 (PBRIDGE_0/1 clock)
0xFBFB_0280	0xFBFB_02BF	OPACR19	CMU_3 (Peripheral clock)
0xFBFB_02C0	0xFBFB_02FF	OPACR19	CMU_4 (System div 2 clock)
0xFBFB_0300	0xFBFF_FFFF	—	Reserved

Table 18. Peripheral bridge 0 memory map

Start address	End address	PBRIDGE Access Control Register	Description
0xFC00_0000	0xFC00_3FFF	PACR0	PBRIDGE_0
0xFC00_4000	0xFC00_7FFF	PACR1	XBAR_0
0xFC00_8000	0xFC00_FFFF	—	Reserved
0xFC01_0000	0xFC01_3FFF	PACR4	SMPU_0
0xFC01_4000	0xFC01_7FFF	—	Reserved
0xFC01_8000	0xFC01_BFFF	PACR6	XBIC_0
0xFC01_C000	0xFC01_FFFF	—	Reserved
0xFC02_0000	0xFC02_3FFF	PACR8	PRAM_0
0xFC02_4000	0xFC02_7FFF	—	Reserved
0xFC02_8000	0xFC02_BFFF	PACR10	PCM_0
0xFC02_C000	0xFC02_FFFF	—	Reserved
0xFC03_0000	0xFC03_3FFF	PACR12	PFLASH_0
0xFC03_4000	0xFC03_FFFF	—	Reserved
0xFC04_0000	0xFC04_3FFF	PACR16	INTC_0
0xFC04_4000	0xFC05_7FFF	—	Reserved
0xFC05_8000	0xFC05_BFFF	PACR22	SWT_0
0xFC05_C000	0xFC06_7FFF	—	Reserved
0xFC06_8000	0xFC06_BFFF	PACR26	STM_0
0xFC06_C000	0xFC09_FFFF	—	Reserved
0xFC0A_0000	0xFC0A_3FFF	PACR40	eDMA_0
0xFC0A_4000	0xFFBF_FFFF	—	Reserved
0xFFC0_0000	0xFFC0_3FFF	OPACR255	FlexPWM Synchronizer
0xFFC0_4000	0xFFC0_7FFF	OPACR254	FlexPWM_0
0xFFC0_8000	0xFFC0_BFFF	OPACR253	FlexPWM_1

Table 18. Peripheral bridge 0 memory map (continued)

Start address	End address	PBRIDGE Access Control Register	Description
0xFFC0_C000	0xFFC1_FFFF	—	Reserved
0xFFC2_0000	0xFFC2_3FFF	OPACR247	eTimer_0
0xFFC2_4000	0xFFC2_7FFF	OPACR246	eTimer_1
0xFFC2_8000	0xFFC2_BFFF	—	Reserved
0xFFC2_C000	0xFFC2_FFFF	OPACR244	CTU_0
0xFFC3_0000	0xFFDF_FFFF	—	Reserved
0xFFE0_0000	0xFFE0_3FFF	OPACR127	SAR_ADC_12BIT_0
0xFFE0_4000	0xFFE0_7FFF	OPACR126	SAR_ADC_12BIT_1
0xFFE0_8000	0xFFE0_BFFF	OPACR125	SAR_ADC_12BIT_2
0xFFE0_C000	0xFFE0_FFFF	—	Reserved
0xFFE1_0000	0xFFE1_3FFF	OPACR123	ADC 0 seq
0xFFE1_4000	0xFFE1_7FFF	OPACR122	ADC 1 seq
0xFFE1_8000	0xFFE1_BFFF	OPACR121	ADC 2 seq
0xFFE1_C000	0xFFE3_BFFF	—	Reserved
0xFFE3_C000	0xFFE3_FFFF	OPACR112	SAR_ADC_12BIT_B0 (ADC Supervisor 0)
0xFFE4_0000	0xFFE4_3FFF	OPACR111	ADC Supervisor 0 seq
0xFFE4_4000	0xFFE4_FFFF	—	Reserved
0xFFE5_0000	0xFFE5_3FFF	OPACR107	FlexRay
0xFFE5_4000	0xFFE5_BFFF	—	Reserved
0xFFE5_C000	0xFFE5_FFFF	OPACR104	SENT_0
0xFFE6_0000	0xFFE6_FFFF	—	Reserved
0xFFE7_0000	0xFFE7_3FFF	OPACR99	DSPI_0
0xFFE7_4000	0xFFE7_7FFF	OPACR98	DSPI_1
0xFFE7_8000	0xFFE8_BFFF	—	Reserved

Table 18. Peripheral bridge 0 memory map (continued)

Start address	End address	PBRIDGE Access Control Register	Description
0xFFE8_C000	0xFFE8_FFFF	OPACR92	LinFlexD_0
0xFFE9_0000	0xFFE9_3FFF	OPACR91	LinFlexD_2
0xFFE9_4000	0xFFED_3FFF	—	Reserved
0xFFED_4000	0xFFED_7FFF	OPACR74	CAN_SUB_0 Message RAM
0xFFED_8000	0xFFEE_3FFF	—	Reserved
0xFFEE_4000	0xFFEE_7FFF	OPACR70	CAN_SUB_0_M_CAN_0
0xFFEE_8000	0xFFEE_BFFF	OPACR69	CAN_SUB_0_M_CAN_2
0xFFEE_C000	0xFFF0_3FFF	—	Reserved
0xFFF0_4000	0xFFF0_7FFF	OPACR62	CCCU_0 (CAN_SUB_0)
0xFFF0_8000	0xFFF3_BFFF	—	Reserved
0xFFF3_C000	0xFFF3_FFFF	OPACR48	JDC
0xFFF4_0000	0xFFF4_3FFF	—	Reserved
0xFFF4_4000	0xFFF4_7FFF	OPACR46	STCU2
0xFFF4_8000	0xFFF4_BFFF	OPACR45	JTAGM
0xFFF4_C000	0xFFF4_FFFF	—	Reserved
0xFFF5_0000	0xFFF5_3FFF	OPACR43	MEMU
0xFFF5_4000	0xFFF5_7FFF	—	Reserved
0xFFF5_8000	0xFFF5_BFFF	OPACR41	FCCU
0xFFF5_C000	0xFFF6_3FFF	—	Reserved
0xFFF6_4000	0xFFF6_7FFF	OPACR38	CRC_0
0xFFF6_8000	0xFFF6_BFFF	—	Reserved
0xFFF6_C000	0xFFF6_C1FF	OPACR36	DMAMUX_0
0xFFF6_C200	0xFFF6_C3FF	OPACR36	DMAMUX_1
0xFFF6_C400	0xFFF8_3FFF	—	Reserved

Table 18. Peripheral bridge 0 memory map (continued)

Start address	End address	PBRIDGE Access Control Register	Description
0xFFF8_4000	0xFFF8_7FFF	OPACR30	PIT_0
0xFFF8_8000	0xFFF9_7FFF	—	Reserved
0xFFF9_8000	0xFFF9_BFFF	OPACR25	WKPU
0xFFF9_C000	0xFFF9_FFFF	—	Reserved
0xFFFA_0000	0xFFFA_3FFF	OPACR23	PCU/PMC/Temperature sensor
0xFFFA_4000	0xFFFA_7FFF	—	Reserved
0xFFFA_8000	0xFFFA_BFFF	OPACR21	MC_RGM
0xFFFA_C000	0xFFFA_FFFF	—	Reserved
0xFFFB_0000	0xFFFB_003F	OPACR19	RC16M_DIG (IRCOSC)
0xFFFB_0040	0xFFFB_007F	—	Reserved
0xFFFB_0080	0xFFFB_00BF	OPACR19	OSC40M_DIG (XOSC)
0xFFFB_00C0	0xFFFB_00FF	—	Reserved
0xFFFB_0100	0xFFFB_013F	OPACR19	PLL DIG
0xFFFB_0140	0xFFFB_01FF	—	Reserved
0xFFFB_0200	0xFFFB_023F	OPACR19	CMU_0 (PLL)
0xFFFB_0240	0xFFFB_06FF	—	Reserved
0xFFFB_0700	0xFFFB_3FFF	OPACR19	MC_CGM
0xFFFB_4000	0xFFFB_7FFF	—	Reserved
0xFFFB_8000	0xFFFB_BFFF	OPACR17	MC_ME
0xFFFB_C000	0xFFFB_FFFF	—	Reserved
0xFFFC_0000	0xFFFC_3FFF	OPACR15	SIUL
0xFFFC_4000	0xFFFD_FFFF	—	Reserved
0xFFFE_0000	0xFFFE_3FFF	OPACR7	FLASH_0 (Flash memory registers)
0xFFFE_4000	0xFFFF_3FFF	—	Reserved

Table 18. Peripheral bridge 0 memory map (continued)

Start address	End address	PBRIDGE Access Control Register	Description
0xFFFF_4000	0xFFFF_7FFF	OPACR2	PASS
0xFFFF_8000	0xFFFF_BFFF	OPACR1	SSCM
0xFFFF_C000	0xFFFF_DFFF	—	Reserved
0xFFFF_E000	0xFFFF_FFFF	—	Reserved

Table 19. UTEST memory map

Start offset	End offset	Description
0x0000	0x0007	Temperature sensor calibration
0x0008	0x000B	Reserved
0x000C	0x000F	Test Mode Disable Seal
0x0010	0x001F	Test Mode Disable Group A
0x0020	0x002F	Factory Erase Diary Location
0x0030	0x003F	Test Mode Disable Group B
0x0040	0x005F	Customer Single Bit Correction area
0x0060	0x007F	Customer Double Bit Detection area
0x0080	0x009F	Customer EDC after ECC area
0x00A0	0x00BF	UID
0x00C0	0x00C3	SW-DCF Start address
0x00C4	0x00FF	Reserved
0x0100	0x0103	Test Mode Disable Override Passcode
0x0104	0x011F	Password to enable the Flash Test Mode when in FA
0x0120	0x013F	JTAG Password
0x0140	0x015F	PASS password - Group 0
0x0160	0x017F	PASS password - Group 1

Table 19. UTEST memory map (continued)

Start offset	End offset	Description
0x0180	0x019F	PASS password - Group 2
0x01A0	0x01BF	PASS password - Group 3
0x01C0	0x01FF	Reserved
0x0200	0x020F	Lifecycle slot 0 - ST Production
0x0210	0x021F	Lifecycle slot 1 - Customer Delivery
0x0220	0x022F	Lifecycle slot 2 - OEM Production
0x0230	0x023F	Lifecycle slot 3 - In Field
0x0240	0x024F	Lifecycle slot 4 - Failure Analysis
0x0250	0x02FF	Reserved
0x0300	0x0FFF	DCF Records
0x1000	0x1FFF	Customer OTP data

6 Functional safety

6.1 Introduction

Functional Safety implies requirements on several levels. This chapter comprises a collection of the hardware safety requirements for the SPC574Sx microcontroller unit (MCU).

6.2 Assumptions and usage constraints

The SPC574Sx offer a set of features to support applications that need to fulfill functional safety requirements as defined by automotive safety integrity level ASIL D of ISO 26262. The MCU is used within a system and, to allow analysis of the MCU's ability to satisfy safety requirements, certain assumptions on this system have to be imposed in line with the 'SEooC: Safety Element out of Context' concept. These assumptions are documented in the *SPC574Sx Safety Manual* (AN4247) as constraints on the allowed usage of the MCU in order to satisfy the requested safety level. Note that these requirements do not specify constraints on the MCU itself. Rather, they specify which features of the environment can be relied upon.

The MCU is used in automotive applications with integrity level up to ASILD, requiring a fail-silent fail-indicate MCU.

Safety analysis assumes a fault tolerant time interval (FTTI) of 10ms. ISO 26262 defines the FTTI as the time-span in which a fault or faults can be present in a system before a hazardous event occurs. It is the maximum acceptable time to move the system to a safe state in case of fault.

Shorter , and longer, time intervals are possible, depending on the actual combination of failure detection and signaling mechanisms as documented in the *SPC574Sx Safety Manual* (AN4247).

A driving cycle, that is the time of operation of the MCU without a power-on reset, is assumed to be 12 hours. Driving cycle specifies the multiple-point fault detection interval which is defined by the ISO 26262 as time span to detect multiple-point fault before it can contribute to a multiple-point failure.

The primary safety goal is that the MCU does not leave its safe states for intervals equal or longer than the fault tolerant time interval unless configured by the user to do so.

The secondary safety goal is that the MCU, or the software running on it, is able to detect the permanent unavailability of any safety mechanism required to achieve the primary safety goal. The unavailability of any such mechanisms to be detected at the latest in the successive driving cycle.

Safety goals are achieved by transitioning/maintaining the MCU in the following safe states:

- Unpowered
- In reset
- Operating correctly with respect to random hardware faults
- Explicitly indicating an internal error

Permanent cycling through otherwise safe states or permanent cycling between a safe state and an unsafe one, even if the latter is left before end of Fault Tolerant Time Interval (also

knowned as FTTI), is also considered a violation of the safety goal. This specifically concerns a continuous Reset-Start-of-Operation-Reset or Reset-Selftest-Reset sequence.

Safety analysis assumes that the system reacts safely to the MCU being in or entering all safe states.

If High-Z on an output pin is not safe, pull-ups or pull-downs need to be added at system level to safety-critical outputs depending on application requirements.

The overall system includes measures to monitor the error output pin(s) of the MCU and move the system into a safe state when an error is indicated.

If the MCU signals a failure via its error output pin(s), other output pins cannot be relied on.

Safety analysis assumes that external power of the appropriate voltage is supplied.

In addition different supplies, both internal and external, are supervised for high and low deviations by internal low-voltage and high-voltage detectors..

Note: Power supervision is implemented using LVD/HVD provided by the MCU as other safety mechanisms (for example external watchdog) exist to cover cases when the appropriate reaction of the MCU to the detection of the power failure cannot be guaranteed.

The MCU is kept in reset if external voltage is outside spec. User should protect the device against voltage over the maximum survivable voltage of the technology.

Safety analysis assumes that external watchdog detects any failures that completely disable the MCU and its safety mechanisms.

The external watchdog is assumed to be triggered periodically either by the software providing the safety function on the MCU or by a toggling protocol on the error_out pin.

Communication over high bandwidth communication protocols (for example CAN) is assumed to be protected by a fault-tolerant communication protocol (implemented by the operating system or the application).

6.3 Module categorization

The definition of the SPC574Sx safety concept and its validation (particularly the computation of Probabilistic Metric for Random Hardware Failures (PMHF)) has been done by splitting system modules into four categories as shown in *Table 20*.

Table 20. PMHF module categories

Module category	Category abbreviation	Description
Vital system modules	ViMo	All modules that can directly influence the correct operation of the software execution (by influence of the correct function of CPU, RAM, flash, and the bus interconnect). Examples of ViMo are the core, the eDMA, the crossbar, and the interrupt controller (INTC), as well as clock or power generation.
Peripheral modules	PeMo	All microcontroller modules involved in I/O operation, specifically including the I/O bridge.

Table 20. PMHF module categories (continued)

Module category	Category abbreviation	Description
Non-safety modules	NoSaMo	Modules that are not part of the safety function. For such modules, no hardware safety mechanisms have been provided on the chip. Examples include the Test and Debug logic, WKPU_NMI. Only their non-interference with modules from the other categories is ensured by hardware. In case they are used in a safety-relevant way, application measures are necessary to ensure their safe operation.
Other support modules	SuMo	These are modules which, although safety-relevant and in principle potential members of the ViMo category, cannot lead to a violation of the safety goal on their own. Typically, these will only have multiple point faults, but not single point faults. Example SuMo's are the RCCU and Clock Monitoring Unit (CMU).

Module classifications for SPC574Sx are shown in *Table 21*.

Table 21. Module classification

Category abbreviation	Description
ViMo	Master Safety Core, eDMA, INTC, XBAR, SRAM controller, System RAM array, flash memory controller, flash memory array, MC_RGM, Dual PLL Digital, XOSC, MC_CGM, MC_ME, voltage regulator, MC_PMC, STCU2, SSCM, PIT, Decorated Storage Memory Controller (DSMC), MC_PCU, PMC_dig, DMACHMUX, Safety Core Cache controller and memories, Safety Core TCMs
PeMo	System Timer (STM), eCTU, Flexray, SENT, FlexPWM, eTimer, PIT, DMA RAM, SAR ADC, LinFlexD, DSPI, SIUL2 and PBRIDGEn
NoSaMo	Nexus, JTAG, WKUP and SPU
SuMo	RCCU, Checker Safety Core, CRC Units, Temperature Sensors, IRCOSC, CMUs, HVDs, LVDs, Bandgap reference, LBIST controllers, REG_PROT, SWT, FCCU, MEMU, and SMPU

6.4 Safety architecture overview

This chapter gives an overview of the safety implementation for the SPC574Sx, and discusses the operation of the main safety mechanisms.

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

- For the safety related bus masters, namely core and DMA, safety is ensured by delayed lockstep approach. The same approach is adopted for the INTC.
- The integrity of stored data is ensured by ECC with address encoding (RAM only) and selected additional measures for individual modules.
- The safety of the data path to storage is ensured by End-to-End ECC (E2E ECC) with address encoding and selected additional measures for individual modules. For

peripherals, safety of the data path is ensured by E2E EDC. The E2E EDC protection terminates at peripheral bridge.

- The safety of the periphery is ensured by application-level measures (such as connecting one sensor to different I/O modules, sensor validation by sensor fusion, and implementation of a fault tolerant communication protocol). Hardware supports this application-level redundancy by providing redundant modules connected to different peripheral bridges (PBRIDGE) to maximize the independence between the monitored and monitoring resources.
- Clock and power generation and distribution are supervised by dedicated monitors.
- The Fault Collection and Control Unit is responsible for collecting and reacting to failure notifications. The Memory Error Management Unit (MEMU) is responsible for the collection and reporting (to the FCCU) of error events in system memories including both volatile and volatile ones. Memory related error events include ECC corrections and detections as well as faults detected by memory BISTs.
- Logic and Memory BIST (L/MBIST) are provided to avoid the accumulation of latent faults in the functional logic as well as the safety integrity mechanisms and in memory arrays. Dedicated mechanisms are provided to check the availability of safety mechanisms and the functionality of each error reaction path (such as by fake fault injection).
- Common Cause Failures (CCF) are dealt with by a set of measures for both control and avoidance of CCFs spanning system-level approaches (such as temperature and nonfunctional signal monitoring).

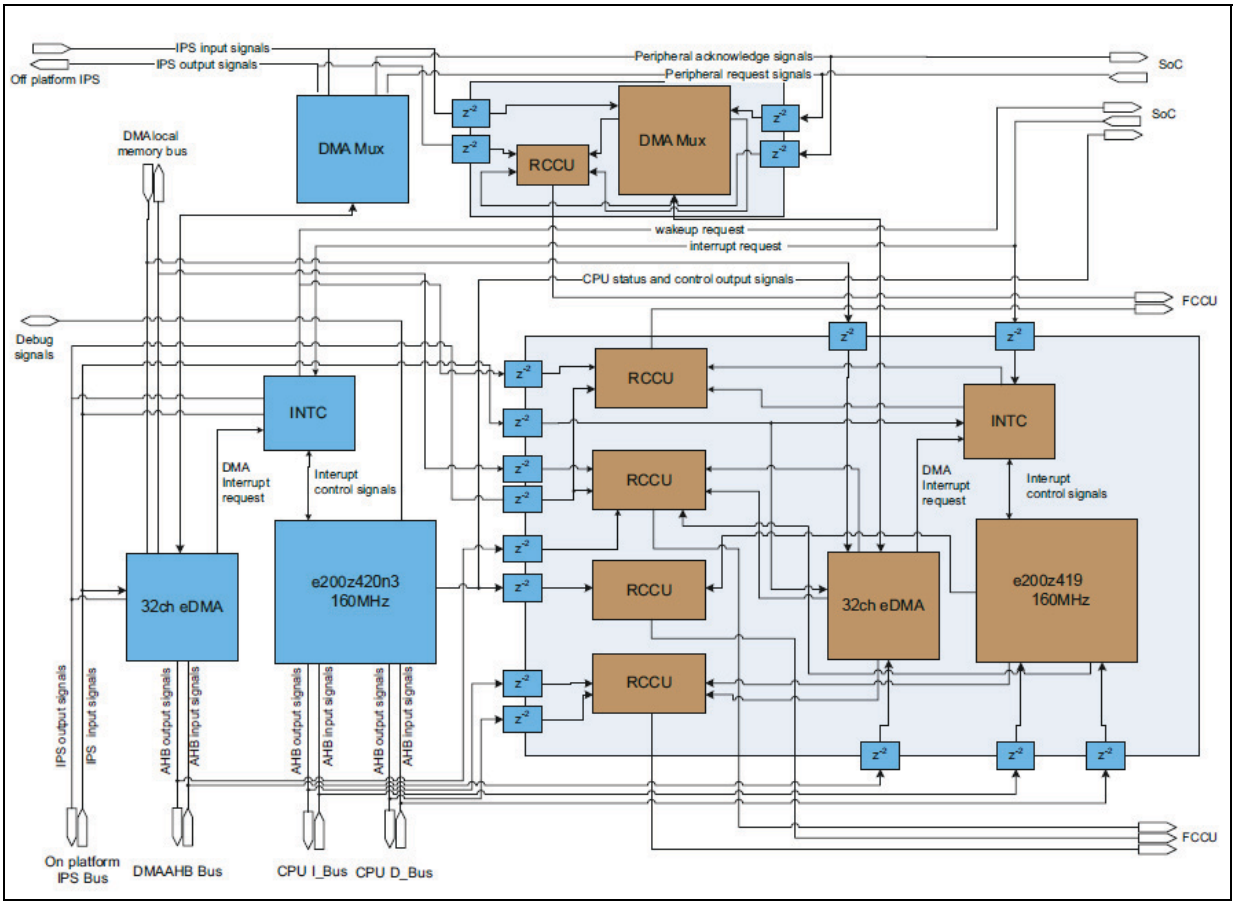
6.4.1 Lockstep

SPC574Sx replicates its safety-relevant processing elements and compares their operation in lockstep.

The following elements are replicated:

- Core
- DMA (including DMA Mux)
- Interrupt controller (INTC)

Figure 9. Replication architecture



Equivalent operation of both channels are supervised by checker units (RCCU) on all functional signals, leaving the channels for the rest of the MCU. A functional signal is a signal participating in the functional operation of a channel. Test signals are not functional signals.

The FCCU is notified of any deviations between the supervised signals.

There is a two-clock delay between the two channels to reduce susceptibility to clock and power glitches.

The checker core and the replicated DMA do not have a direct connection to the Crossbar Switch (XBAR). All their outputs directed to the XBAR end in RCCUs and all their inputs associated with the XBAR are split off to the master IP's inputs from the XBAR.

If additional connections exist via which information can leave the checker channel, these are also compared to the respective signals from the main channel. Examples are DMA acknowledgements to peripherals.

Similarly, if additional connections exist via which information can enter the checker channel, these are split off from the master IP's input. Examples are IRQ signals from peripheral to INTC and peripheral DMA requests.

Signals internal to the channel are not compared (as they are already covered by redundancy). Particularly no RCCU is needed on signals from or to core, to or from INTC and DMA, from or to DMA and to or from INTC.

No dynamic switching of lockstep mode is possible. Lockstep mode can only be toggled by reprogramming information in the Flash and executed on the subsequent boot. When Lockstep mode is switched off, the checker IPs are disabled (see Miscellaneous DCF in *Chapter 9: Device Configuration Format (DCF) Records*).

A register visible to the core indicates whether or not the system is in lockstep mode. Before starting safety-relevant operations, Software checks that the lockstep mode is enabled and also configures the FCCU to react to lockstep disablement.

6.4.2 End to end (E2E) protection on the data path and memory ECC

Error correction and detection mechanism is used to protect the access to the system memories. The general scheme is:

- An ECC error is reported to (stored into) the MEMU, which in turn, will trigger the FCCU
- An EDC or signal integrity check error is reported to the FCCU

All codes have Hamming distance of at least 4 (ECC) or 3 (EDC) except for the Cache where also EDC needs to have a Hamming distance of 4.
All storage arrays in the MCU use column MUXing to sufficiently reduce the probability of triple bit errors.
Error correction to be performed on data and check-bits and both data and check-bits to be returned to the requesting master.
Error correction operations do not cause additional wait states.

Addressing errors to be covered by the inclusion of the address into the End-to-End EDC/ECC. CAN RAM (Message Buffers) to be protected by ECC.

Figure 10 shows the overall ECC architecture. The picture shows when the ECC checkbits are connected between the various modules. Each ECC engine has a number ID, which is looked up into *Table 22* for additional details of each ECC engines. *Table 22* clarifies whether the ECC checkbits contain the address component and where an ECC error is reported to. Refer to *Figure 24* for the list of FCCU input triggers related to the ECC/EDC.

Figure 10. e2eECC general architecture

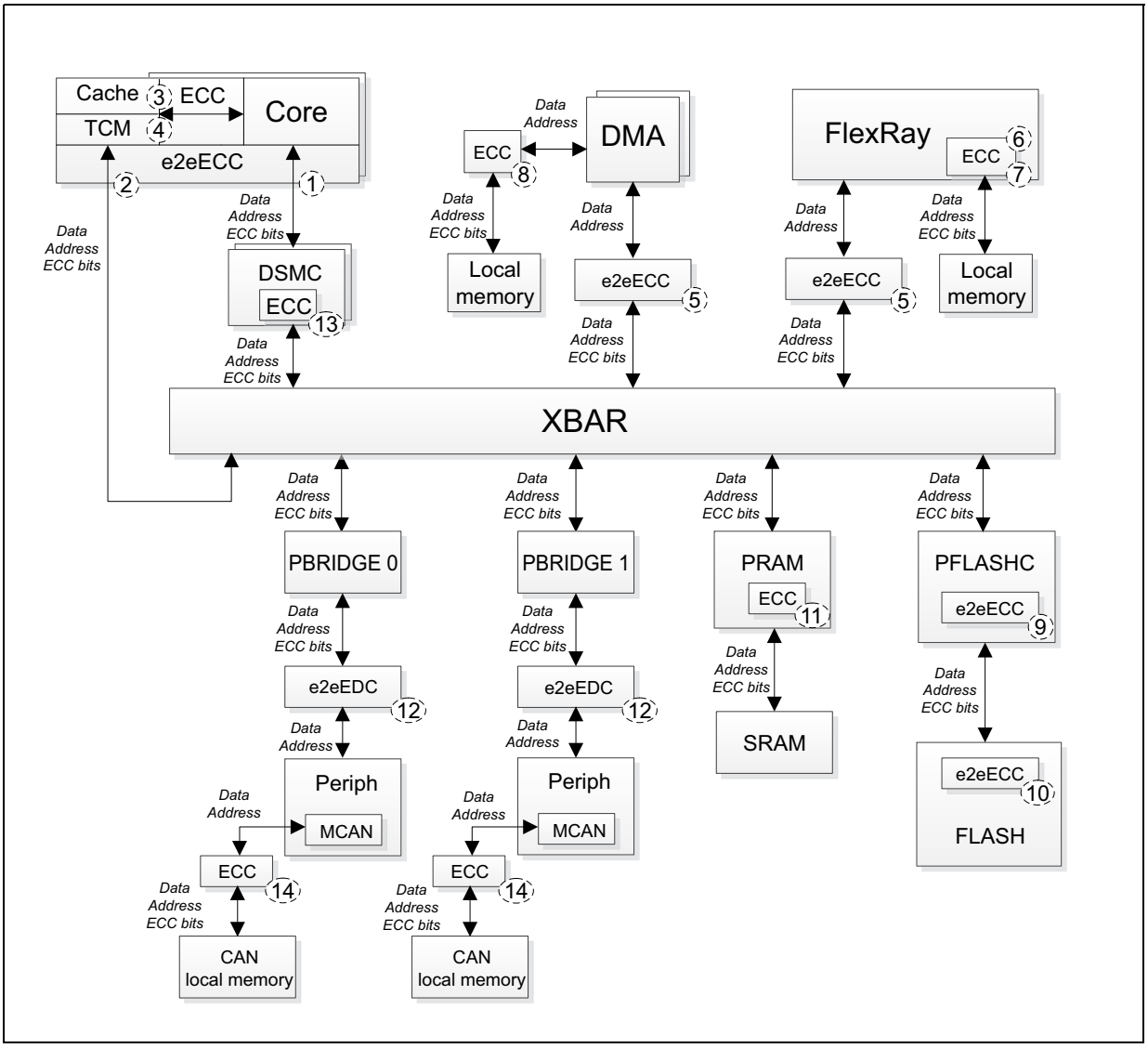


Table 22 shows the characteristics of the ECC engines distributed around the system. The “ID” column reflects the numbering used in *Figure 10*.

For all ECC which includes address into the ECC codebit calculation, the All-X words will not be handled specially. They can be a valid, a correctable or an non-correctable word depending on address.

6.4.3 Additional address coverage

The values stored in the address and control signal input latches of a RAM array are fed back to the control module of that array together with whatever result an access had.

RAM controller, of RAM with address coverage requirements, compare the values fed back from the latches with the values which were written to the latches when it evaluates/forwards the results from the access to the RAM array.

If a deviation is detected during the comparison it is signaled to the FCCU.

Only non-replicated, safety-relevant RAM controller require additional address coverage.

The wordline and the column selected to retrieve the data from the Flash array are re-encoded to the address which corresponds to that wordline/column combination.

This re-encoding is also capable of detecting if more than one wordline/column or none is selected.

The re-encoded address is returned to the Flash controller together with the read data or the write acknowledgement.

Table 22. ECC Engine Summary

ID	Interface	Correct	Detect	Address covered by ECC	Reaction upon ECC error within the address	Data and ECC bit size	Report to
1	CPU I/D-Bus	1-bit	2-bits	Yes	No correction is done. A machine check exception is triggered. Reported as uncorrectable Error to FCCU via MEMU.	64+8	MEMU/FCCU
2	TCM I/D External Port	1-bit	2-bits	Yes	No correction is done. A machine check exception is triggered. Reported as uncorrectable Error to FCCU via MEMU.	64+8	MEMU/FCCU
4	CPU to I/D-TCM	1-bit	2-bits	Yes	No correction is done.	32+8	MEMU/FCCU
3	CPU to I/D-Cache	—	2-bits	Yes	Cache is invalidated. The Error is reported as correctable error to FCCU via MEMU.	64+8	MEMU/FCCU
5	DMA and FlexRay AHB I/F	1-bit	2-bits	Yes	Reported as uncorrectable multi-bit error	64+8	MEMU/FCCU
6	FlexRay PE RAM I/F	1-bit	2-bits	No	—	16+10	MEMU/FCCU
7	FlexRay CHI RAM I/F	—	1/ 2-bits	No	—	96+30	MEMU/FCCU
8	DMA local RAM I/F	1-bit	2-bits	Yes	Indicated by a dedicated output signal (dma_tcd_addr_ecc_err) at platform top	64+8	MEMU/FCCU

Table 22. ECC Engine Summary (continued)

ID	Interface	Correct	Detect	Address covered by ECC	Reaction upon ECC error within the address	Data and ECC bit size	Report to
9	Flash controller	1-bit	2-bits	Yes	Reported as noncorrectable multi-bit error	64+8	MEMU/FCCU
10	Flash	1-bit	2-bits	No	Reported as noncorrectable multi-bit error	64+8	MEMU/ FCCU (through Flash controller)
11	RAM controller	1-bit	2-bits	Yes	Reported as noncorrectable multi-bit error	64+8	MEMU/FCCU
12	AIPS0/1	—	2-bits	Yes	Reported as fault to FCCU.	64+8	FCCU
13	DMSC	1-bit	2-bits	Yes	Reported as noncorrectable multi-bit error	64+8	MEMU/FCCU
14	CAN local RAM	1-bit	2-bits	No	—	32+7	MEMU/FCCU

6.4.3.1 ECC engine embedded into the CPU vs. Machine check exception

The CPU embeds an ECC engine, which detects/corrects the failed bits occurring during the access to either the system (AHB interface) or local (TCM and Cache) memories. The ECC errors are considered HW errrors and are classified as “machine check”. Machine check causes are divided into different types:

- Error Report Machine Check conditions
- Non-Maskable Interrupt (NMI) machine check exceptions
- Asynchronous machine check exceptions

In addition to the ECC errors, the following additional sources can generate a machine check exception:

1. p_mcp_b signal transition from negated to asserted
2. ISI or Bus Error on first instruction fetch for an exception handler
3. External bus error
4. Stack limit check failure

Upon a machine check exception, the p_mcp_out signal is asserted. This signal is connected to the FCCU. In order to understand what is the cause of a machine check, the application reads the CPU MSCR register. In fact, when the processor takes a machine

check interrupt, it updates the Machine Check Syndrome register (MCSR) to differentiate between machine check conditions. Following status bits are available:

- Machine check input pin
- Instruction Cache data array parity error (if I-Cache is available)
- Data Cache data array parity error (if D-Cache is available)
- ISI or Bus Error on first instruction fetch for an exception handler
- Instruction Cache Tag parity error
- Data Cache Tag parity error (if D-Cache is available)
- Instruction Cache Lock error (if I-Cache is available)
- Data Cache Lock error (if D-Cache is available)
- NMI input pin
- MCAR Address Valid
- MCAR holds Effective Address
- User (MCAR value captured in user mode)
- Instruction Fetch Error Report
- Load type instruction Error Report
- Store type instruction Error Report
- Guarded instruction Error Report
- Stack Access Limit Check Error
- Instruction Mem (IMEM) Parity Error (if I-MEM is available)
- Data Mem (DMEM) Parity Error (if D-MEM is available)
- Data Mem (DMEM) Write Parity Error (if D-MEM is available)
- Read bus error on Instruction recovery linefill
- Read bus error on data load or linefill
- Write bus error on store to bus or DMEM imprecise write error
- Write bus error on buffered store to bus with DSI signaled. Set concurrently with BUS_WRERR for this case

In addition to the machine check exception, the CPU reports the following signals about the ECC errors:

- single error
- double error
- address error
- address of failed memory location

These information are separated for errors happening in the TCM, Cache, or external AHB interface. The MEMU is triggered accordingly to the type of error and where did it happen.

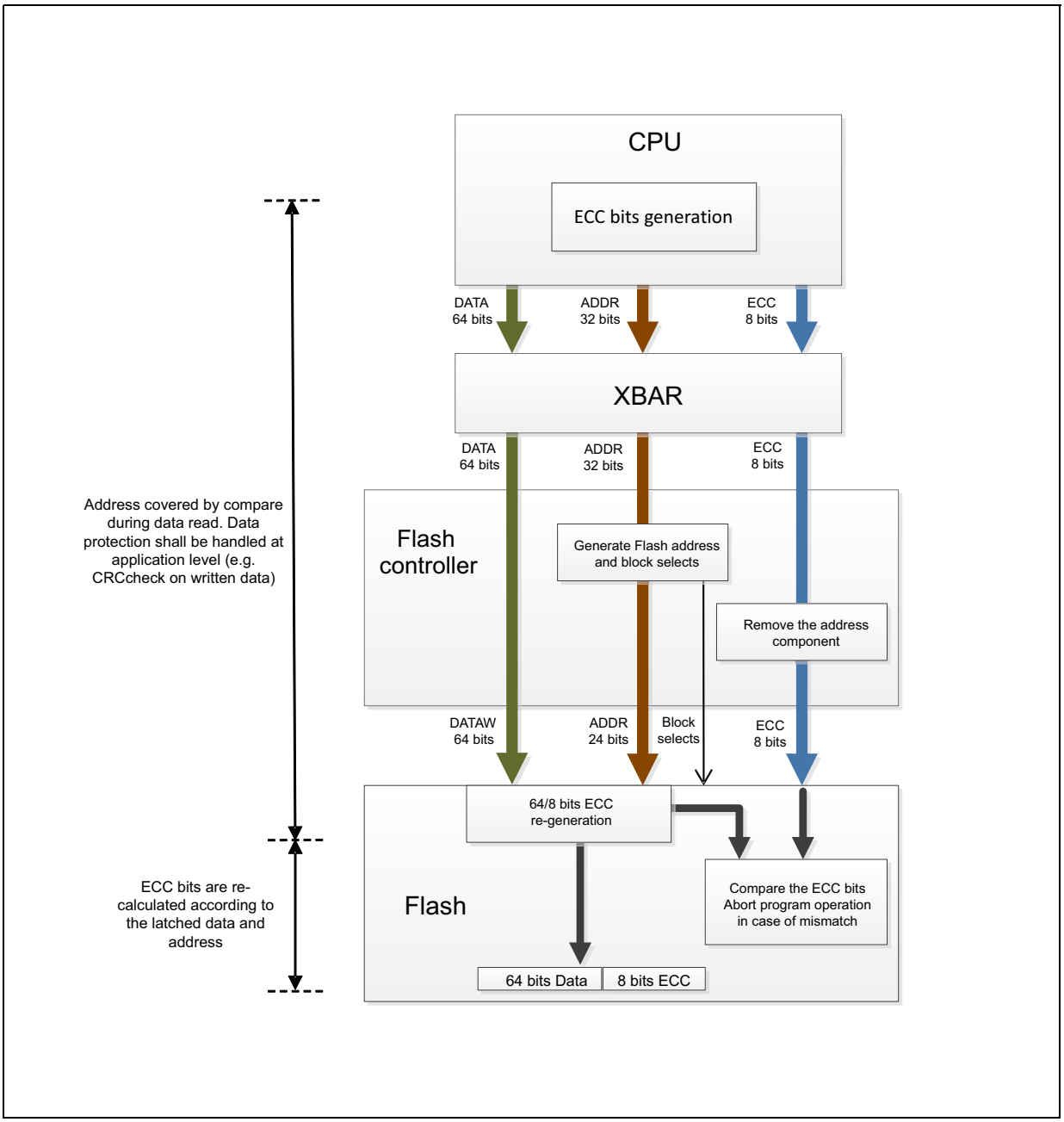
6.4.3.2 CPU to Flash write path protection

Figure 11 shows the CPU to the Flash memory write path. The ECC bits are generated by the CPU, based on AHB data and address, and checked within the Flash. The Flash memory controller translates the address into the Flash block sector and block enable. The Flash controller removes the AHB address component from the ECC bits and transmits the new ECC bits to the Flash memory. The Flash memory checks the received ECC bits vs. the latched data and aborts the writing operation in case a mismatch is detected. The ECC bits being effectively programmed into the Flash are generated within the Flash memory, based

on the latched data and covers them with the ECC bit generated within the Flash memory itself.
Because all-1 (data+checkbits) is a valid combination (e.g. erased sector), it sets up (as a result of a design failure mode analysis) some safety HW/SW mechanism to detect whether we are reading all-1 because of some failures or because the block is really erased.

Data read and CRC are the measures intended to verify the integrity of the programmed data.

Figure 11. CPU to Flash write path protection



6.4.3.3 CPU to Flash read path protection

Figure 12 shows the CPU to the Flash memory read path. The data is supposed to have been stored into the Flash memory together with the ECC bits necessary to cover the data. The Flash memory is able to correct 1 bit failure and detect 2 bit failures.
Upon data reading, the Flash memory executes an ECC correction, followed by an EDC check after ECC.
The occurrence of an EDC error after ECC error is reported to the FCCU via glue logic within the Flash controller, which takes care to report non- EEProm failures only whereas the occurrence of an ECC error is reported to the Flash memory controller, which in turn will report to the MEMU, which in turn will trigger the FCCU.

In addition, the Flash executes a compare between the latched address and the address used internally to address the memory array. The wordline and the column selected to retrieve the data from the Flash array are re-encoded to the address which corresponds to that wordline/column combination.

This re-encoding is also capable to detect if more than one wordline/column or none is selected. A mismatch between them is reported to the Flash controller, which in turn will trigger the FCCU.

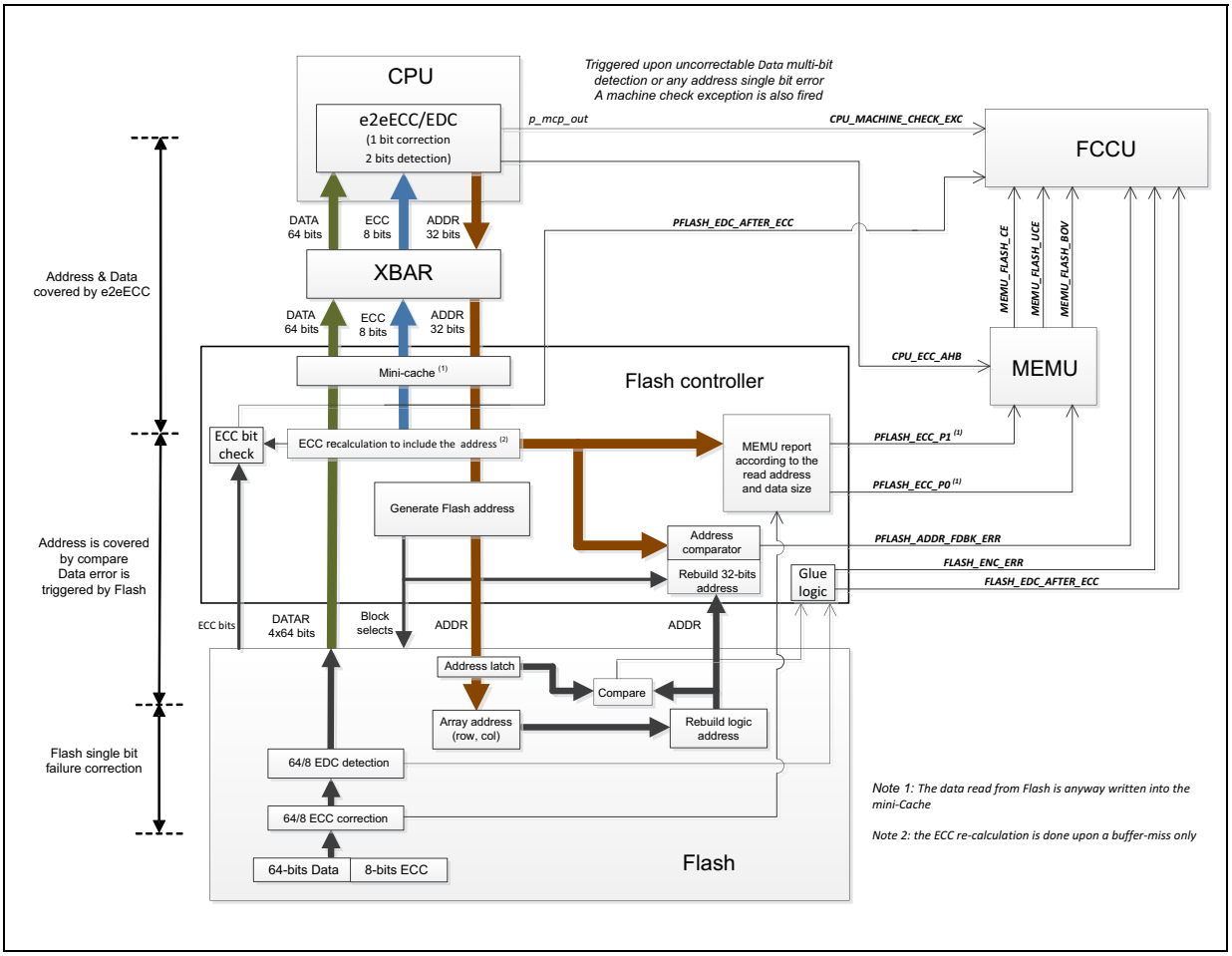
The Flash memory also sends to the Flash memory controller the address where the data has been read from. The Flash memory controller compares this address with the one sent to the Flash memory. An error is reported to the FCCU. This comparison covers the address path between the Flash memory controller and the Flash memory for both the read and write operations.

The Flash memory controller re-calculate the ECC bits taking the address received from the XBAR, in order to match the different address between the one sent to Flash and the one received from the XBAR. After that, the data and ECC checkbits are stored into the mini-cache.

The Flash controller also executes an integrity check on the newly calculated ECC bits and triggers an alarm to the FCCU in case of failure.

The data is finally transferred to the CPU, which will evaluate the check bits. In case of any address bit failure or data multi-bit non-correctable error, a machine check exception is fired and the FCCU is triggered by the p_mcp_out signal. The reason for the machine check can be read from the CPU MCSR register. In parallel, the information about the error is reported to the MEMU, which in turn triggers the FCCU.

Figure 12. CPU to Flash read path protection

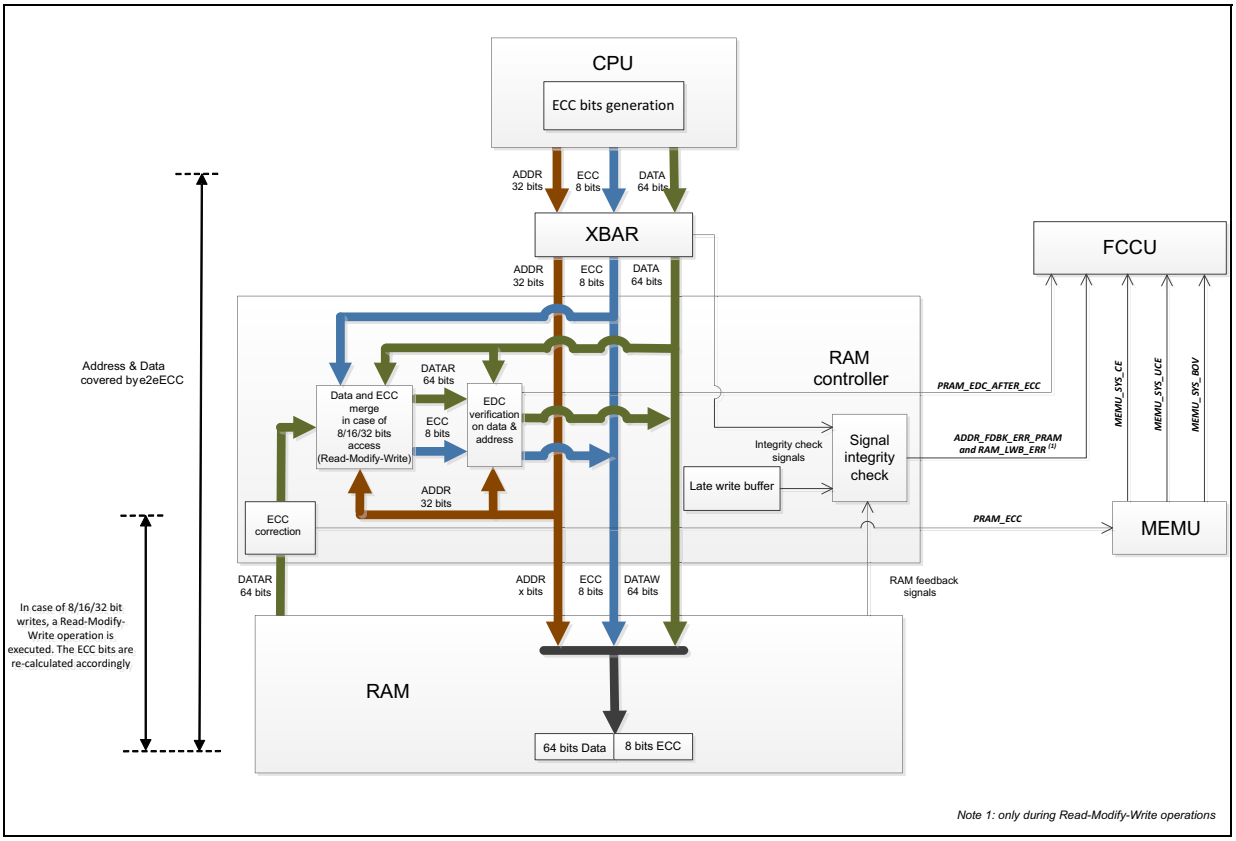


6.4.3.4 CPU to RAM write path protection

Figure 13 shows the CPU to the RAM system memory write path. The ECC bits are generated by the CPU and cover both the AHB address and data (8 ECC bits are used). The RAM controller checks the size of the data being written. In case the size is 64-bit, then the data and the ECC bits are directly stored into the RAM, without any EDC check. Vice verse, in case a RAM write operation is requested for some data, whose size is less than maximum data width, the RAM controller executes a Read-Modify-Write operation.

The signal integrity check module is embedded inside the RAM controller and guarantees the integrity of the signals at the input and output of the RAM controller. If a deviation is detected during the comparison it is signaled to the FCCU.

Figure 13. CPU to RAM write path protection



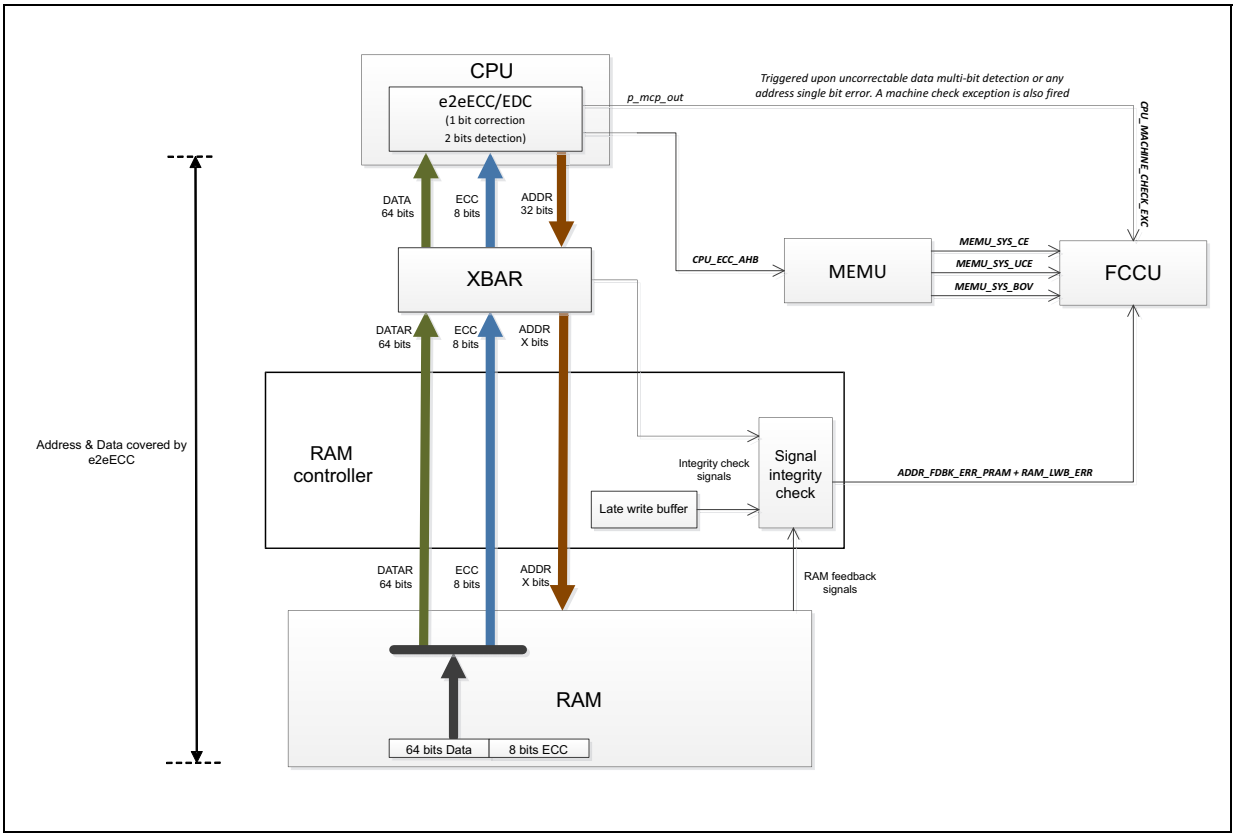
6.4.3.5 CPU to RAM read path protection

Figure 14 shows the CPU to the RAM system memory read path. The data and ECC bits are stored into the RAM and read back from the RAM controller. The signal integrity check module is embedded inside the RAM controller and guarantees the integrity of the signal at the input and output of the RAM controller. The data is finally transferred to the CPU, which will detect any single or double bit fails and report them to the FCCU.

The signal integrity check module is embedded inside the RAM controller and guarantees the integrity of the signals at the input and output of the RAM controller. If a deviation is detected during the comparison it is signaled to the FCCU.

In case of any address bit failure or data multi-bit uncorrectable error detected by the CPU, a machine check exception is fired and the FCCU is triggered by the p_mcp_out signal. The reason for the machine check can be read from the CPU MCSR register. The reason for the machine check can be read from the CPU MCSR register. In parallel, the information about the error is reported to the MEMU, which in turns triggers the FCCU.

Figure 14. CPU to RAM read path protection

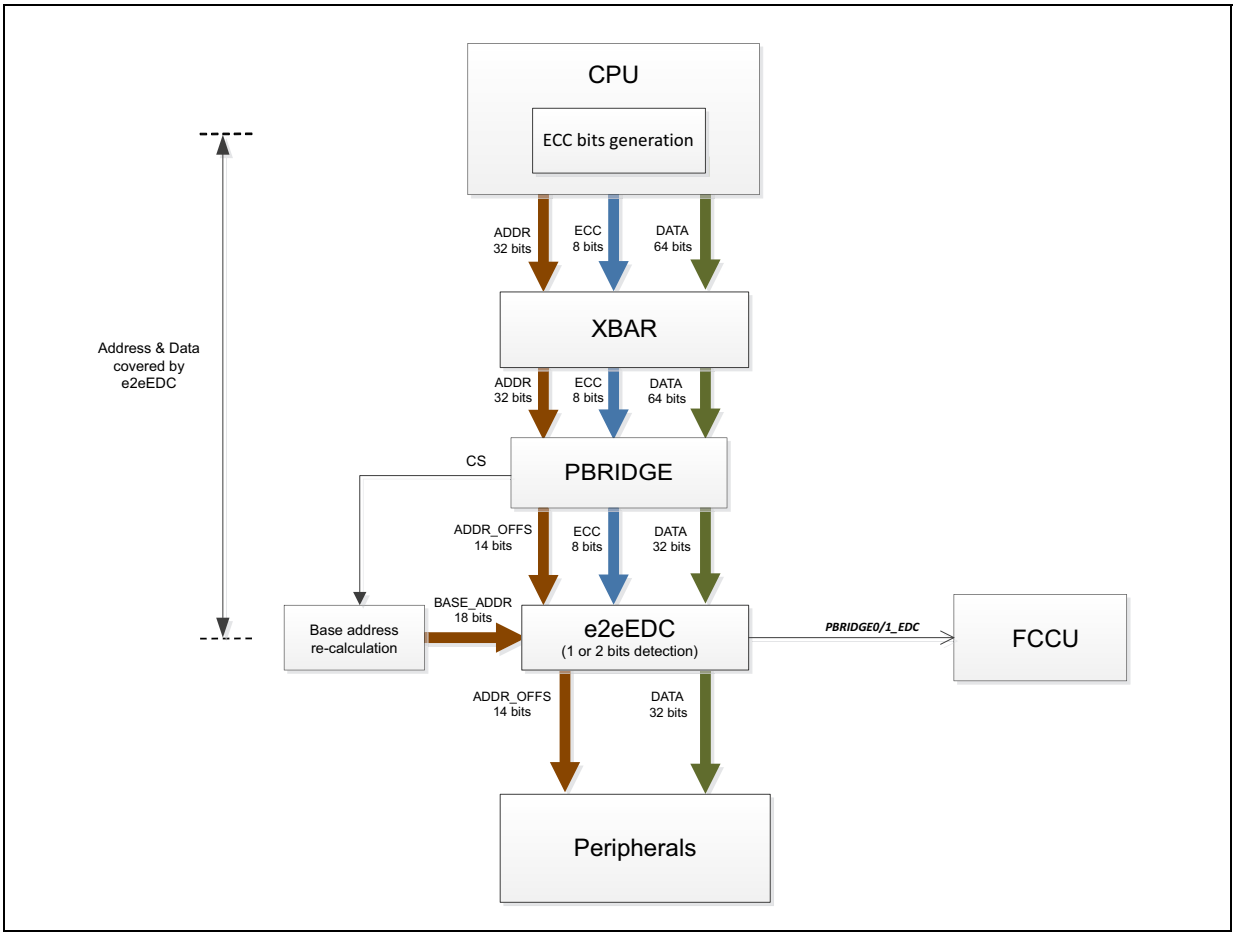


6.4.3.6 CPU to peripheral write path protection

Figure 15 shows the CPU to the peripheral write path. The ECC bits are generated by the CPU and cover both the address and data (8 ECC bits are used). The e2eEDC block after the AIPS bridge checks the received data and address for any bit failures and eventually trigger a signal to the FCCU. No data correction is performed.

The safety mechanism covers also the correctness of the re-mapping of the AHB read/write data busses onto the IPS read/write data busses.

Figure 15. CPU to peripheral write path protection

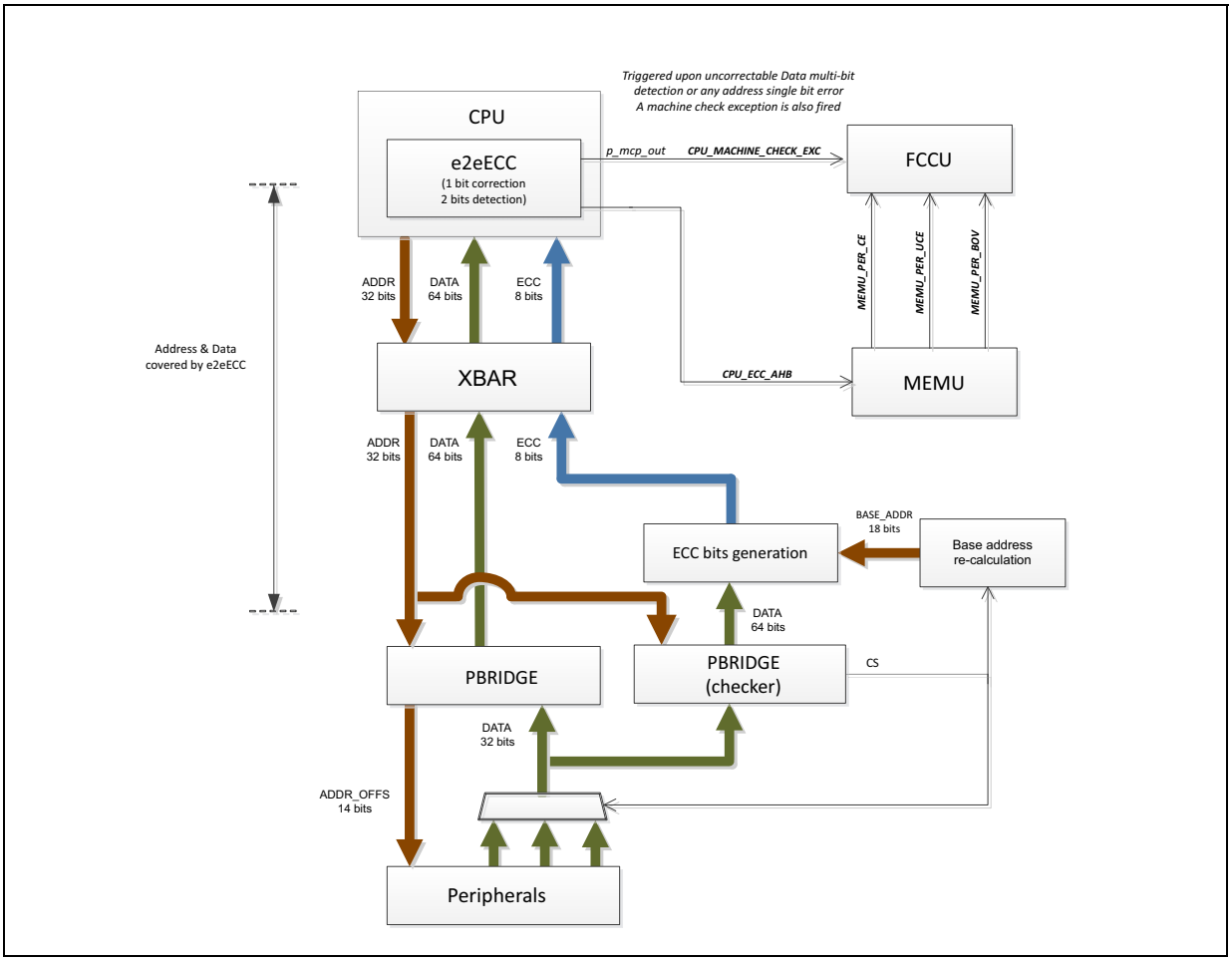


6.4.3.7 CPU to peripheral read path protection

Figure 16 shows the CPU to the peripheral read path. The ECC bits are generated by the e2eECC module connected under the AIPS port and cover both the address and data (8 ECC bits are used), the latter being output from the multiplexed peripheral data read busses. The AIPS is duplicated in order to check the correctness of the signals. In case of any address bit failure or data multi-bit non-correctable error, a machine check exception is fired and the FCCU is triggered by the p_mcp_out signal. The reason for the machine check can be read from the CPU MCSR register. In parallel, the information about the error is reported to the MEMU, which in turn triggers the FCCU.

The safety mechanism covers also the correctness of the re-mapping of the AHB read/write data busses onto the IPS read/write data busses.

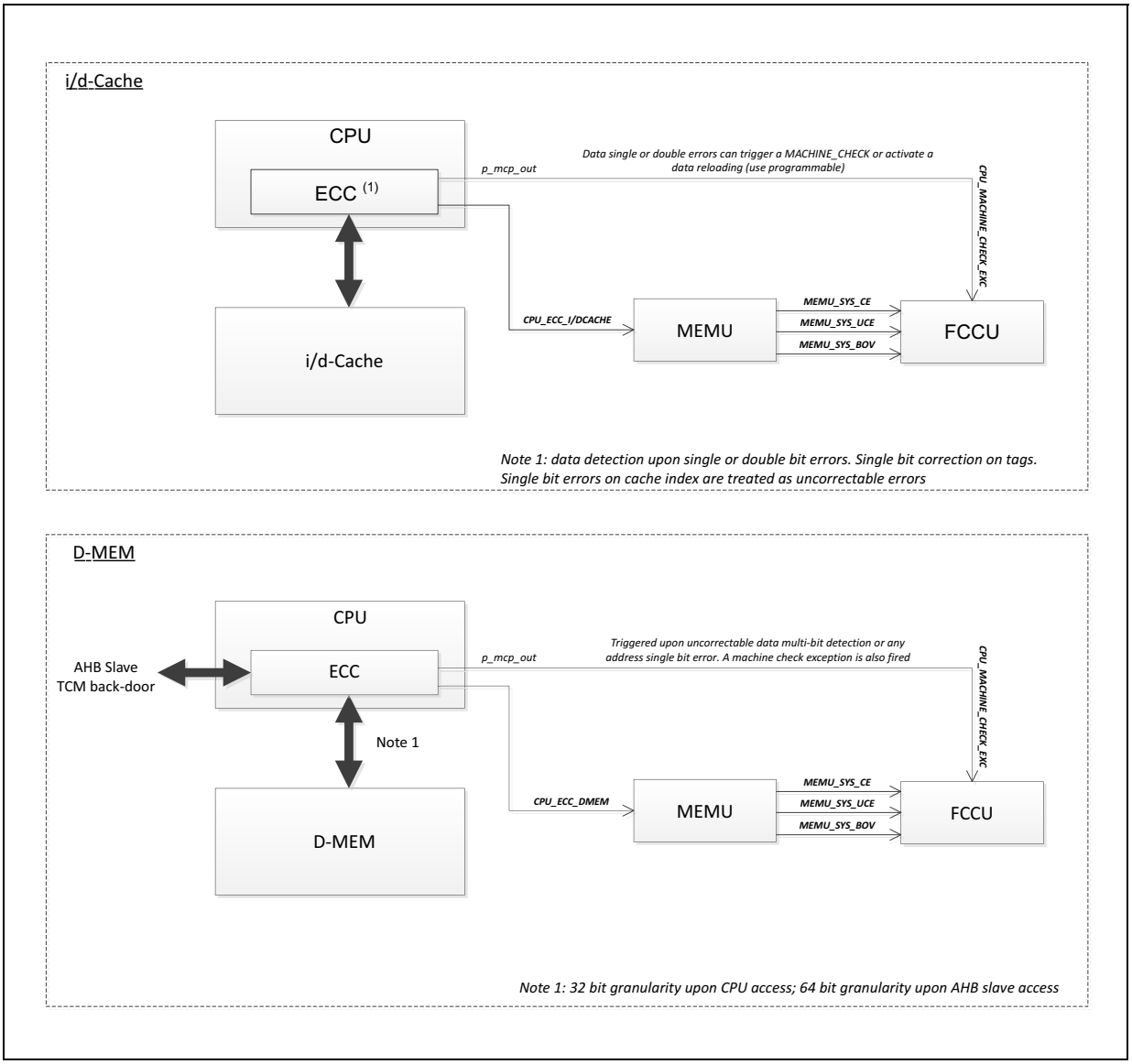
Figure 16. CPU to peripheral read path protection



6.4.3.8 ECC protection for TCM and Cache memories

Figure 17 shows the ECC protection when the CPU accesses the TCM or the Cache. Upon writing, the ECC engine generates the check bits. Upon reading, the check bits are verified and, for the TCM, the data eventually corrected, while for the cache memory no correction is done (except for the cache tags data). In case of any TCM address bit failure, cache index failure, cache tags non-correctable errors, or data multi-bit non-correctable error, a machine check exception can be fired and the FCCU is triggered by the p_mcp_out signal. The reason for the machine check can be read from the CPU MCSR register. In parallel, the information about the error is reported to the MEMU, which in turns triggers the FCCU. The cache configuration registers can be optionally programmed in order to reload the data value upon an ECC error.

Figure 17. ECC protection for TCM and Cache memories



When a write operation to the TCM is done through the XBAR TCM slave port, the ECC checkbits, as generated by the accessing master, are stored into the TCM, following a CPU internal transformation, because the DMEM ECC granularity is 32-bits.
When a read operation from the TCM is done through the XBAR TCM slave port, the originally written ECC bits are internally transformed back to the AHB system ECC granularity (64-bits).

In order to have a match between the checkbits written by non-CPU masters, it is mandatory that the DMEM is seen at the same base address, for both the CPU and

non-CPU accesses.

6.4.3.9 ECC protection for Decorated Memory Storage

The z4h processor embeds a feature used to run atomic read-modify-write instructions. A set of extended instructions are available for this. A functional module, called DSMC

(Decorated Storage Memory Controller), is placed between the CPU and the XBAR. The DSMC decodes the extended instruction and drives the necessary read-modify-write operations. The DSMC module contains the ECC logic applied on the data read during the read-modify-write operation. *Figure 18* and *Figure 19* show the organization of the ECC for the DMSC operations. In reality, only one single operation within the DSMC has no write-back result, while all the other operations are effectively Read-Modify-Write operations. The DSMC reports single/multi-bits errors to the MEMU for the data read from the slave memory address, and to the FCCU for the detection of ECC errors on the master write data and address. In order to cover the faults of the DSMC and the ECC re-generation logic, the DSMC is duplicated and the signals are checked with a dedicated RCCU.

Figure 18. ECC scheme for DSMC operation including a Write-Back operation

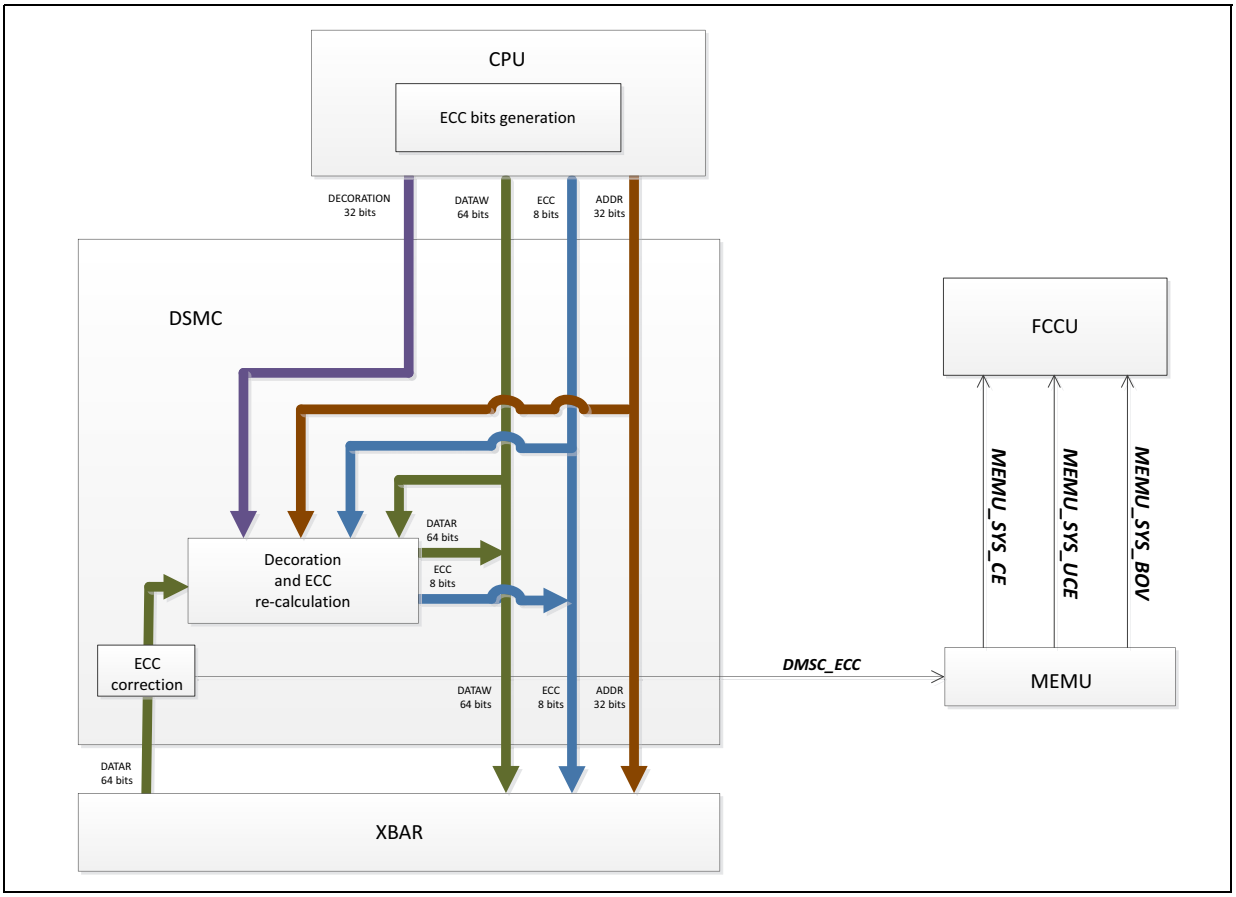
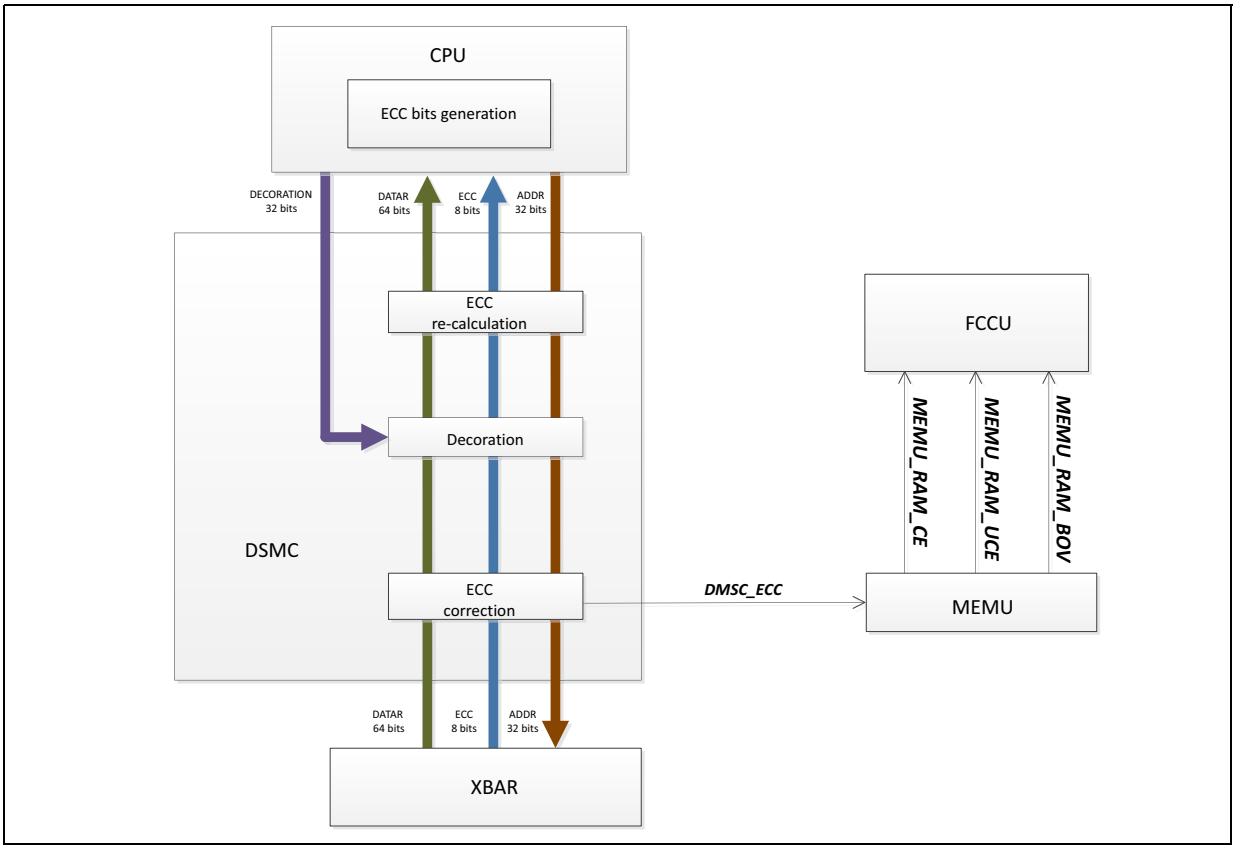


Figure 19. ECC scheme for DSMC operation with no Write-Back operation



6.4.3.10 ECC protection for DMA

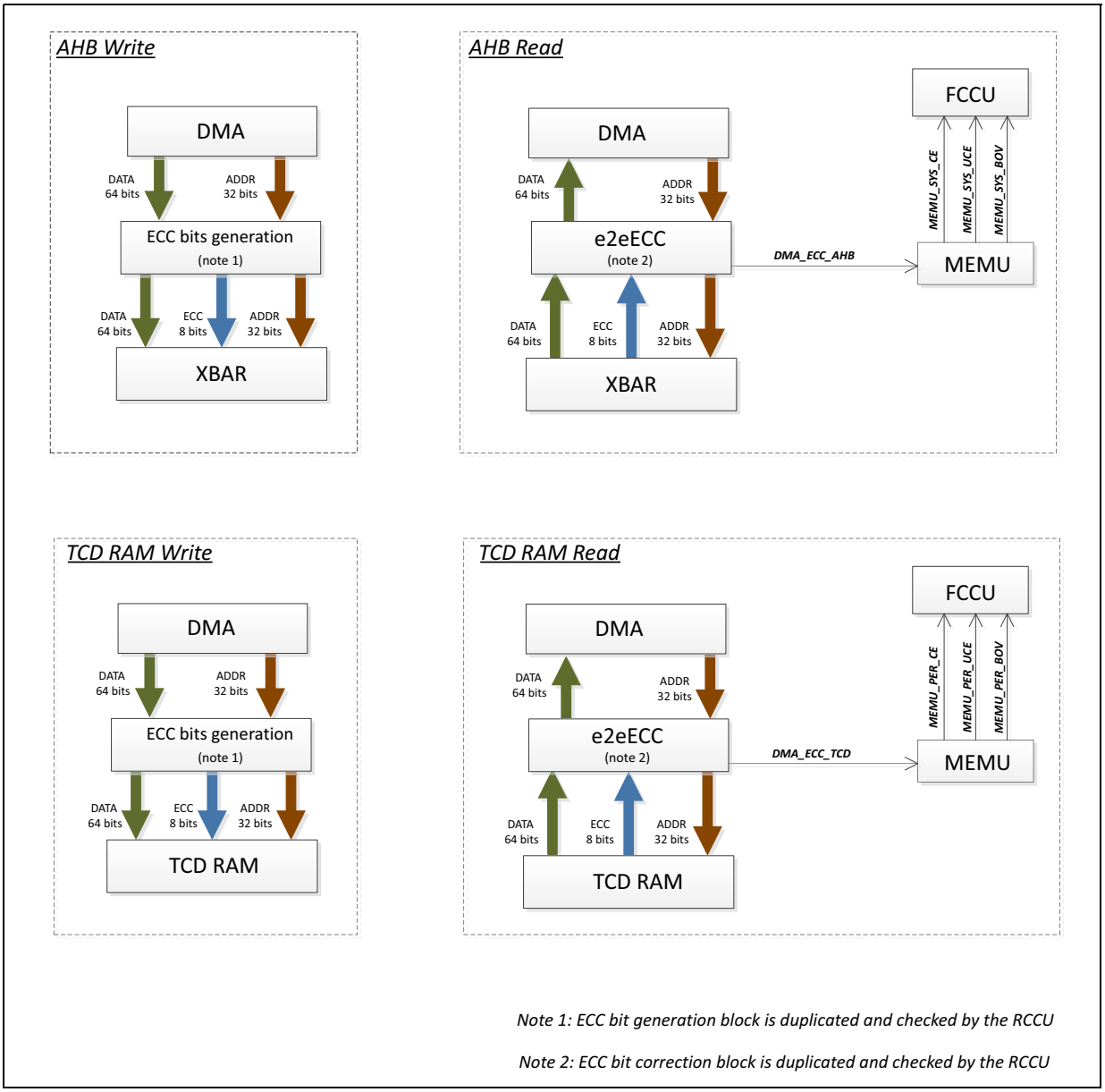
The ECC protection for the DMA is shown in *Figure 20*. The scheme is similar for an AHB transaction towards the XBAR or a local TCD memory transaction.

The e2eECC engines are placed between the DMA IP and the XBAR or TCD RAM.

Upon DMA write operations, the ECC bits are generated. ECC generation faults are covered by replication of the ECC engine.

Upon DMA read operations, the ECC bits are checked and the data eventually corrected, with a reporting to the MEMU. The ECC correction module is also replicated.

Figure 20. ECC protection scheme for DMA



Note 1: ECC bit generation block is duplicated and checked by the RCCU

Note 2: ECC bit correction block is duplicated and checked by the RCCU

6.4.3.11 ECC protection for FlexRay

The ECC protection for the FlexRay IP is shown in *Figure 21*. The ECC engines are embedded into the IP, therefore separated from the external local memories. The FlexRay IP uses local memory to save the data structures used for their operations and the ECC is applied to the local connections to these memories. In addition the FlexRay is a master on the AHB system. The memory error protections are listed in *Table 23*. EDC/ECC errors are reported to both the FCCU and MEMU.

Figure 21. ECC protection scheme for FlexRay

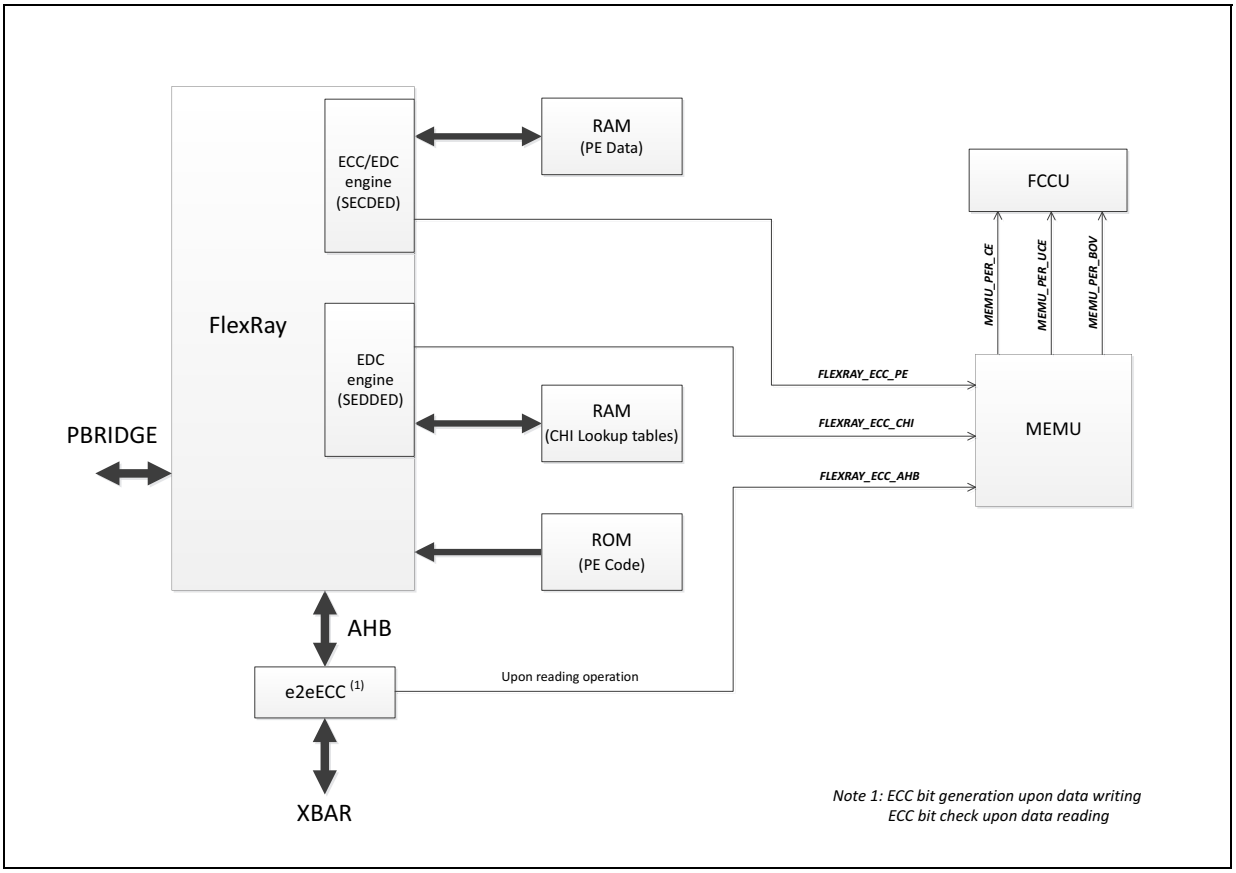


Table 23. FlexRay ECC/EDC protection

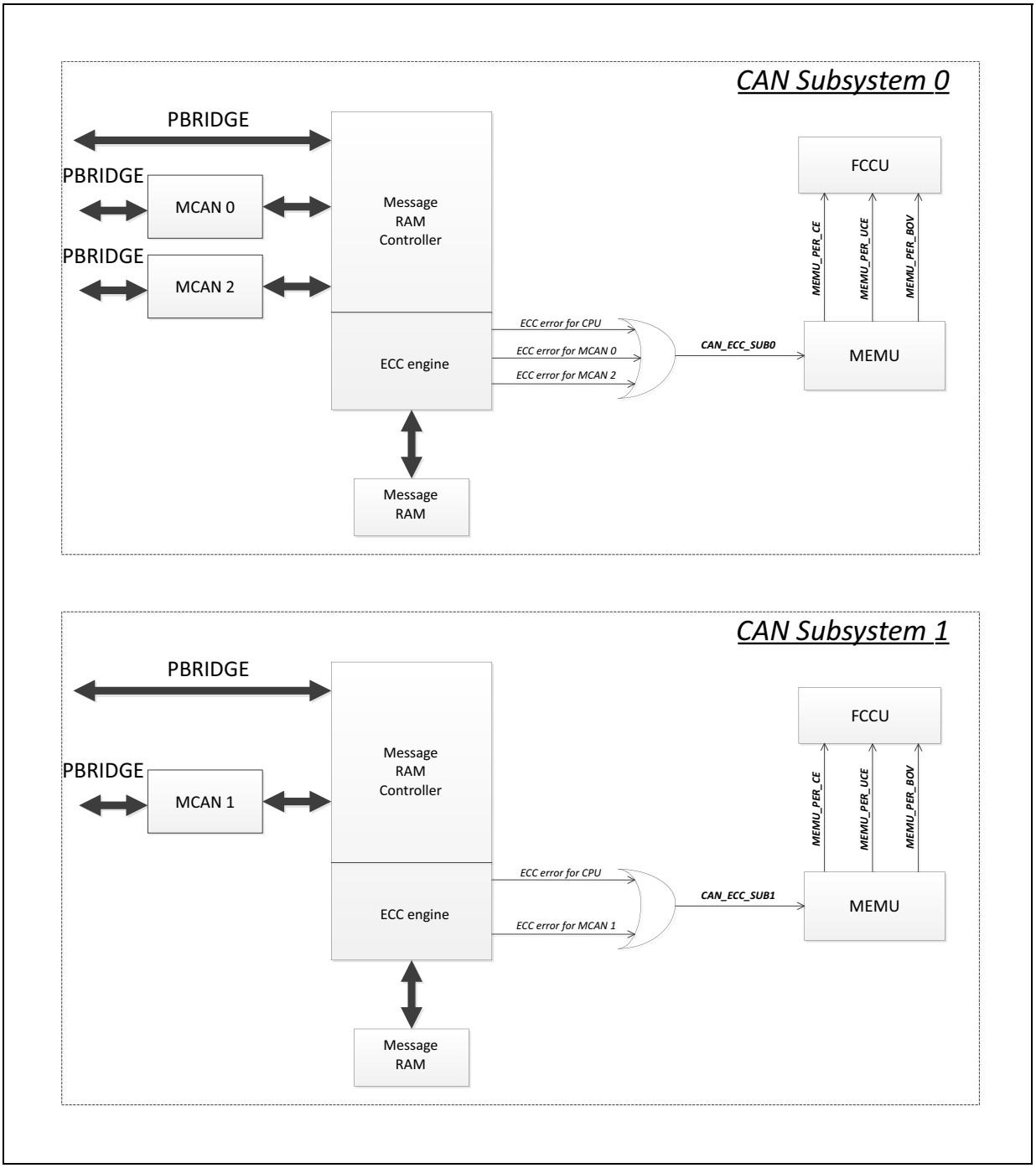
Path	Protection type
FlexRay AHB port	e2eECC on both data and address busses
FlexRay CHI connection to its local data memory	SEDDED on data bus only
FlexRay PE to its local data memory	SECDED on data bus only
PE connection to its local code memory	No protection

The ECC engine towards the FlexRay local memories make sure that if all data and code bits are all 0 or all 1, ECC/EDC do not report a correct word.

6.4.3.12 ECC protection for CAN subsystems

The message RAM access is regulated by the Message RAM Controller, which is part of the CAN subsystem. The ECC protection for the CAN subsystem is shown in *Figure 22*. The Message RAM Controller embeds an ECC engine. An ECC error is reported to the MEMU upon bit errors. The MEMU will in turn trigger the FCCU.

Figure 22. ECC protection scheme for CAN subsystems



The ECC engine towards the MCAN local memory makes sure that if all data and code bits are all 0 or all 1, ECC/EDC do not report a correct word.

6.4.3.13 ECC protection for SSCM configuration data reading

During the reset phase the device configuration data, stored in Test Flash and UTEST Flash, are read by the SSCM. The same ECC engine embedded into the Flash is used and it reports to the SSCM.

6.4.4 I/O safety

All I/O peripherals (except the SIUL2 and “hi-bandwidth” communication controllers such as CAN) used for safety purposes exist at least twice and the members of each pair are connected to different peripheral bridges. For example, DSPI provided by the MCU, two MCU IP modules (DSPI1, DSPI2) are included with different external pin connections. DSPI1 is internally connected to PBRIDGE_0 and DSPI2 to PBRIDGE_1 and is accessible via different addresses.

Redundant usage of I/O peripherals by software is possible.

6.4.5 Clock

The PLL provides error indications to the FCCU.

Clocks are supervised for correct frequency using CMU (Clock Monitoring Units).

CMU status is forwarded to the FCCU.

There is no element (specifically no flip flop) clocked by the supervised clock on the path from a CMU to FCCU and continuing on to the error_out pin.

The CMUs are driven by the IRCOSC to ensure independence from the monitored clocks.

Generally, in order to check the clock source and the clock path for any supervised clock, the CMU module is placed at the end of the clock path.

Modules are not directly clocked by the XOSC.

Diagnostic measures use a different clock to the function they are supervising. Particularly, FCCU and internal SWT run on a different clock than the system clock.

Master and checker channels have separate clock trees that conform as close as possible to the PLL-VCO.

One CMU is connected to the master channel clock path after the branch of the clock line for master and checker channel.

For other clock branches, one CMU is placed after the corresponding clock muxing logic.

CMUs are not required for clocks that can be considered implicitly supervised due to their failures affecting other supervised or diagnosed functionalities.

6.4.6 Power

There are two types of voltage supervisors on SPC574Sx, Low Voltage Detect (LVD) and High Voltage Detect (HVD) monitors. Safety-relevant voltages are supervised for voltages that are out of these ranges. Since safety relevant voltages have the potential to disable the failure indication mechanisms of the MCU (such as FCCU, pads) the indication of these errors can be used to cause a direct transition of the MCU into the safe state (reset assertion). For details on power monitoring, see *Chapter 10: Power management* and *Chapter 51: Power Control Unit (MC_PCU)*.

6.4.7 Failure handling

To prevent a failure of a safety related component (particularly the safety core) from affecting the appropriate reaction to the failure indication, failures are handled by dedicated modules (such as FCCU, MEMU, FOSU) as described below:

An independent module called Failure Collection and Control Unit (FCCU) is responsible for reacting to all failure indications.

For each failure source a different reaction is configurable.

Failure sources include:

- All failure indication signals from modules within the MCU
- Control logic and signals monitored by the FCCU itself (see *Chapter 65: Fault Collection and Control Unit (FCCU)*)
- Software-initiated failure indications. For example, software signals the FCCU that it has evidence of a failure. Software can also directly influence the state of the F*n* pins

Available failure reactions are:

- Assertion of an interrupt (maskable or non-maskable)
- Resetting the MCU
- Changing the state of the failure indication pins, F0 and F1
- Disabling the transmission capabilities of communication controllers

Note: This is only possible in conjunction with changing the state of the failure indication pins.

- No reaction

It is possible for the software to read out the failure source which caused a failure reaction. This failure source storage is not influenced by functional (not Destructive, such as Power-on) resets.

In order to prevent FCCU failures from affecting the overall failure reaction capability, a secondary reaction path exists to supervise the FCCU reactions. This module is called FOSU (FCCU Output Supervisor Unit).

A Memory Error Management Unit (MEMU) provides information on where (which module) and at which address an error in the RAM or Flash was detected.

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

- Single-Bit-Error (SBE) vs. Uncorrectable Error (UCE)
- Address of error (the accessed address)

Note: EDC logic does not report to the MEMU but failure indication is directly forwarded to the FCCU.

MBIST error address information is also stored in the MEMU.

In case an error is detected during MBIST, the following information is transmitted from the MBIST collar to the MEMU:

- Single-Bit-Error vs. Multi-Bit-Error
- Address of error (the tested address)
- Bit position of the error (if a single-bit error)

The MEMU stores the addresses at which ECC errors occurred in the following way:

- Uniqueness of the addresses in the buffer is ensured
- Software is able to read the buffers and remove individual entries from them (for example by marking them as invalid or free)
- If the buffer overflows, an additional signal is sent to the FCCU

The MEMU aggregates ECC failures and forwards them to the FCCU. For each of the SRAM, Peripheral RAM and Flash categories, the MEMU provides the following aggregated signals to the FCCU:

- Single-bit error corrected
- Uncorrectable error detected
- MEMU buffer overflow

Correctable errors which do not cause a new entry into the address table because they are known are not reported to the FCCU.

Uncorrectable errors always cause a report to the FCCU, even if they were already stored in the reporting table.

The following table lists the FCCU failure inputs.

Table 24. FCCU failure inputs

FCCU input	Input trigger name	IP	Description	Default reaction after Reset	HW fault behaviour	Error reaction path check
0	TEMP_ERROR	PMC	Temperature out of range	No reaction	Pulse, no SW intervention needed	—
1	LVD_ERROR		Voltage out of range from LVDs	No reaction	Pulse, no SW intervention needed	selftest by tap point manipulation
2	HVD_ERROR		Voltage out of range from HVDs	No reaction	Pulse, no SW intervention needed	selftest by tap point manipulation
3	DPMC_DCF_SAFETY_ERR		DCF read error (PMC trimming configuration)	No reaction	Persistent till error negation	LBIST
4	MEMORY_DCF_SAFETY_ERR	RAM	DCF read error (RAM redundancy data)	No reaction	Persistent till error negation	LBIST

Table 24. FCCU failure inputs (continued)

FCCU input	Input trigger name	IP	Description	Default reaction after Reset	HW fault behaviour	Error reaction path check
5	SSCM_XFER_FLASH_ERR	SSCM	SSCM transfer error (during DCF reading used to configure the SSCM and the STCU) + Flash memory initialization error	No reaction	SSCM gets cleared by SW. Fatal error pulse no SW needed	Not testable (but wrong configuration can trigger other detection)
6	STCU_UF(1)	STCU2	STCU2 Unrecoverable Fault during the BIST execution, corresponding to the STCU_ERR_STAT.UFSF error bit	Functional Reset	Clear status through FCCU	SW check of BIST MISR
7	STCU_RF(1)		STCU2 Recoverable Fault during the BIST execution, corresponding to the STCU_ERR_STAT.RFSF error bit	No reaction	Clear status through FCCU	SW check of BIST MISR
8	STCU_LMBIST_USR_ERR		STCU2 fault in case LBIST or MBIST control signals go to wrong condition during user application	No reaction	—	STCU activation during User application running
9	JTAG_NPC_SSCM_MON	Debug logic(2)	JTAG, NPC, or SSCM not in reset or activation of dangerous debug functionality	No reaction	Clear status through FCCU	Feedback loop via FCCU Fake bit

Table 24. FCCU failure inputs (continued)

FCCU input	Input trigger name	IP	Description	Default reaction after Reset	HW fault behaviour	Error reaction path check
10	RCCU_CPU_NON_AHB	RCCU	RCCU error trigger for the CPU non- AHB signals	No reaction	Clear status through FCCU	LBIST + Feedback loop via FCCU Fake bit and additional RCCU input
11	RCCU_CPU_AHB		RCCU error trigger for the CPU AHB signals	No reaction	Clear status through FCCU	LBIST + Feedback loop via FCCU Fake bit and additional RCCU input
12	SWT0_ResetReq	SWT	Software watchdog timer, 2nd timeout	Functional Reset	Cleared by reset only	LBIST + SW test by not triggering SWT during startup
13	SWT1_InterruptReq		Software watchdog timer, 1st timeout	No reaction	Needs to be cleared by SW, wc1 in SWT register	LBIST + SW test by not triggering SWT during startup

Table 24. FCCU failure inputs (continued)

FCCU input	Input trigger name	IP	Description	Default reaction after Reset	HW fault behaviour	Error reaction path check
14	RCCU_AIPS_0	RCCU	RCCU error trigger for the AIPS 0 protocol checker	No reaction	Clear status through FCCU	LBIST + Feedback loop via FCCU Fake bit and additional RCCU input
15	RCCU_AIPS_1		RCCU error trigger for the AIPS 1 protocol checker	No reaction	Clear status through FCCU	LBIST + Feedback loop via FCCU Fake bit and additional RCCU input

Table 24. FCCU failure inputs (continued)

FCCU input	Input trigger name	IP	Description	Default reaction after Reset	HW fault behaviour	Error reaction path check
16	MEMU_RAM_CE	MEMU	MEMU System RAM correctable error	No reaction	Clear Status in MEMU	LBIST + Injection of RAM ECC error via ECC bit access and later reading
17	MEMU_RAM_UCE		MEMU System RAM non-correctable error	No reaction	Clear Status in MEMU	LBIST + Injection of RAM ECC error via ECC bit access and later reading
18	MEMU_RAM_BOV		MEMU System RAM overflow(3)	No reaction	Clear Status in MEMU	LBIST + Injection of overflow error indication by MEMU debug register
19	MEMU_PER_CE		MEMU Peripheral RAM correctable error	No reaction	Clear Status in MEMU	LBIST + Injection of correctable error entry by writing one into the MEMU
20	MEMU_PER_UCE		MEMU Peripheral RAM non-correctable error	No reaction	Clear Status in MEMU	LBIST + Injection of uncorrectable error entry by writing one into the MEMU

Table 24. FCCU failure inputs (continued)

FCCU input	Input trigger name	IP	Description	Default reaction after Reset	HW fault behaviour	Error reaction path check
21	MEMU_PER_BOV	MEMU	MEMU Peripheral RAM overflow(2)	No reaction	Clear Status in MEMU	LBIST + Injection of overflow error indication by MEMU debug register
22	MEMU_FLASH_CE		MEMU FLASH correctable Error	No reaction	Clear Status in MEMU	LBIST + Reading preprogrammed Correctable ECC error from UTESTFlash
23	MEMU_FLASH_UCE		MEMU FLASH non-correctable Error	No reaction	Clear Status in MEMU	LBIST + Reading preprogrammed uncorrectable ECC error from UTESTFlash
24	MEMU_FLASH_BOV		MEMU FLASH overflow(2)	No reaction	Clear Status in MEMU	LBIST + Injection of Overflow error indication by MEMU debug register

Table 24. FCCU failure inputs (continued)

FCCU input	Input trigger name	IP	Description	Default reaction after Reset	HW fault behaviour	Error reaction path check
25	RCCU_INTC	RCCU	RCCU error trigger for the INTC	No reaction	Clear status through FCCU	LBIST + Feedback loop via FCCU Fake bit and additional RCCU input
26	RCCU_DMA		RCCU error trigger for the DMA	No reaction	Clear status through FCCU	LBIST + Feedback loop via FCCU Fake bit and additional RCCU input
27	SAR_012_WDG	ADC	SAR0/1/2 watchdog (or-ed signals)	No reaction	SW clear required in ADCDIG	LBIST + ADC conversion outside the wdg range
28	SAR_012_SPV_WDG		Supervisor SAR0/1/2 watchdog	No reaction	SW clear required in ADCDIG	LBIST + ADC conversion outside the wdg range
29	FM_PLL0	PLL	Loss of lock	No reaction	SW clear required in PLLDIG	Forced loss by PLL reconfiguration
30	FM_PLL1		Loss of lock	No reaction	SW clear required in PLLDIG	Forced loss by PLL reconfiguration

Table 24. FCCU failure inputs (continued)

FCCU input	Input trigger name	IP	Description	Default reaction after Reset	HW fault behaviour	Error reaction path check
31	CMU_0_OSC	CGM (clock monitor units)(4)	Loss of XOSC clock	No reaction	Clear Status in CMU	—
32	CMU_0_PLL0		PLL0 frequency out of range	No reaction	Clear Status in CMU	Test by misconfiguration
33	CMU_1		AHB system clock out of range	No reaction	Clear Status in CMU	Test by misconfiguration
34	CMU_2		IPS clock out of range	No reaction	Clear Status in CMU	Test by misconfiguration
35	FLASH_REF_ERR	Flash	Flash Read Reference Error	No reaction	Clear status through FCCU	LBIST

Table 24. FCCU failure inputs (continued)

FCCU input	Input trigger name	IP	Description	Default reaction after Reset	HW fault behaviour	Error reaction path check
36	FLASH_EDC_AFTER_ECC	Flash controller	EDC after ECC for Flash array	No reaction	Clear status through FCCU	LBIST
37	PFLASH_EDC_AFTER_ECC		EDC after ECC for FLASH Controller	No reaction	Clear status through FCCU	LBIST
38	FLASH_ENC_ERR		Flash encoding error(5)	No reaction	Clear status through FCCU	LBIST
39	PFLASH_ADDR_FDBK_ERR		Flash Controller Address Feedback Error	No reaction	Clear status through FCCU	LBIST

Table 24. FCCU failure inputs (continued)

FCCU input	Input trigger name	IP	Description	Default reaction after Reset	HW fault behaviour	Error reaction path check
40	XBAR	XBIC	XBAR transaction monitor mismatch	No reaction	Clear status through FCCU	LBIST
41	PMC VD BIST	PMC	PMC Voltage Detectors BIST (enabled via SW)	No reaction	Clear status through FCCU	LBIST
42	SSCM_ACTIVE	SSCM	Indication of SSCM spurious activation during application run-time	No reaction	Pulse	Feedback loop via FCCU Fake bit
43	ADDR_FDBK_ERR_ PRAM + RAM_LWB_ERR	RAM controller	System RAM Address Feedback + RAM Late-Write Buffer mismatch	No reaction	Persistent till SW clearing from FCCU	LBIST
44	RCCU_DSMC	RCCU	DSMC RCCU checker	No reaction	Clear status through FCCU	LBIST + Feedback loop via FCCU Fake bit and additional RCCU input

Table 24. FCCU failure inputs (continued)

FCCU input	Input trigger name	IP	Description	Default reaction after Reset	HW fault behaviour	Error reaction path check
45	PRAM_EDC_AFTER_ECC	RAM controller	EDC after ECC for RAM controller (read-modify-write ECC)	No reaction	Persistent till SW clearing from FCCU	LBIST
46	DFT1	TCU	Design For Test : Test circuitry Group 1 unexpected activation	No reaction	Persistent till reset	—
47	DFT2		Design For Test : Test circuitry Group 2 unexpected activation	No reaction	Persistent till reset	—
48	DFT3		Design For Test : Test circuitry Group 3 unexpected activation	No reaction	Persistent till reset	—
49	DFT4		Design For Test : Test circuitry Group 4 unexpected activation	No reaction	Persistent till reset	—
50	MACHINE_CHECK_EXC	CPU	Core machine check exception	No reaction	Pulse, no SW intervention needed	LBIST + Willfully cause an appropriate exception (that is, address a non-existing address)
51[6]	Lockstep Mode	SOC	Indication of disablement of Checker Core or RCCUs	No reaction	Persistent till HW negates it (comes out of debug mode)	Feedback loop via FCCU Fake bit

Table 24. FCCU failure inputs (continued)

FCCU input	Input trigger name	IP	Description	Default reaction after Reset	HW fault behaviour	Error reaction path check
52	Safe Mode	MC_ME	Safe Mode entry indication	No reaction	Persistent till system remains in Safe mode	LBIST + (in principle requesting an RGM "safe mode" but that is not typically something want to do)
53	Compensation Disable	IO	Any of the macro-blocks, dedicated to guarantee that the IO control through the PVT (process, voltage, temperature) compensation cell, has been disabled	No reaction	Persistent till error negation from HW	—
54	CMU_3/4	CGM (clock monitor units)(7)	Peripheral clock out of range	No reaction	Clear Status in CMU	Test by misconfiguration
55	AIPS0_EDC	AIPS0	EDC fail	No reaction	Pulse, no SW intervention needed	LBIST
56	AIPS0_ECC_RCCU		ECC bit generation/correction failure	No reaction	Pulse, no SW intervention needed	LBIST

Table 24. FCCU failure inputs (continued)

FCCU input	Input trigger name	IP	Description	Default reaction after Reset	HW fault behaviour	Error reaction path check
57	AIPS1_EDC	AIPS1	EDC fail	No reaction	Pulse, no SW intervention needed	LBIST
58	AIPS1_ECC_RCCU		ECC bit generation/correction failure	No reaction	Pulse, no SW intervention needed	LBIST
59	RCCU_DMA_MUX	RCCU	RCCU error trigger for the DMA Mux	No reaction	—	LBIST + Feedback loop via FCCU Fake bit and additional RCCU input
60	SAR_345_WDG	ADC	SAR3/4/5 watchdog (or-ed signals)	No reaction	—	LBIST + ADC conversion outside the watchdog range
61	SAR_345_SPV_WDG		Supervisor SAR3/4/5 watchdog	No reaction	—	LBIST + ADC conversion outside the watchdog range
62	ECTU0	CTU	CRC Mismatch output to FCCU	No reaction	—	Feedback loop via FCCU Fake bit

Table 24. FCCU failure inputs (continued)

FCCU input	Input trigger name	IP	Description	Default reaction after Reset	HW fault behaviour	Error reaction path check
63	ECTU1	CTU	CRC Mismatch output to FCCU	No reaction	—	Feedback loop via FCCU Fake bit
64	DMA_GASKET_MON_ERROR	Platform	Monitor error from DMA frequency gasket	No reaction	—	Feedback loop via FCCU Fake bit
65	Flexray_GASKET_MON_ERROR	Platform	Monitor error from Flexray frequency gasket	No reaction	—	Feedback loop via FCCU Fake bit
66	AIPS0_GASKET_MON_ERROR	AIPS	Monitor error from AIPS0 frequency gasket	No reaction	—	Feedback loop via FCCU Fake bit

Table 24. FCCU failure inputs (continued)

FCCU input	Input trigger name	IP	Description	Default reaction after Reset	HW fault behaviour	Error reaction path check
67	AIPS1_GASKET_MON_ERROR	AIPS	Monitor error from AIPS1 frequency gasket	No reaction	—	Feedback loop via FCCU Fake bit
68	Flexray EDC after ECC	Platform	Alarm from flexray EDC after ECC	No reaction	—	Feedback loop via FCCU Fake bit
69	PLATFORM/CORE	Platform	CORE 0 D-Cache address ERR	No reaction	—	Feedback loop via FCCU Fake bit
70	PLATFORM/CORE	Platform	CORE 0 D-MEM address ERR	No reaction	—	Feedback loop via FCCU Fake bit

Table 24. FCCU failure inputs (continued)

FCCU input	Input trigger name	IP	Description	Default reaction after Reset	HW fault behaviour	Error reaction path check
71	PLATFORM/CORE	Platform	CORE 0 I-Cache address ERR	No reaction	—	Feedback loop via FCCU Fake bit
72	IMEM Acess monitor	Platform	Imem access monitor error	No reaction	—	Feedback loop via FCCU Fake bit
73	PMC_DIG	PMC	QVF FCCU error	No reaction	—	Feedback loop via FCCU Fake bit
74	PLATFORM/CORE	Platform	FCCU alarm from SMPU	No reaction	—	Feedback loop via FCCU Fake bit
75	PLATFORM/CORE	Platform	Monitor error from SMPU	No reaction	—	Feedback loop via FCCU Fake bit

1. The STCU2 error conditions can be configured as unrecoverable or recoverable. The effect is to trigger two different output triggers, which are are mapped to separate fault inputs of the FCCU.
2. This signal is driven by the “intg_debug” logic and it is asserted if the either TRST or TMS signals are high or if the Nexus or DCI controller report an operative status of the Nexus or debug resources.
3. This signal indicates that the correctable, non-correctable, or overflow error triggers are asserted at the same time. This signal is the result of a logical “OR” of “memu_xxxx_corr_err_oflw_flag”, “memu_xxxx_uncorr_err_oflw_flag”, and “memu_xxxx_err_buf_oflw_flag” done within the “intg_fccu” glue logic, put between the MEMU and the FCCU

4. For each CMU, the output alarm signals go through the “intg_fccu” where they are OR-ed together into one single FCCU input fault trigger.
5. The Flash re-calculate the ECC bits and compare them with the one received by the Flash controller. This signal is triggered in case of mismatch. The signal is also triggered by a mismatch between the latched address and the one rebuilt inside the Flash to address the memory array.
6. Since lockstep operation is not guaranteed in debug and stop mode, lockstep is disabled. When the debugger is connected to device this FCCU fault may set. It is suggested to Ignore this fault when operating in lockstep during debug mode.
7. For each CMU, the output alarm signals go through the “intg_fccu” where they are OR-ed together into one single FCCU input fault trigger.

6.4.8 Failure indication

Failure of the MCU is signaled via two pins: F0 and F1.

There are different configurable protocols to indicate a failure via the one or two pins.

At least one of the failure indication pins is high to indicate “operational”. This is true independent of the way the error output pins are configured to indicate a failure.

In case a two-pin bi-stable protocol with different outputs on both pins is implemented, the operational high pin is configurable.

In case of Time Switching protocol the default toggling frequency is less than 100 Hz.

Failure indication via the pin(s) is controlled by the FCCU.

After power-on the error out pins either have high impedance or are in a state that indicates an error. They go to “operational” state only on software request.

In case of a failure event or on software request for error indication, the pin(s) are set to “failure” for a minimum time T_min, even if software tries to release it before.
T_min = 250 µs + delta_T, with delta_T configurable by software up to 10 ms.

In case another failure event occurs within T_min after a previous failure, the “minimal failure indication time” is restarted.

Functional reset does not influence the failure indication state.

At the onset of a “failure” indication on an error pin, the duration of the low state since the last trigger is available to the system.

After the timer has expired (timer > T_min), software can set the error pin back to high.

Software can activate error pins to verify the integrity of the reaction path.

6.4.9 Latent fault detection

SPC574Sx supports some mechanisms to detect faults which can become latent. Two important mechanisms dedicated to this functionality are LBIST and MBIST.

If a safety measure cannot be tested by LBIST, a mechanism is included whereby software can detect its (in)correct function.

The LBIST (or other measures for all logic) targets at least 90% coverage of direct current (d.c.) faults at gate level for digital logic. (See ISO 26262-5 table D.1 for definition of d.c. faults). Coverage can be limited to 60% for peripheral modules.

If one of the built-in selftests fails, either hardware or software prevents normal operation from starting. Failure in this context means:

- Any error detected by the LBIST
- MBIST reports any multi-bit error or more than report the fault table depth single-bit
- Any error in any other selftest module including the selftest controller itself

Self tests can be executed during startup or during operation.

When detection is used, the availability of a safety mechanism and particularly of the corresponding fault reaction path is testable by software via “fake fault” injections following one of the following approaches:

- Memory content manipulation (for example, for Flash and RAM)
- Threshold manipulation (for example, for CMU, LVD/HVDs, SWT)
- Bit-level fault injection (for example, for RCCUs, protocol checkers, EDC, MEMU)

6.4.10 Common Cause Failures (CCFs)

Various prevention and/or detection measures have been included to cover CCFs. These include supervision of configuration registers, SW malfunction, internal signals related to test and debug, temperature and voltage, physical separation of modules on the die.

Configuration registers which can affect the execution of the MCU’s safety function and simultaneously disable the safety mechanism due to their change, are protected against bit flips or those flips are detected by independent measures.

Such registers are also protected against accidental software writes (software register protection).

Failures of control logic/signals do not cause similar failures in both channels or deactivate safety measures. At least the following logic/signals belong to this class:

- Debug circuitry (including on-die JTAG generators)
- Test circuitry
- Mode selection

Pads for redundant signals are distributed over the pad ring as far as possible to avoid redundant signal inputs/outputs to be adjacent. The user will avoid configurations that place redundant signals on neighboring pads or pins.

6.4.11 Resets

A transient single point fault (specifically a single-bit flip in a register) can not cause a reset of an individual module (that is a partial reset).

Note: Resetting the whole MCU is acceptable.

There is a measure to detect a loop of functional resets and resolve it by causing a destructive escalation.

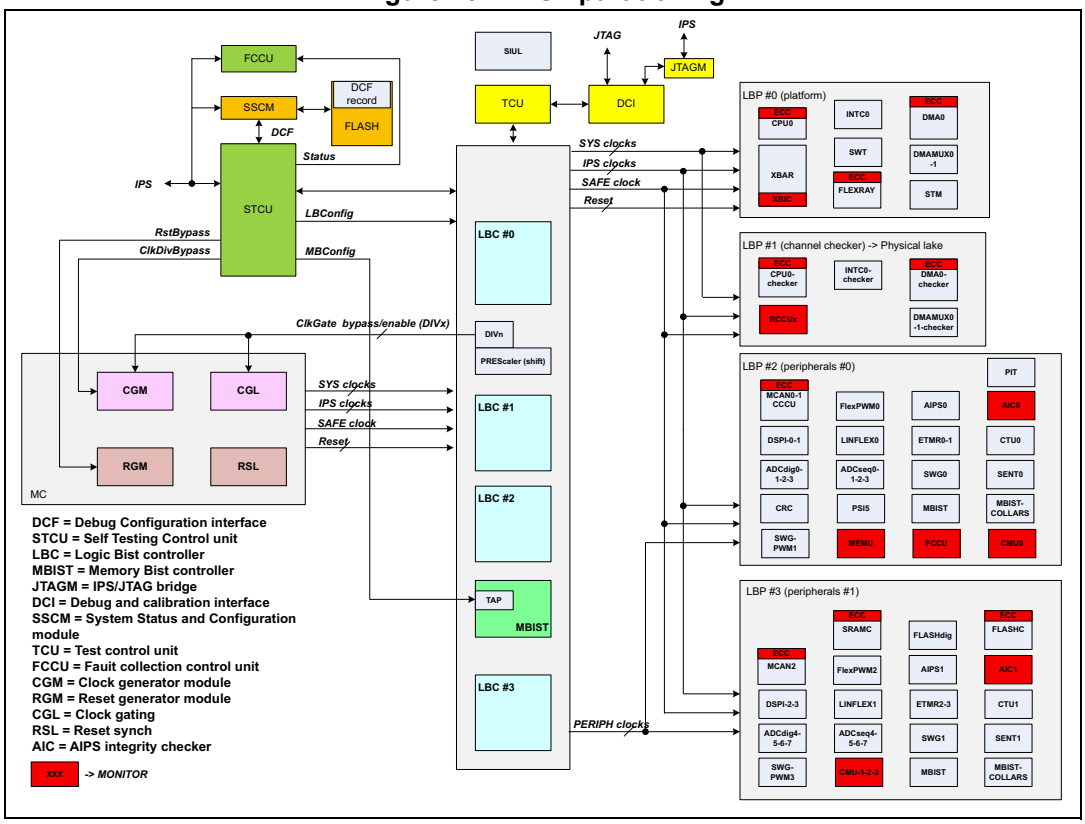
There is be a measure to detect a loop of destructive resets and keep the device in reset (for example by not releasing the reset signal) if detected.

Software is capable of resetting the device even if the IRCOSC has failed.

6.4.12 LBIST / MBIST architecture

The Logic BIST partitioning is provided in *Figure 23*.

Figure 23. LBIST partitioning



The Memory BIST partitioning is provided in *Table 25* :

Table 25. MBIST partitioning

M_BIST	Partition	start_address	end_address
0	dmem_0	0x5000_0000	0x5000_7FFF
1	dmem_1	0x5000_8000	0x5000_FFFF
2	icache_0	not addressable	
3	icache_1	not addressable	
4	icache_2	not addressable	
5	icache_3	not addressable	
6	itag	not addressable	
7	dma_ram	not addressable	
8	fray_dram	not addressable	
9	fray_lram-0	not addressable	
10	fray_lram-1	not addressable	
11	mcan_0	not addressable	

Table 25. MBIST partitioning (continued)

M_BIST	Partition	start_address	end_address
12	mcan_1	not addressable	
13	sram	0x4000_0000	0x4001_7FFF
14	ram_mpflash	0xFC02_0000	0xFC02_3FFF
15	dcache-0	not addressable	
16	dcache-1	not addressable	
17	dcache-2	not addressable	
18	dcache-3	not addressable	
19	dtag	not addressable	
20	bam_rom	not addressable	

The settings of LBIST / MBIST configurations described as preliminary data in *Table 26* are provided *Section 7.11.6.1: BIST Configuration* of *Chapter 7: Device configuration*.

Note: All configurations can run both in offline and in online mode. The reader shall follow the configuration indicated below and take care about choosing the relevant STCU registers depending on online or offline mode

Process time considers the IRC as frequency source in offline and XOSC in online mode (time at nominal source clock frequency)

The power consumption data are enclosed in the final device datasheet release

Table 26. LBIST / MBIST configurations(1)

Number	Description	BIST	Mode	Frequency settings	Number of patterns	Process time (ms)	Configuration(2)
1	All MBISTs in parallel mode	MBIST	Parallel	40 MHz (PLL)	—	26.00	Mbist Conf_1_offline
				128 MHz (PLL)	—	9.00	Mbist Conf_1_online
2	All LBISTs in parallel mode	LBIST	Parallel	35 MHz (PLL)	5K (0x1388)	13.2	Lbist Conf_2_offline Conf_2_online

Table 26. LBIST / MBIST configurations(1) (continued)

Number	Description	BIST	Mode	Frequency settings	Number of patterns	Process time (ms)	Configuration(2)
3	Full self test (LBIST + MBIST)	LBIST	Parallel	16 MHz (RC)	5K (0x1388)	30.00	Full Selftest Conf_3_offline Conf_3_online
		MBIST		40 MHz (PLL)	—	26.00	Mbist Conf_3_offline
				128 MHz (PLL)	—	9.00	Mbist Conf_3_online
4	Full self test (LBIST + MBIST)	LBIST	Sequential	35 MHz (PLL)	5K (0x1388)	50.00	Full Selftest Conf_4_offline Conf_4_online
		MBIST			—	88.00	
5	Full self test (LBIST + MBIST)	LBIST	L0//L1 + L2//L3	35 MHz (PLL)	5K (0x1388)	27.00	Full Selftest Conf_5_offline Conf_5_online
		MBIST	Parallel		—	30.00	

1. The timing has been measured by programming only STCU related DCF clients. Timing may change based on actual DCF clients programmed.
2. For detail about the configuration see *Section 7.11.6.1: BIST Configuration* of *Chapter 7: Device configuration*.

6.5 Register classifications

- Safety Critical – Capable of causing a violation of the Safety Goal and disabling the safety mechanism at the same time
- Safety Latent – Capable of disabling a safety mechanism
- Safety Relevant – Capable of causing a violation of the Safety Goal
- Not Safety Relevant – Does not fall into any of the other three classifications

7 Device configuration

7.1 Introduction

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

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

7.2 Core modules

The SPC574Sx has two z4 cores that perform various computational and control functions. The main core is a e200z420n3 core and is used for general computational functions. The second core, referred to as the Checker Core, uses an e200z419 core which is a true subset of the e200z420n3 core. When enabled, the Checker Core operates in lock step mode with the Main Core executing exactly the same instructions as the Main Core. Thus, the Checker Core checks to insure that the Main Core is executing correctly.

For more information, please see *Chapter 12: Core e200z420n3 Description*.

7.2.1 Core Reset Settings

Table 27 shows the state of the *PowerISA 2.06* architected registers and other optional resources immediately following a system reset.

Table 27. Reset settings for Z resources

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

Table 27. Reset settings for Z resources (continued)

Resource	System reset setting
DDAM	0x0000_0000(3)
DDEAR	Unaffected(2)
DEAR	Unaffected(2)
DEVENT	0x0000_0000(3)
DSRR0	Unaffected(2)
DSRR1	Unaffected(2)
DVC1	Unaffected(2)
DVC2	Unaffected(2)
ESR	0x0000_0000
HID0	0x0000_0000
HID1	0x0000_0000
IAC1	0x0000_0000(3)
IAC2	0x0000_0000(3)
IAC3	0x0000_0000(3)
IAC4	0x0000_0000(3)
IAC5	0x0000_0000(3)
IAC6	0x0000_0000(3)
IAC7	0x0000_0000(3)
IAC8	0x0000_0000(3)
IVPR	Unaffected(2)
LR	Unaffected(2)
L1CFG0, L1CFG1(4)	—
L1CSR0, 1	0x0000_0000
L1FINV0, 1	0x0000_0000
MAS1	Unaffected(2)
MAS2	Unaffected(2)
MAS3	Unaffected(2)
MAS4	Unaffected(2)
MCAR	Unaffected(2)
MCSR	0x0000_0000
MCSRR0	Unaffected(2)
MCSRR1	Unaffected(2)
MMUCFG(4)	—
MPUCSR0	0x0000_0000
MPU0CFG(4)	—

Table 27. Reset settings for Z resources (continued)

Resource	System reset setting
MSR	0x0000_0000
NPIDR	0x0000_0000
PID0	0x0000_0000
PIR	0x0000_0000
PVR(4)	0x815D_0000
SPEFSCR	0x0000_0000
SPRG0	Unaffected(2)
SPRG1	Unaffected(2)
SPRG2	Unaffected(2)
SPRG3	Unaffected(2)
SRR0	Unaffected(2)
SRR1	Unaffected(2)
SVR(4)	0x0000_0000
XER	0x0000_0000

1. The levels that determine these bits are set by the factory and may change between revisions of the device.
2. Undefined on **m_por** assertion, unchanged on **p_reset_b** assertion
3. Reset by processor reset **p_reset_b** if DBCR0[EDM]=0, as well as unconditionally by **m_por**.
4. Read-only registers

7.2.2 Special Purpose Register Summary

PowerISA 2.06 and implementation-specific SPRs for the e200z420n3 and e200z419 cores are listed in *Table 28*. All registers are 32 bits in size. Register bits are numbered from bit 0 to bit 31 (most-significant to least-significant). Shaded entries represent optional registers. An SPR register may be read or written with the **mfspr** and **mtspr** instructions. In the instruction syntax, compilers should recognize the mnemonic name given in *Table 28*.

Table 28. Special Purpose Registers

Mnemonic	Name	SPR Number	Access	Privileged	e200z Specific
XER	Integer Exception Register	1	R/W	No	No
LR	Link Register	8	R/W	No	No
CTR	Count Register	9	R/W	No	No
SRR0	Save/Restore Register 0	26	R/W	Yes	No
SRR1	Save/Restore Register 1	27	R/W	Yes	No
PID0	Process ID Register	48	R/W	Yes	No
CSRR0	Critical Save/Restore Register 0	58	R/W	Yes	No
CSRR1	Critical Save/Restore Register 1	59	R/W	Yes	No

Table 28. Special Purpose Registers (continued)

Mnemonic	Name	SPR Number	Access	Privileged	e200z Specific
DEAR	Data Exception Address Register	61	R/W	Yes	No
ESR	Exception Syndrome Register	62	R/W	Yes	No
IVPR	Interrupt Vector Prefix Register	63	R/W	Yes	No
SPRG0	SPR General 0	272	R/W	Yes	No
SPRG1	SPR General 1	273	R/W	Yes	No
SPRG2	SPR General 2	274	R/W	Yes	No
SPRG3	SPR General 3	275	R/W	Yes	No
PIR	Processor ID Register	286	R/W	Yes	No
PVR	Processor Version Register	287	Read-only	Yes	No
DBSR	Debug Status Register	304	Read/Clear(1)	Yes	No
DBCR0	Debug Control Register 0	308	R/W	Yes	No
DBCR1	Debug Control Register 1	309	R/W	Yes	No
DBCR2	Debug Control Register 2	310	R/W	Yes	No
IAC1	Instruction Address Compare 1	312	R/W	Yes	No
IAC2	Instruction Address Compare 2	313	R/W	Yes	No
IAC3	Instruction Address Compare 3	314	R/W	Yes	No
IAC4	Instruction Address Compare 4	315	R/W	Yes	No
DAC1	Data Address Compare 1	316	R/W	Yes	No
DAC2	Data Address Compare 2	317	R/W	Yes	No
DVC1	Data Value Compare 1	318	R/W	Yes	No
DVC2	Data Value Compare 2	319	R/W	Yes	No
SPEFSCR	LSP/EFP APU status and control register	512	R/W	No	No
L1CFG0	L1 cache config register 0	515	Read-only	No	Yes
L1CFG1	L1 cache config register 1	516	Read-only	No	Yes
NPIDR	Nexus 3 Process ID register	517	R/W	No	Yes
DBCR3	Debug control register 3	561	R/W	Yes	Yes
DBCNT	Debug Counter register	562	R/W	Yes	Yes
DBCR4	Debug control register 4	563	R/W	Yes	Yes
DBCR5	Debug control register 5	564	R/W	Yes	Yes
IAC5	Instruction Address Compare 5	565	R/W	Yes	Yes
IAC6	Instruction Address Compare 6	566	R/W	Yes	Yes
IAC7	Instruction Address Compare 7	567	R/W	Yes	Yes
IAC8	Instruction Address Compare 8	568	R/W	Yes	Yes
DBERC0	Debug external resource control register 0	569	Read-only	Yes	Yes

Table 28. Special Purpose Registers (continued)

Mnemonic	Name	SPR Number	Access	Privileged	e200z Specific
MCSRR0	Machine Check Save/Restore Register 0	570	R/W	Yes	Yes
MCSRR1	Machine Check Save/Restore Register 1	571	R/W	Yes	Yes
MCSR	Machine Check Syndrome Register	572	Read/Clear(2)	Yes	Yes
MCAR	Machine Check Address Register	573	R/W	Yes	Yes
DSRR0	Debug save/restore register 0	574	R/W	Yes	Yes
DSRR1	Debug save/restore register 1	575	R/W	Yes	Yes
DDAM	Debug Data Acquisition Messaging register	576	R/W	No	Yes
DAC3	Data Address Compare 3	592	R/W	Yes	Yes
DAC4	Data Address Compare 4	593	R/W	Yes	Yes
DBCR7	Debug control register 7	596	R/W	Yes	Yes
DBCR8	Debug control register 8	597	R/W	Yes	Yes
DDEAR	Debug Data Effective Address register	600	R/W	Yes	Yes
DBCR6	Debug control register 6	603	R/W	Yes	Yes
MAS0	MPU assist register 0	624	R/W	Yes	Yes
MAS1	MPU assist register 1	625	R/W	Yes	Yes
MAS2	MPU assist register 2	626	R/W	Yes	Yes
MAS3	MPU assist register 3	627	R/W	Yes	Yes
MPU0CFG	MPU0 configuration register	692	Read-only	Yes	Yes
L1FINV1	L1 cache flush and invalidate control register 0	959	R/W	Yes	Yes
DEVENT	Debug Event register	975	R/W	No	Yes
HID0	Hardware implementation dependent reg 0	1008	R/W	Yes	Yes
HID1	Hardware implementation dependent reg 1	1009	R/W	Yes	Yes
L1CSR0	L1 cache control and status register 0	1010	R/W	Yes	Yes
L1CSR1	L1 cache control and status register 1	1011	R/W	Yes	Yes
BUCSR	Branch Unit Control and Status Register	1013	R/W	Yes	Yes
MPUCSR0	MPU configuration register	1014	R/W	Yes	Yes
MMUCFG	MMU/MPU configuration register	1015	Read-only	Yes	Yes
L1FINV0	L1 cache flush and invalidate control register 0	1016	R/W	Yes	Yes
SVR	System Version Register	1023	Read-only	Yes	Yes

1. The Debug Status Register can be read using *mfspr RT,DBSR*. The Debug Status Register cannot be directly written to. Instead, bits in the Debug Status Register corresponding to ‘1’ bits in GPR(RS) can be cleared using *mtspr DBSR,RS*.
2. The Machine Check Syndrome Register can be read using *mfspr RT,MCSR*. The Machine Check Syndrome Register cannot be directly written to. Instead, bits in the Machine Check Syndrome Register corresponding to ‘1’ bits in GPR(RS) can be cleared using *mtspr MCSR,RS*.

7.3 System modules

7.3.1 SIUL2 configuration

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

For details please refer *Chapter 14: System Integration Unit Lite2 (SIUL2)*.

7.3.2 Crossbar switch

This device contains one crossbar switch module.

7.3.2.1 Crossbar switch (XBAR) configuration

This section summarizes how the module has been configured in the chip. For a comprehensive description of the module itself, see *Chapter 13: Crossbar switch (XBAR)*.

Figure 24. Computational shell's crossbar switch integration

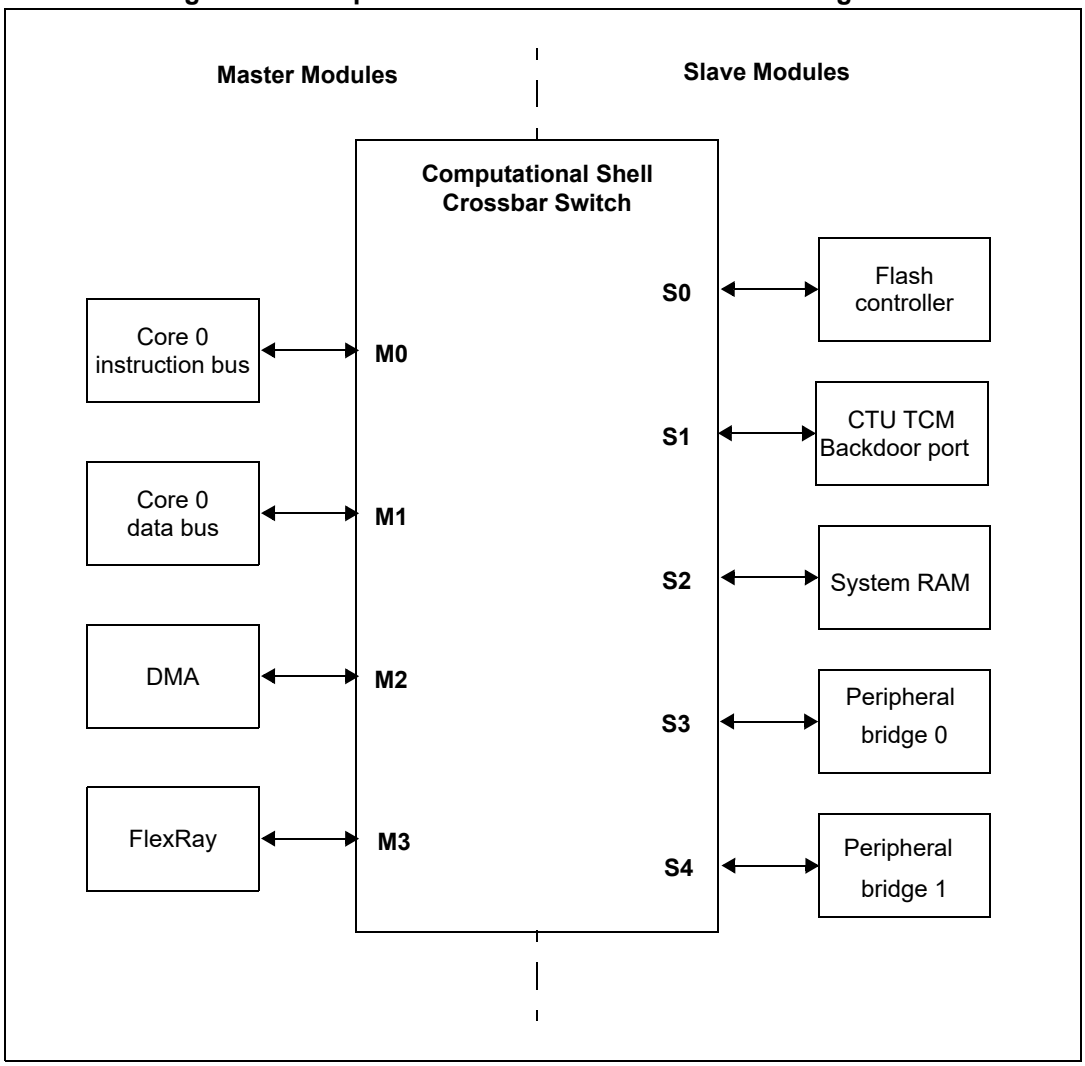


Table 29. Reference links to related information

Topic	Related module	Reference
Full description	Crossbar switch	*Chapter 13: Crossbar switch (XBAR)*
Clocking	—	*Chapter 24: Clocking*
Crossbar switch master	Cores	*Section 7.2: Core modules*
	DMA	*Chapter 20: Enhanced Direct Memory Access (eDMA)*
Crossbar switch slave	Flash memory controller	*Section 3.2.2: Flash memory*
	System RAM	*Section 3.2.1: SRAM*
System memory map	—	*Chapter 5: Memory map*

7.3.2.1.1 Crossbar switch master assignments

This device contains three masters connected to the crossbar switch.

Table 30 contains the master assignments.

Table 30. Crossbar switch master assignments

Master module	Physical master port number
Core_0 instruction bus	0
Core_0 data bus	1
DMA	2
FlexRay	3

7.3.2.1.2 Crossbar switch slave assignments

This device contains five slaves connected to the crossbar switch.

Table 31 contains the slave assignments.

Table 31. Crossbar switch slave assignments

Slave module	Physical slave port number
Flash memory controller port 0	0
CPU TCM backdoor	1
RAM controller	2
PBRIDGE_0	3
PBRIDGE_1	4

7.3.3 System Memory Protection Unit (SMPU) configuration

SPC574Sx contains one instance of the System Memory Protection Unit (SMPU). The following sections explain how SMPU_0 is configured on this chip.

7.3.3.1 SMPU_0 configuration

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

Figure 25. SMPU_0 configuration

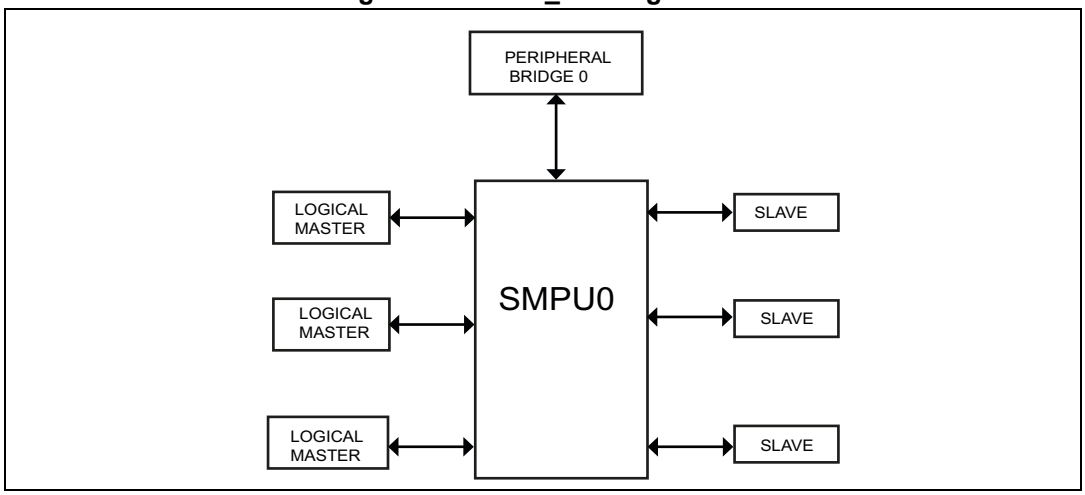


Table 32. Reference links to related information

Topic	Related module	Reference
Full description	System Memory Protection Unit (SMPU)	*Chapter 17: System Memory Protection Unit (SMPU)*
System memory map	—	*Chapter 5: Memory map*
Clocking	—	*Chapter 24: Clocking*
Power management	—	*Chapter 10: Power management*
Logical masters	—	*Section 7.3.3.2: SMPU_0 logical bus master assignments*
Region descriptors	—	*Section 7.3.3.3: Number of region descriptors supported by SMPU_0*
RGD_WORD2 formats	—	*Section 7.3.3.4: RGD_WORD2 formats supported by SMPU_0*

7.3.3.2 SMPU_0 logical bus master assignments

Table 33 contains the logical bus master assignments for the SMPU_0.

Table 33. SMPU_0 logical bus master assignments

SMPU logical bus master number	Bus master
0	Core_0
3	DMA
5	FlexRay
8	Core_0 Debug

7.3.3.3 Number of region descriptors supported by SMPU_0

This instance of the SMPU module supports up to eight region descriptors.

7.3.3.4 RGD_WORD2 formats supported by SMPU_0

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

7.3.3.5 Master Access permissions for SMPU_0

The only master allowed to access the SMPU_0 registers is Core_0. It is also required that the core be in supervisor mode.

7.3.4 Peripheral bridge configuration

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

Figure 26. Peripheral bridge configuration

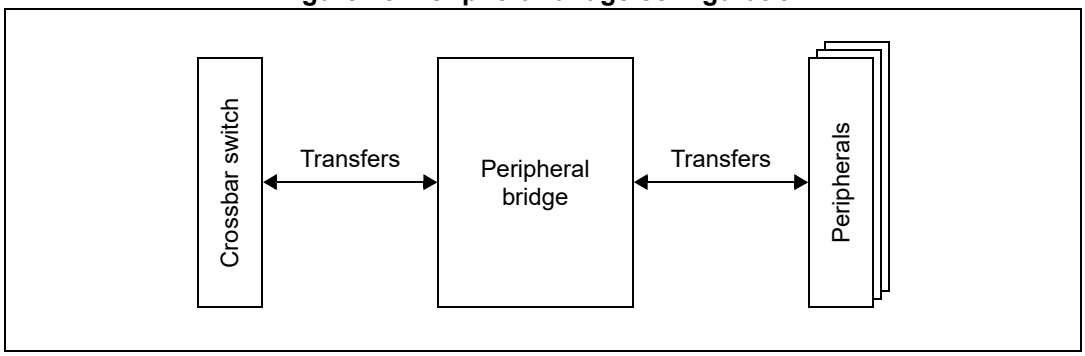


Table 34. Reference links to related information

Topic	Related module	Reference
Full description	Peripheral bridge	*Chapter 16: Peripheral bridge*
System memory map	—	*Chapter 5: Memory map*
Clocking	—	*Chapter 24: Clocking*
Crossbar switch	Crossbar switch	*Section 7.3.2: Crossbar switch*

7.3.4.1 Peripheral bridge instantiation

This device contains two identical peripheral bridges: PBRIDGE_0 and PBRIDGE_1.

7.3.4.2 Memory maps

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

7.3.4.3 MPR registers

Each of the two peripheral bridges supports up to eight crossbar switch masters, each assigned to a MPROT*n* field in the MPRA registers. However, fewer are supported on this device. See *Section 7.3.2: Crossbar switch* for details of the master port assignments for this device.

7.3.4.4 PACR/OPACR registers

Each of the two peripheral bridges support up to 320 peripherals, each assigned to a PACR*n* or OPACR*n* field within the PACRA-H/OPACRA-AF registers. However, fewer peripherals are supported on this device. See *Chapter 5: Memory map* for details of the peripheral slot assignments for this device. Unused PACR*n* fields are reserved.

7.3.5 Interrupt controller (INTC) configuration

7.3.5.1 INTC configurable parameters

The INTC on this chip supports 1 processor.

7.3.5.2 INTC implemented registers

Table 35 shows the registers implemented on the INTC on this chip.

Table 35. INTC implemented registers

Address offset	Register
0x0000	INTC Block Configuration Register (INTC_BCR)
0x0004	INTC Master Protection Register (INTC_MPROT)
0x0010	INTC Current Priority Register for Processor 0 (INTC_CPR0)
0x0020	INTC Interrupt Acknowledge Register for Processor 0 (INTC_IACKR0)
0x0030	INTC End Of Interrupt Register for Processor 0 (INTC_EOIR0)
0x0040–0x005C	INTC Software Set/Clear Interrupt Register 0_3 (INTC_SSCIR0_3) – INTC Software Set/Clear Interrupt Register 28_31 (INTC_SSCIR28_31)
0x0060–0x085E	INTC Priority Select Register 0_1 (INTC_PSR0_1) – INTC Priority Select Register 1022_1023 (INTC_PSR1022_1023)

7.3.5.3 Interrupt channel map

Table 36 shows the list of interrupt channels.

Table 36. Interrupt channel map

Number	Description	Register bit / Signal name
0 - 31	Software interrupt flag 0 - 31	INTC_SSCIR0-7[SET/CLR0-3]
32 - 33	Reserved	—
34	Watchdog timer	SWT_IR[TIF]
35 - 43	Reserved	—
44	Platform periodic timer (STM)	STM_CIR0[CIF]
45	Platform periodic timer (STM)	STM_CIR1[CIF]
46	Platform periodic timer (STM)	STM_CIR2[CIF]
47	Platform periodic timer (STM)	STM_CIR3[CIF]

Table 36. Interrupt channel map (continued)

Number	Description	Register bit / Signal name
48 - 51	Reserved	—
52	eDMA Error	OR-ed DMA_INTL[INT0 – INT31]
53 - 84	eDMA Channel 0 - 31	DMA_INTL[INT0 – INT31]
85 - 184	Reserved	—
185	Flash controller Prog/Erase/Suspend IRQ_0	MCR[DONE]
186 - 225	Reserved	—
226 - 233	Periodic Interrupt Timer 0 - 3	PIT_TFLG0-7[TIF]
234 - 241	Reserved	—
242	XOSC counter	XOSC_CTL[I_OSC]
243	SIUL External Interrupt 0	SIUL2_DISR0[EIF0] (EXT_INT 0 signal)
244	SIUL External Interrupt 1	SIUL2_DISR0[EIF1] (EXT_INT 1 signal)
245	SIUL External Interrupt 2	SIUL2_DISR0[EIF2] (EXT_INT 2 signal)
246	SIUL External Interrupt 3	SIUL2_DISR0[EIF3] (OR-ed EXT_INT 3-5 signals)
246 - 250	Reserved	—
251	Magic Carpet Module - ME 0	ME_IS[I_SAFE]
252	Magic Carpet Module - ME 1	ME_IS[I_MTC]
253	Magic Carpet Module - ME 2	ME_IS[I_IMODE]
254	Magic Carpet Module - ME 3	ME_IS[I_ICONF]
255	Magic Carpet Module - RGM 0	MC_RGM Functional and destructive reset alternate event interrupt
256 - 258	Reserved	—
259	DSPI0_0	DSPI_0_SR[TFUF]:OR: DSPI_0_SR[RFOF]:OR: DSPI_0_SR[TFIWF]
260	DSPI0_1	DSPI_0_SR[EOQF]
261	DSPI0_2	DSPI_0_SR[TFFF]
262	DSPI0_3	DSPI_0_SR[TCF]
263	DSPI0_4	DSPI_0_SR[RFDF]
264	DSPI0_5	DSPI_0_SR[CMD_TCF]
265	DSPI0_6	DSPI_0_SR[CMDFFF]
266	DSPI0_7	DSPI_0_SR[SPEF]
267	Reserved	—
268	DSPI1_0	DSPI_1_SR[TFUF]:OR: DSPI_1_SR[RFOF]:OR: DSPI_1_SR[EOQF]

Table 36. Interrupt channel map (continued)

Number	Description	Register bit / Signal name
269	DSPI1_1	DSPI_1_SR[EOQF]
270	DSPI1_2	DSPI_1_SR[TFFF]
271	DSPI1_3	DSPI_1_SR[TCF]
271	DSPI1_4	DSPI_1_SR[RFDF]
273	DSPI1_5	DSPI_1_SR[CMD_TCF]
274	DSPI1_6	DSPI_1_SR[CMDFFF]
275	DSPI1_7	DSPI_1_SR[SPEF]
276	Reserved	—
277	DSPI2_0	DSPI_2_SR[TFUF]:OR: DSPI_2_SR[RFOF]:OR: DSPI_2_SR[TFIWF]
278	DSPI2_1	DSPI_2_SR[EOQF]
279	DSPI2_2	DSPI_2_SR[TFFF]
280	DSPI2_3	DSPI_2_SR[TCF]
281	DSPI2_4	DSPI_2_SR[RFDF]
282	DSPI2_5	DSPI_2_SR[CMD_TCF]
283	DSPI2_6	DSPI_2_SR[CMDFFF]
284	DSPI2_7	DSPI_2_SR[SPEF]
285	Reserved	—
286	DSPI3_0	DSPI_3_SR[TFUF]:OR: DSPI_3_SR[RFOF]:OR: DSPI_3_SR[TFIWF]
287	DSPI3_1	DSPI_3_SR[EOQF]
288	DSPI3_2	DSPI_3_SR[TFFF]
289	DSPI3_3	DSPI_3_SR[TCF]
290	DSPI3_4	DSPI_3_SR[RFDF]
291	DSPI3_5	DSPI_3_SR[CMD_TCF]
292	DSPI3_6	DSPI_3_SR[CMDFFF]
293	DSPI3_7	DSPI_3_SR[SPEF]
294 - 375	Reserved	—
376	LINFlexD/eSCI0_0	LINFlexD_0_RXI
377	LINFlexD/eSCI0_1	LINFlexD_0_TXI
378	LINFlexD/eSCI0_2	LINFlexD_0_ERR
379 - 383	Reserved	—
384	LINFlexD/eSCI1_0	LINFlexD_1_RXI

Table 36. Interrupt channel map (continued)

Number	Description	Register bit / Signal name
385	LINFlexD/eSCI1_1	LINFlexD_1_TXI
386	LINFlexD/eSCI1_2	LINFlexD_1_ERR
387 - 391	Reserved	—
392	LINFlexD/eSCI2_0	LINFlexD_2_RXI
393	LINFlexD/eSCI2_1	LINFlexD_2_TXI
394	LINFlexD/eSCI2_2	LINFlexD_2_ERR
395 - 399	Reserved	—
400	LINFlexD/eSCI3_0	LINFlexD_3_RXI
401	LINFlexD/eSCI3_1	LINFlexD_3_TXI
402	LINFlexD/eSCI3_2	LINFlexD_3_ERR
387 - 452	Reserved	—
453	FlexRay	LRNEIF:OR: DNREIF
454	FlexRay	LRCEIF:OR: DRCEIF
455	FlexRay	FNEAIF
456	FlexRay	FNEBIF
457	FlexRay	WUPIF
458	FlexRay	PRIF
459	FlexRay	CHIF
460	FlexRay	TBIF
461	FlexRay	RBIF
462	FlexRay	MIF
463 - 476	Reserved	—
477	Power Monitor Unit	OR-ed GR_S voltage detector flags
478	Power management Unit (temperature sensor)	OR-ed EPR_TD temperature detector flags
479	Reserved	—
480	PLL_0_0	PLL0SR[LOLF]
481	Reserved	—
482	PLL_1_0	PLL1SR[LOLF]
483	Reserved	—
484	PLL_0_1	PLL0SR[EXTPDF]
485	PLL_1_1	PLL1SR[EXTPDF]
486	PLL_0_2	PLL0SR[CLKSW]
487	PLL_1_2	PLL1SR[CLKSW]

Table 36. Interrupt channel map (continued)

Number	Description	Register bit / Signal name
488	FCCU_0	irq_alarm
489	FCCU_1	irq_misc
490	FCCU IRQ	irq_eout
491- 527	Reserved	—
528	SAR_ADC0	SAR_ADC0
529	SAR_ADC1	SAR_ADC1
530	SAR_ADC2	SAR_ADC2
531	SAR_ADC3	SAR_ADC3
532	SAR_ADC4	SAR_ADC4
533	SAR_ADC5	SAR_ADC5
534 - 542	Reserved	—
543	SAR_ADCB0 (supervisor)	SAR_ADCB0
544	SAR_ADCB1 (supervisor)	SAR_ADCB1
545 - 557	Reserved	—
558	SENT 0	SENT combined fast (ipi_int_sent_global_rx_fast)
559	SENT 0	SENT combined slow (ipi_int_sent_global_rx_serial)
560	SENT 0	SENT combined error (ipi_int_sent_global_rx_err)
561	SENT 1	SENT combined fast (ipi_int_sent_global_rx_fast)
562	SENT 1	SENT combined slow (ipi_int_sent_global_rx_serial)
563	SENT 1	SENT combined error (ipi_int_sent_global_rx_err)
564	SENT 0	Channel 0 fast (ipi_int_sent_rx_fast[0])
565	SENT 0	Channel 0 slow (ipi_int_sent_rx_serial[0])
566	SENT 0	Channel 0 error (ipi_int_sent_rx_err[0])
567	SENT 0	Channel 1 fast (ipi_int_sent_rx_serial[1])
568	SENT 0	Channel 1 slow (ipi_int_sent_rx_fast[1])
569	SENT 0	Channel 1 error (ipi_int_sent_rx_err[1])

Table 36. Interrupt channel map (continued)

Number	Description	Register bit / Signal name
570	SENT 0	Channel 2 fast (ipi_int_sent_rx_serial[2])
571	SENT 0	Channel 2 slow (ipi_int_sent_rx_fast[2])
572	SENT 0	Channel 2 error (ipi_int_sent_rx_err[2])
573 - 578	Reserved	—
579	SENT 1	Channel 0 fast (ipi_int_sent_rx_fast[0])
580	SENT 1	Channel 0 slow (ipi_int_sent_rx_serial[0])
581	SENT 1	Channel 0 error (ipi_int_sent_rx_err[0])
582	SENT 1	Channel 1 fast (ipi_int_sent_rx_fast[1])
583	SENT 1	Channel 1 slow (ipi_int_sent_rx_serial[1])
584	SENT 1	Channel 1 error (ipi_int_sent_rx_err[1])
585	SENT 1	Channel 2 fast (ipi_int_sent_rx_fast[2])
586	SENT 1	Channel 2 slow (ipi_int_sent_rx_serial[2])
587	SENT 1	Channel 2 error (ipi_int_sent_rx_err[2])
588 - 673	Reserved	—
674	JTAGM	JTAGM_SR[SPU_INT]:OR: JTAGM_SR[IDLE]
675	JDC	JDC_MSR[JIN_INT]:OR: JDC_MSR[JOUT_INT]
676 - 685	Reserved	—
686	MCAN 0	MCAN_INT_0
687	MCAN 0	MCAN_INT_1
688	MCAN 1	MCAN_INT_0
689	MCAN 1	MCAN_INT_1
690	MCAN 2	MCAN_INT_0
691	MCAN 2	MCAN_INT_1
692 - 705	Reserved	—
706 - 711	eTimer 0 Channel 0 - 5	STS register (1) for channels 0 - 5
712 - 713	Reserved	—

Table 36. Interrupt channel map (continued)

Number	Description	Register bit / Signal name
714	eTimer 0 Channel 0 Watchdog	STS.WDF for channel 0
715	eTimer 0 Redundancy Check	STS.RCF bit
716	Reserved	—
717 - 722	eTimer 1 Channel 0 - 5	STS register (1) for channels 0 - 5
723 – 724	Reserved	—
725	eTimer 1 Channel 0 Watchdog	STS.WDF for channel 0
726	eTimer 1 Redundancy Check	STS.RCF bit
727	Reserved	—
728 - 733	eTimer 2 Channel 0 - 5	STS register (1) for channels 0 - 5
734 – 735	Reserved	—
736	eTimer 2 Channel 0 Watchdog	STS.WDF for channel 0
737	eTimer 2 Redundancy Check	STS.RCF bit
738	Reserved	—
739 - 744	eTimer 3 Channel 0 - 5	STS register (1) for channels 0 - 5
745 – 746	Reserved	—
747	eTimer 3 Channel 0 Watchdog	STS.WDF for channel 0
748	eTimer 3 Redundancy Check	STS.RCF bit
749 - 799	Reserved	—
800	CTU 0	MRS event
801	CTU 0	ADC command
802	CTU 0	IEE
803 - 810	CTU 0	T0…T7 events
811 - 818	CTU 0	FIFO0…FIFO7
819 - 829	Reserved	—
830	CTU 1	MRS event
831	CTU 1	ADC command
832	CTU 1	IEE
833 - 840	CTU 1	T0…T7 events
841 - 848	CTU 1	FIFO0…FIFO7
849 - 859	Reserved	—
860	FlexPWM 0	Fault
861 - 864	FlexPWM 0	Input capture channel 0…3
865 - 868	FlexPWM 0	Output compare channel 0…3
869 - 872	FlexPWM 0	Reload channel 0…3

Table 36. Interrupt channel map (continued)

Number	Description	Register bit / Signal name
873	FlexPWM 0	Reload error
874	Reserved	—
875 - 876	FlexPWM 1	Input capture channel 0…1
877 - 878	FlexPWM 1	Output compare channel 0…1
879 - 880	FlexPWM 1	Reload channel 0…1
881	FlexPWM 1	Reload error
882	FlexPWM 2	Fault
883 - 886	FlexPWM 2	Input capture channel 0…3
887 - 890	FlexPWM 2	Output compare channel 0…3
891 - 894	FlexPWM 2	Reload channel 0…3
895	FlexPWM 2	Reload error
896	Reserved	—
897 - 898	FlexPWM 3	Input capture channel 0…1
899 - 900	FlexPWM 3	Output compare channel 0…1
901 - 902	FlexPWM 3	Reload channel 0…1
903	FlexPWM 3	Reload error
904 - 934	Reserved	—
935	CCCU 0	—
936	CCCU 1	—
937 - 964	Reserved	—
965	PWM Synchronizer	Interrupt event
966 - 1023	Reserved	—

1. Except WDF and RCF bits.

7.3.5.4 Interrupt latency

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

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

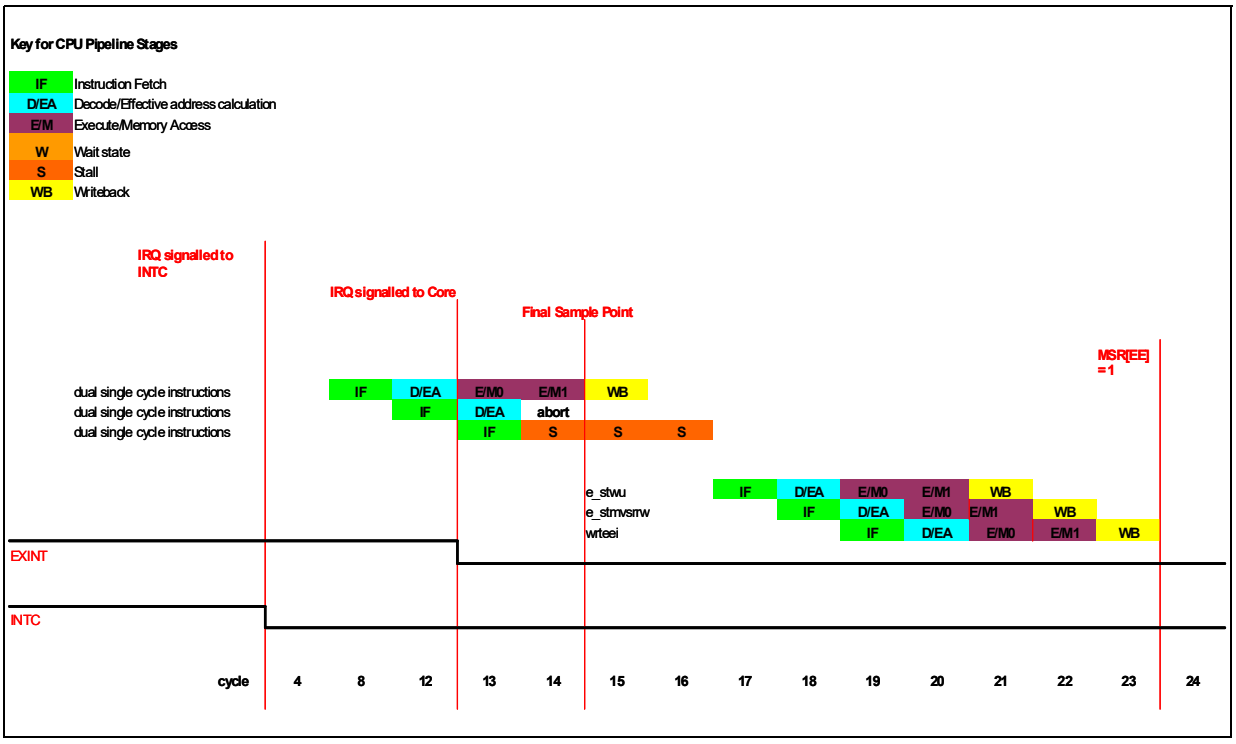
In general, the latency time is divided into three subsections:

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

For the SPC574Sx, the INTC operates at 70 MHz and the CPU operates at 140 MHz. Interrupt latency is:

INTC Latency = 43 ns
CPU Latency = 28.6 ns
Software Latency = 50 ns
Interrupt Latency Total = 121.6 ns

Figure 27. Interrupt timing



7.3.6 DMA controller configuration

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

Figure 28. DMA controller configuration

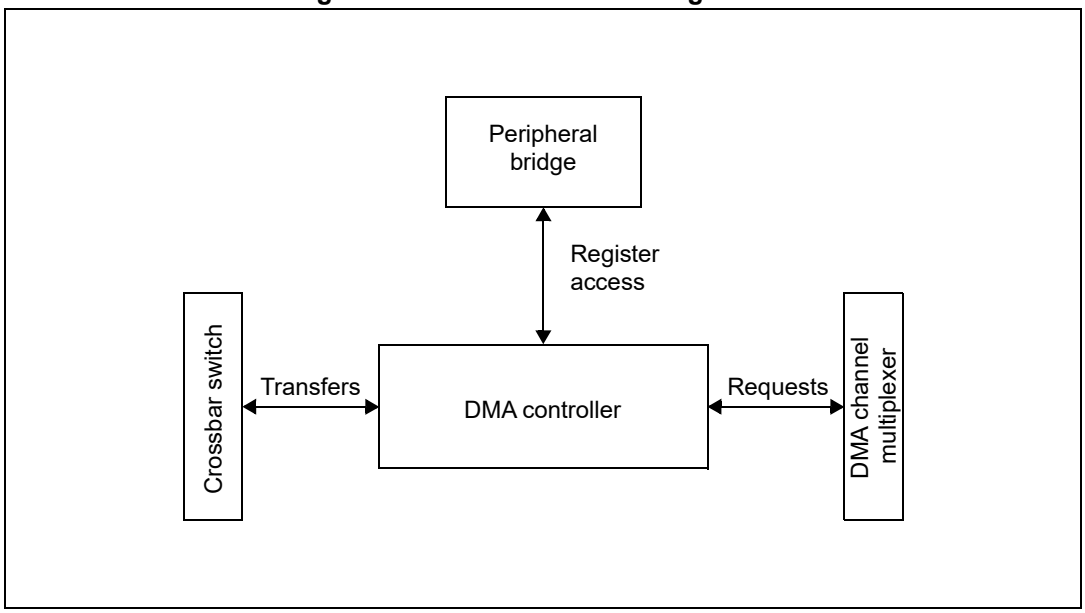


Table 37. Reference links to related information

Topic	Related module	Reference
Full description	DMA controller	*Chapter 20: Enhanced Direct Memory Access (eDMA)*
System memory map	—	*Chapter 5: Memory map*
Register access	Peripheral bridge	*Chapter 16: Peripheral bridge*
Clocking	—	*Chapter 24: Clocking*
Transfers	Crossbar switch	*Chapter 13: Crossbar switch (XBAR)*

7.3.7 DMA channel multiplexer (DMACHMUX) configuration

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

7.3.7.1 DMA channel assignment

SPC574Sx has two multiplexers. *Table 38* lists which channels apply to each multiplexer.

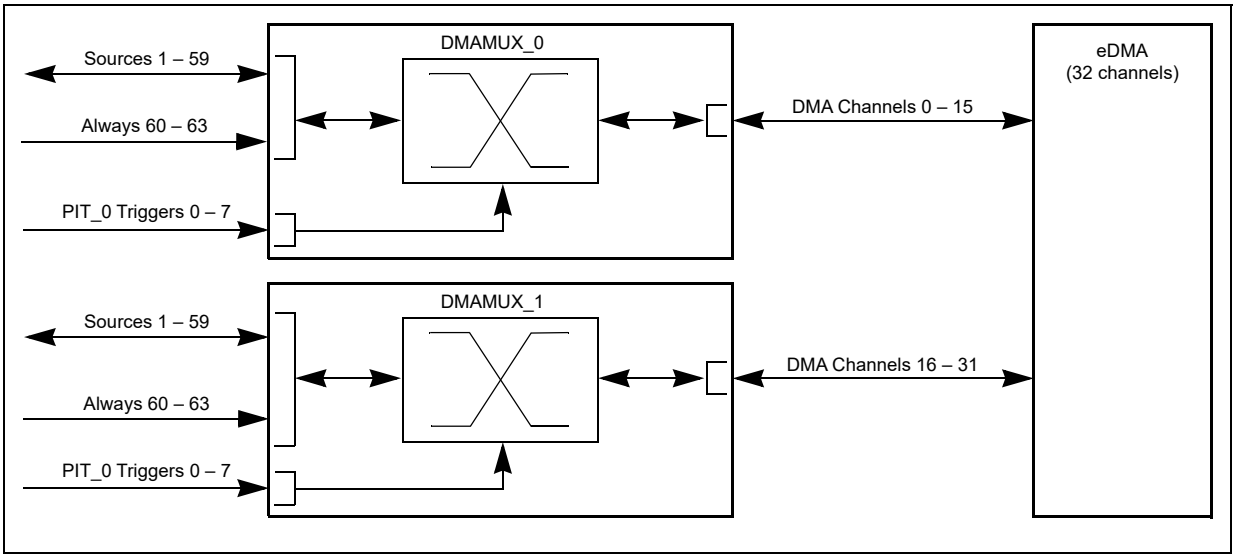
Table 38. SPC574Sx multiplexer and channel list

DMACHMUX instance	DMA channel
0	0–15
1	16–31

7.3.7.2 Block diagram

The connections of the DMA channel multiplexers to the eDMA are shown in *Figure 29*. One DMACHMUX is instantiated for every sixteen DMA channels.

Figure 29. DMA channel multiplexing



7.3.7.3 DMA request mapping

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

Table 39. DMAMUX peripheral DMA request to input source mapping

Source	Peripheral DMA requests			
	DMAMUX_0		DMAMUX_1	
	Peripheral	Event	Peripheral	Event
0	Reserved		Reserved	
1	DSPI 0	Transmit FIFO is full (DSPI_SR[TFFF])	DSPI 2	Transmit FIFO is full (DSPI_SR[TFFF])
2	DSPI 0	Receive FIFO is empty (DSPI_SR[RFDF])	DSPI 2	Receive FIFO is empty (DSPI_SR[RFDF])
3	DSPI 0	Command FIFO is full (DSPI_SR[CMDFFF_RE])	DSPI 2	Command FIFO is full (DSPI_SR[CMDFFF_RE])
4	DSPI 1	Transmit FIFO is full (DSPI_SR[TFFF])	DSPI 3	Transmit FIFO is full (DSPI_SR[TFFF])
5	DSPI 1	Receive FIFO is empty (DSPI_SR[RFDF])	DSPI 3	Receive FIFO is empty (DSPI_SR[RFDF])
6	DSPI 1	Command FIFO is full (DSPI_SR[CMDFFF_RE])	DSPI 3	Command FIFO is full (DSPI_SR[CMDFFF_RE])
7	Reserved		Reserved	
8	Reserved		Reserved	
9	LinFlexD 0	RX	LinFlexD 1	RX
10	LinFlexD 0	TX	LinFlexD 1	TX
11	LinFlexD 3	RX	LinFlexD 2	RX

Table 39. DMAMUX peripheral DMA request to input source mapping (continued)

Source	Peripheral DMA requests			
	DMAMUX_0		DMAMUX_1	
	Peripheral	Event	Peripheral	Event
12	LinFlexD 3	TX	LinFlexD 2	TX
13	SIUL2	REQ0	SIUL2	REQ3
14	SIUL2	REQ1	SIUL2	REQ4
15	SIUL2	REQ2	SIUL2	REQ5
16	Reserved		Reserved	
17	Reserved		Reserved	
18	Reserved		Reserved	
19	Reserved		Reserved	
20	eTimer 0	DREQ0	eTimer 2	DREQ0
21	eTimer 0	DREQ1	eTimer 2	DREQ1
22	Reserved		Reserved	
23	Reserved		Reserved	
24	eTimer 1	DREQ0	eTimer 3	DREQ0
25	eTimer 1	DREQ1	eTimer 3	DREQ1
26	Reserved		Reserved	
27	Reserved		Reserved	
28	SENT 0	FAST	SENT 1	FAST
29	SENT 0	SLOW	SENT 1	SLOW
30	Reserved		Reserved	
31	Reserved		Reserved	
32	FlexPWM 0	Read register	FlexPWM 2	Read register
33	FlexPWM 0	Write register	FlexPWM 2	Write register
34	FlexPWM 1	Read register	FlexPWM 3	Read register
35	FlexPWM 1	Write register	FlexPWM 3	Write register
36	Reserved		Reserved	
37	Reserved		Reserved	
38	CTU 0	REQ0	CTU 1	REQ0
39	CTU 0	REQ1	CTU 1	REQ1
40	CTU 0	REQ2	CTU 1	REQ2
41	CTU 0	REQ3	CTU 1	REQ3
42	CTU 0	REQ4	CTU 1	REQ4
43	CTU 0	REQ5	CTU 1	REQ5

Table 39. DMAMUX peripheral DMA request to input source mapping (continued)

Source	Peripheral DMA requests			
	DMAMUX_0		DMAMUX_1	
	Peripheral	Event	Peripheral	Event
44	CTU 0	REQ6	CTU 1	REQ6
45	CTU 0	REQ7	CTU 1	REQ7
46	CTU 0	GLOBAL	CTU 1	GLOBAL
47	Reserved		Reserved	
48	Reserved		Reserved	
49	FCCU	DMA trigger	Reserved	
50	Reserved		Reserved	
51	Reserved		Reserved	
52	Reserved		Reserved	
53	ADC 0	EOC	ADC 3	EOC
54	ADC 1	EOC	ADC 4	EOC
55	ADC 2	EOC	ADC 5	EOC
56	ADC 0/1/2 Supervisor	EOC	ADC 3/4/5 Supervisor	EOC
57	Reserved		Reserved	
58	MCAN 0	DMA trigger (Debug over CAN)	MCAN 1	DMA trigger (Debug over CAN)
59	MCAN 2	DMA trigger (Debug over CAN)	Reserved	
60	Always on		Always on	
61	Always on		Always on	
62	Always on		Always on	
63	Always on		Always on	

Note: Multiple CHCONFIG registers within the same instance or among multiple instances must not select the same source value.

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

Table 40. PIT_0 periodic trigger to DMA channel mapping

PIT_0 periodic trigger	DMA channel
0	0, 16
1	1, 17
2	2, 18
3	3, 19

Table 40. PIT_0 periodic trigger to DMA channel mapping (continued)

PIT_0 periodic trigger	DMA channel
4	4, 20
5	5, 21
6	6, 22
7	7, 23

7.3.7.4 Platform Configuration Module (PCM)

The Platform Configuration Module (PCM) contains one register. This PCM register is used for controlling different aspects of the Platform functionality including the Frequency Gasket Configurations of the Intelligent Bus Bridging Gaskets.

The PCM module is mapped to PBRIDGE_0 on platform slot 10. Its base address is 0xFC02_8000.

7.3.7.5 Platform Configuration Module 0 register (PCM0)

Offset: 0x0000 Access: Supervisor read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	MSTR_STALL_NXMC_OVF_FLEXRAY_B	MSTR_STALL_NXMC_OVF_DMA_B	EN_HSIZE_MUX_PRAM	PRE_PBRIDGE1_GSKT
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	BRE_PBRIDGE1_GSKT	BWE_PBRIDGE1_GSKT	PRE_PBRIDGE0_GSKT	BRE_PBRIDGE0_GSKT	BWE_PBRIDGE0_GSKT	PRE_FLEXRAY_GSKT	BRE_FLEXRAY_GSKT	BWE_FLEXRAY_GSKT	PRE_DMA_GSKT	BRE_DMA_GSKT	BWE_DMA_GSKT	IPS_CGM_INTC0	IPS_CGM_PRAM0	IPS_CGM_PFLASH0	IPS_CGM_SMPU0	IPS_CGM_XBAR0
W																
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Figure 30. Platform Configuration Module 0 register (PCM0)

Table 41. PCM0 field descriptions

Field	Description
12 MSTR_STALL_NXMC _OVF_FLEXRAY_B	This bit controls the stalling of FLEXRAY Master when trace buffer in NXMC is full. 0 Master is stalled when NXMC overflow occurs 1 Master is not stalled when NXMC overflow occurs
13 MSTR_STALL_NXMC _OVF_DMA_B	This bit controls the stalling of DMA Master when trace buffer in NXMC is full. 0 Master is stalled when NXMC overflow occurs 1 Master is not stalled when NXMC overflow occurs
14 EN_HSIZE_MUX _PRAM	DMC operations HSIZE control for PRAM_0 Force HSIZE=32 bit for Decorated Memory operations as decorated operations compute write checkbits (ECC) on 32-bit word. 0 Any decorated write to SRAM peripherals will be done with native HSIZE of the instruction (8b/16b/32b) 1 Any decorated write to SRAM peripherals will be done with HSIZE=32bit
15 PRE_PBRIDGE1_GSKT	Pending read enable (PRE) for PBRIDGE_1 IAHB Gasket This bit controls the bus gasket's handling of pending read transactions. 0 Pending reads are disabled 1 Pending reads are enabled
16 BRE_PBRIDGE1_GSKT	Burst read enable (BRE) for PBRIDGE_1 IAHB Gasket This bit controls the bus gasket's handling of burst read transactions. 0 Burst reads are converted into a series of single transactions on the slave side of the gasket 1 Burst reads are optimized for best system performance
17 BWE_PBRIDGE1_GSKT	Burst write enable (BWE) for PBRIDGE_1 IAHB Gasket. This bit controls the bus gasket's handling of burst write transactions. 0 Burst writes are converted into a series of single transactions on the slave side of the gasket 1 Burst writes are optimized for best system performance
18 PRE_PBRIDGE0_GSKT	Pending read enable (PRE) for PBRIDGE_0 IAHB Gasket This bit controls the bus gasket's handling of pending read transactions. 0 Pending reads are disabled 1 Pending reads are enabled
19 BRE_PBRIDGE0_GSKT	Burst read enable (BRE) for PBRIDGE_0 IAHB Gasket This bit controls the bus gasket's handling of burst read transactions. 0 Burst reads are converted into a series of single transactions on the slave side of the gasket 1 Burst reads are optimized for best system performance
20 BWE_PBRIDGE0_GSKT	Burst write enable (BWE) for PBRIDGE_0 IAHB Gasket This bit controls the bus gasket's handling of burst write transactions. 0 Burst writes are converted into a series of single transactions on the slave side of the gasket 1 Burst writes are optimized for best system performance

Table 41. PCM0 field descriptions (continued)

Field	Description
21 PRE_FLEXRAY_GSKT	Pending read enable (PRE) for FLEXRAY IAHB Gasket This bit controls the bus gasket's handling of pending read transactions. 0 Pending reads are disabled 1 Pending reads are enabled
22 BRE_FLEXRAY_GSKT	Burst read enable (BRE) for FLEXRAY IAHB Gasket This bit controls the bus gasket's handling of burst read transactions. 0 Burst reads are converted into a series of single transactions on the slave side of the gasket 1 Burst reads are optimized for best system performance
23 BWE_FLEXRAY_GSKT	Burst write enable (BWE) for FLEXRAY IAHB Gasket This bit controls the bus gasket's handling of burst write transactions. 0 Burst writes are converted into a series of single transactions on the slave side of the gasket 1 Burst writes are optimized for best system performance
24 PRE_DMA_GSKT	Pending read enable (PRE) for DMA IAHB Gasket This bit controls the bus gasket's handling of pending read transactions. 0 Pending reads are disabled 1 Pending reads are enabled
25 BRE_DMA_GSKT	Burst read enable (BRE) for DMA IAHB Gasket This bit controls the bus gasket's handling of burst read transactions. 0 Burst reads are converted into a series of single transactions on the slave side of the gasket 1 Burst reads are optimized for best system performance
26 BWE_DMA_GSKT	Burst write enable (BWE) for DMA IAHB Gasket This bit controls the bus gasket's handling of burst write transactions. 0 Burst writes are converted into a series of single transactions on the slave side of the gasket 1 Burst writes are optimized for best system performance
27 IPS_CGM_INTC0	IPS clock gating enable for INTC 0 Disable clock gating 1 Enable clock gating, one IPS clock delay is introduced between master and slave
28 IPS_CGM_PRAM0	IPS clock gating enable for PRAM 0 Disable clock gating 1 Enable clock gating, one IPS clock delay is introduced between master and slave
29 IPS_CGM_PFLASH0	IPS clock gating enable for PFLASHC 0 Disable clock gating 1 Enable clock gating, one IPS clock delay is introduced between master and slave

Table 41. PCM0 field descriptions (continued)

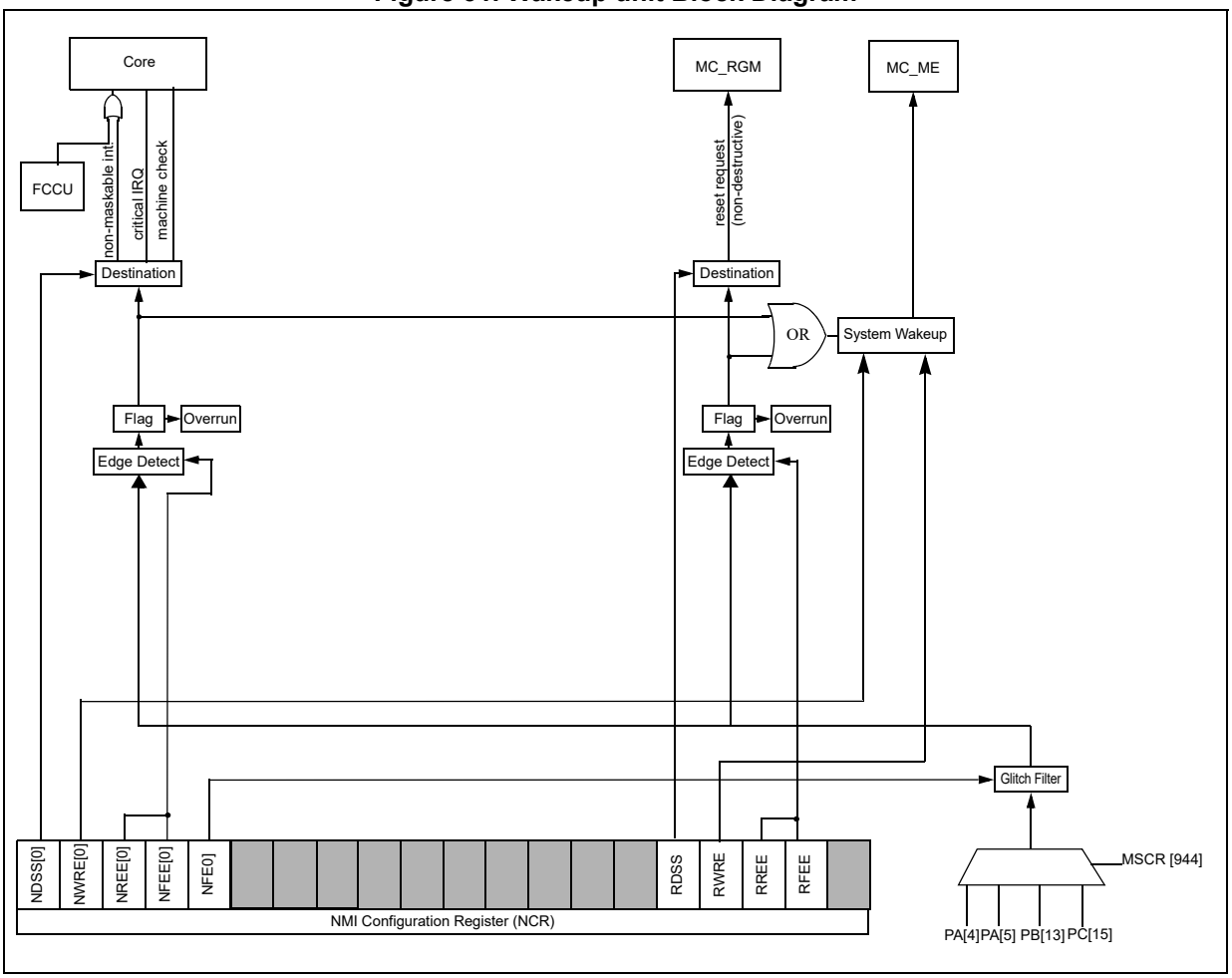
Field	Description
30 IPS_CGM_SMPU0	IPS clock gating enable for SMPU 0 Disable clock gating 1 Enable clock gating, one IPS clock delay is introduced between master and slave
31 IPS_CGM_XBAR0	IPS clock gating enable for XBAR 0 Disable clock gating 1 Enable clock gating, one IPS clock delay is introduced between master and slave

7.3.8 Wakeup unit (WKPU) configuration

The WKPU supports 4 NMI sources. Please refer to *Chapter 22: Wakeup unit (WKPU)* for details.

In *Figure 31: Wakeup unit Block Diagram*, it has to be noted that only one filter is implemented.

Figure 31. Wakeup unit Block Diagram



NMI[0] and NMI[3] sources are tied together while NMI[1] and NMI[2] sources are tied to '1' (refer to *Section 22.2.2.1: NMI Status Flag Register (NSR)*).

Although access to all the bits in the *Section 22.2.2.2: NMI Configuration Register (NCR)* is read/write, writing is only effective for the bits corresponding to NMI[0] and NMI[3].

7.3.9 Crossbar integrity checker (XBIC)

- XBIC master definition follows the AHB master.
- XBIC slave definition follows the XBAR slave numbering (see *Section 7.3.2.1.2: Crossbar switch slave assignments*).

7.4 Clocking

See *Chapter 24: Clocking* for a description of the architecture for the system level clocks including clock generation, the clock sources and the peripheral clocks. The chapter also provides configuration details for the Clock Monitor Unit (CMU).

7.5 Memories and memory interfaces

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

Table 42. Reference links to related information

Topic	Related module	Reference
Embedded memories	—	*Chapter 3: System architecture*
System memory map	—	*Chapter 5: Memory map*
Platform Flash	—	*Chapter 31: Flash memory controller (PFLASH Controller)*
Platform RAM controller	—	*Chapter 30: Platform RAM controller (PRAM)*
Embedded flash memory	—	*Chapter 32: Embedded Flash Memory*

7.5.1 Flash memory controller (PFLASH) configuration

The flash controller provides flash configuration and control functions and manages the interface between the flash memory array and the crossbar switch. The configuration and control functions are accessed via control registers, which are read/write accessible only in supervisor mode.

Three of the registers, PFCR1, and PFAPR, control the interaction of master modules with the flash array by enabling/disabling prefetch or by controlling access to the flash array on a per-master basis. Use of the fields in these registers requires knowledge of the number assigned to each master. For example, the PFAPR[M2AP] field controls flash array access by master 2, the PFAPR[M3AP] field controls flash array access by master 3, and so on.

The master assignments are shown in the following table. Note that these are the same master assignments used elsewhere in the chip, e.g., the SMPU.

Table 43. Flash controller bus master assignments

Bus Master Number	Bus Master
0	Core0
1	Reserved
2	Reserved
3	DMA
4	Reserved
5	FlexRay
6	Reserved
7	Reserved
8	Core0 Debug
9	Reserved
10	Reserved
11	Reserved
12	Reserved
13	Reserved
14	Reserved
15	Reserved

Table 44. Reference links to related information

Topic	Related module	Reference
Full description	Flash Memory Controller (PFLASH)	*Chapter 31: Flash memory controller (PFLASH Controller)*
Flash memory architecture overview	Flash memory, Flash controller	*Section 3.2.2: Flash memory*
Flash memory	Flash memory	*Chapter 32: Embedded Flash Memory*
System memory map	—	*Chapter 5: Memory map*

7.5.2 Decorated Storage Memory Controller (DSMC)

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

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

As the DSMC generates the atomic read-modify-write bus transactions to the attached slave memory controller, the two transactions (read, write) are fully pipelined with no idle cycles

introduced. During these transactions, the AHB hlock control signal is asserted to the slave during the entire read-modify-write.

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

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

7.6 Analog modules

This section highlights the following analog modules that are implemented on SPC574Sx:

Table 45. Reference links to related information

Module or Subsystem	Device-specific information	Block details
Temperature sensor	*Section 7.6.1: Temperature sensor configuration*	*Chapter 23: Temperature sensor (TSENS)*
ADC subsystem	*Section 33.2: Overview*	—
Successive Approximation Register Analog-to-Digital Converters (SARADCs)	*Section 33.2: Overview*	*Chapter 33: Successive Approximation Register Analog-to-Digital Converter (SARADC) Digital Interface*

7.6.1 Temperature sensor configuration

The temperature sensor is used to measure the temperature of the SPC574Sx devices. In general, the temperature sensor generates an analog voltage that is linearly proportional to temperature. Refer to *Section 23.5: Obtaining device temperature using TSENS* in *Chapter 23: Temperature sensor (TSENS)*

7.6.2 SARADC

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

SPC574Sx implements eight independent 12-bit successive approximation register analog-to-digital converters (SARADCs). The numbering scheme is aligned along the complete family of devices. The SARADCs on the device are referred to as SARADC_B and SARADC_B1(16:1 input mux) and SARADC_0, SARADC_1, SARADC_2, SARADC_3, SARADC_4, and SARADC_5 (maximum 8:1 input mux for each).

7.6.2.1 Register definitions

Each of the following sections shows register definitions for one SARADC instance:

- *Section 7.6.2.1.1: SARADC_B0 register definitions*
- *Section 7.6.2.1.2: SARADC_0 register definitions*
- *Section 7.6.2.1.3: SAR ADC_1 register definitions*
- *Section 7.6.2.1.4: SAR ADC_2 register definitions*
- *Section 7.6.2.1.5: SAR ADC_B1 register definitions*
- *Section 7.6.2.1.6: SAR ADC_3 register definitions*
- *Section 7.6.2.1.7: SAR ADC_4 register definitions*
- *Section 7.6.2.1.8: SAR ADC_5 register definitions*

Table 46: SARADC register descriptions provides a key for the register definition tables. Test channels are only used for SARADC_B.

Note: The following best configurations (in terms of timing) are supported for the SARADC (12 bits) at 12 MHz clock frequency:

1. ICDR[PCE] = 1 (pre-charge enabled)
   - CTRn[PRECHG] = 1 (1 pre-charge cycle)
   - CTRn[INPSAMP] = 6 (6 sampling cycles)
   - 19 clock cycles of total conversion time (1 + 6 + 12)
2. ICDR[PCE] = 0 (pre-charge disabled))
   - CTRn[PRECHG] = 0
   - CTRn[INPSAMP] = 6 (6 sampling cycles)
   - 18 clock cycles of total conversion time (6 + 12)

Table 46. SARADC register descriptions

Name	Description	User access
MCR	Main Configuration Register	Read/Write
MSR	Main Status Register	Read
ISR	Interrupt Status Register	Read
ICIPR*n*	Internal Channel Interrupt Pending Registers	Read
IMR	Interrupt Mask Register	Read/Write
ICIMR*n*	Internal Channel Interrupt Mask Registers	Read/Write
WTISR	Watchdog Threshold Interrupt Status Register	Read
WTIMR	Watchdog Threshold Interrupt Mask Register	Read/Write
DMAE	DMA Enable Register	Read/Write
ICDSR*n*	Internal Channel DMA Select Registers	Read/Write
WTHRHLR0–3	Watchdog Threshold Registers 0–3	Read/Write
CTR*n*	Conversion Timing Registers 0–3	Read/Write
ICNCMR*n*	Internal Channel Normal Conversion Mask Registers	Read/Write
ICJCMR*n*	Internal Channel Injected Conversion Mask Registers	Read/Write
PDEDR	Power Down Exit Delay Register	Read/Write

Table 46. SARADC register descriptions (continued)

Name	Description	User access
ICDR*n*	Internal Channel Data Registers 0–95	Mixed
ICWSELR*n*	Internal Channel Watchdog Select Registers	Read/Write
ICWENR*n*	Internal Channel Watchdog Enable Registers	Read/Write
ICAWORR*n*	Internal Channel Analog Watchdog Out of Range Registers	Read

Table 47. SARADC test channel register descriptions

Name	Description	User access
TCIPR	Test Channel Interrupt Pending Register	Read
TCIMR	Test Channel Interrupt Mask Register	Read/Write
TCDSR	Test Channel DMA Select Register	Read/Write
TCNCMR	Test Channel Normal Conversion Mask Register	Read/Write
TCJCMR	Test Channel Injected Conversion Mask Register	Read/Write
TCWSELR*n*	Test Channel Watchdog Select Register	Read/Write
TCWENR	Test Channel Watchdog Enable Register	Read/Write
TCAWORR	Test Channel Analog Watchdog Out of Range Register	Read
TCCAPR*n*	Test Channel Connection with Analog Pin Registers	Read/Write
TCDR*n*	Test Channel Data Register 96–127	Read/Write

7.6.2.1.1 SARADC_B0 register definitions

Table 48. SARADC_B0 register definitions

Name		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
		16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
MCR	R	OWREN	WLSIDE	MODE	0	NSTAR	0	0	0	JSTART	0	0	0	0	0	0	0
	W																
	R	WTRIGOUT	0	0	0	0	0	0	0	ABORTCHAIN	ABORT	0	FRZ	0	0	0	PWDN
	W																

Table 48. SARADC_B0 register definitions (continued)

Name		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
		16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
MSR	R	0	0	0	0	NSTART	0	0	0	JSTART	0	0	0	0	JABORTCHAIN	0	CTUSTART
	W																
	R	CHADDR								0	0	0	0	0	ADCSTATUS		
	W																
ISR	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	CTUTRGERR	EOCTU	JEOC	JECH	NEOC	NECH
	W											w1c	w1c	w1c	w1c	w1c	w1c
ICIPR0	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	EOC_CH15	EOC_CH14	EOC_CH13	EOC_CH12	EOC_CH11	EOC_CH10	EOC_CH9	EOC_CH8	EOC_CH7	EOC_CH6	EOC_CH5	EOC_CH4	EOC_CH3	EOC_CH2	EOC_CH1	EOC_CH0
	W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
IMR	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	MSKEOCTU	MSKJEOC	MSKJECH	MSKNEOC	MSKNECH
	W																
ICIMR0	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	IM_CH15	IM_CH14	IM_CH13	IM_CH12	IM_CH11	IM_CH10	IM_CH9	IM_CH8	IM_CH7	IM_CH6	IM_CH5	IM_CH4	IM_CH3	IM_CH2	IM_CH1	IM_CH0
	W																

Table 48. SARADC_B0 register definitions (continued)

Name		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
		16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
WTISR	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	WDG3H	WDG3L	WDG2H	WDG2L	WDG1H	WDG1L	WDG0H	WDG0L
	W									w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
WTIMR	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	MSKWDG3H	MSKWDG3L	MSKWDG2H	MSKWDG2L	MSKWDG1H	MSKWDG1L	MSKWDG0H	MSKWDG0L
	W																
DMAE	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	DCLR	DMAEN
	W																
ICDSR0	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R/W	DS_CH15	DS_CH14	DS_CH13	DS_CH12	DS_CH11	DS_CH10	DS_CH9	DS_CH8	DS_CH7	DS_CH6	DS_CH5	DS_CH4	DS_CH3	DS_CH2	DS_CH1	DS_CH0
WTHRHLR0 –WTHRHLR3	R	0	0	0	0	THRH											
	W																
	R	0	0	0	0	THRL											
	W																
CTR0–3	R	CRES	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	PRECHG				INPSAMP							
	W																
ICNCMR0	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R/W	NCE_CH15	NCE_CH14	NCE_CH13	NCE_CH12	NCE_CH11	NCE_CH10	NCE_CH9	NCE_CH8	NCE_CH7	NCE_CH6	NCE_CH5	NCE_CH4	NCE_CH3	NCE_CH2	NCE_CH1	NCE_CH0

Table 48. SARADC_B0 register definitions (continued)

Name		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
		16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
ICJCMR0	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	JCE_CH15	JCE_CH14	JCE_CH13	JCE_CH12	JCE_CH11	JCE_CH10	JCE_CH9	JCE_CH8	JCE_CH7	JCE_CH6	JCE_CH5	JCE_CH4	JCE_CH3	JCE_CH2	JCE_CH1	JCE_CH0
	W																
PDEDR	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	PDED							
	W																
ICDR0–15	R	0	REFSEL	0	0	PCE	0	CTSEL		0	0	0	0	VALID	OVERW	RESULT	
	W																
	R	0	0	0	0	CDATA											
	W																
ICWSELR0	R	0	0	WSEL_CH7		0	0	WSEL_CH6		0	0	WSEL_CH5		0	0	WSEL_CH4	
	W																
	R	0	0	WSEL_CH3		0	0	WSEL_CH2		0	0	WSEL_CH1		0	0	WSEL_CH0	
	W																
ICWSELR1	R	0	0	WSEL_CH15		0	0	WSEL_CH14		0	0	WSEL_CH13		0	0	WSEL_CH12	
	W																
	R	0	0	WSEL_CH11		0	0	WSEL_CH10		0	0	WSEL_CH9		0	0	WSEL_CH8	
	W																
ICWENR0	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	WEN_CH15	WEN_CH14	WEN_CH13	WEN_CH12	WEN_CH11	WEN_CH10	WEN_CH9	WEN_CH8	WEN_CH7	WEN_CH6	WEN_CH5	WEN_CH4	WEN_CH3	WEN_CH2	WEN_CH1	WEN_CH0
	W																

Table 48. SARADC_B0 register definitions (continued)

Name		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
		16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
ICAWORR0	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	AWOR_CH15	AWOR_CH14	AWOR_CH13	AWOR_CH12	AWOR_CH11	AWOR_CH10	AWOR_CH9	AWOR_CH8	AWOR_CH7	AWOR_CH6	AWOR_CH5	AWOR_CH4	AWOR_CH3	AWOR_CH2	AWOR_CH1	AWOR_CH0
	W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c

Table 49. SARADC_B0 test channel register definitions

Name		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
		16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
TCIPR	R	EOC_CH127	EOC_CH126	EOC_CH125	EOC_CH124	0	0	EOC_CH121	EOC_CH120	0	0	0	0	0	0	0	0
	W	w1c	w1c	w1c	w1c			w1c	w1c								
	R	0	0	EOC_CH109	0	0	0	EOC_CH105	EOC_CH104	0	0	0	0	0	0	0	0
	W			w1c				w1c	w1c								
TCIMR	R	IM_CH127	IM_CH126	IM_CH125	IM_CH124	0	0	IM_CH121	IM_CH120	0	0	0	0	0	0	0	0
	W																
	R	0	0	IM_CH109	0	0	0	IM_CH105	IM_CH104	0	0	0	0	0	0	0	0
	W																
TCDSR	R	DS_CH127	DS_CH126	DS_CH125	DS_CH124	0	0	DS_CH121	DS_CH120	0	0	0	0	0	0	0	0
	W																
	R	0	0	DS_CH109	0	0	0	DS_CH105	DS_CH104	0	0	0	0	0	0	0	0
	W																

Table 49. SARADC_B0 test channel register definitions (continued)

Name		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
		16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
TCNCMR	R	NCE_CH127	NCE_CH126	NCE_CH125	NCE_CH124	0	0	NCE_CH121	NCE_CH120	0	0	0	0	0	0	0	0
	W																
	R	0	0	NCE_CH109	0	0	0	NCE_CH105	NCE_CH104	0	0	0	0	0	0	0	0
	W																
TCJCMR	R	JCE_CH127	JCE_CH126	JCE_CH125	JCE_CH124	0	0	JCE_CH121	JCE_CH120	0	0	0	0	0	0	0	0
	W																
	R	0	0	JCE_CH109	0	0	0	JCE_CH105	JCE_CH104	0	0	0	0	0	0	0	0
	W																
TCWSELR1	R	0	0	0	0	0	0	0	0	0	0	WSEL_CH109		0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	WSEL_CH105		0	0	WSEL_CH104	
	W																
TCWSELR3	R	0	0	WSEL_CH127		0		WSEL_CH126		0		WSEL_CH125		0		WSEL_CH124	
	W																
	R	0	0	0	0	0	0	0	0	0	0	WSEL_CH121		0		WSEL_CH120	
	W																
TCWENR	R	WEN_CH127	WEN_CH126	WEN_CH125	WEN_CH124	0	0	WEN_CH121	WEN_CH120	0	0	0	0	0	0	0	0
	W																
	R	0	0	WEN_CH109	0	0	0	WEN_CH105	WEN_CH104	0	0	0	0	0	0	0	0
	W																

Table 49. SARADC_B0 test channel register definitions (continued)

Name		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
		16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
TCAWORR	R	AWOR_CH127	AWOR_CH126	AWOR_CH125	AWOR_CH124	0	0	AWOR_CH121	AWOR_CH120	0	0	0	0	0	0	0	0
	W	w1c	w1c	w1c	w1c			w1c	w1c								
	R	0	0	AWOR_CH109	0	0	0	AWOR_CH105	AWOR_CH104	0	0	0	0	0	0	0	0
	W			w1c				w1c	w1c								
TCCAPR7	R	ESIC_TCH127	ICSEL_TCH127							ESIC_TCH126	ICSEL_TCH126						
	W																
	R	ESIC_TCH125	ICSEL_TCH125							ESIC_TCH124	ICSEL_TCH124						
	W																
TCDR104–105, TCDR109, TCDR120–121, TCDR124–127	R	0	0	0	0	PCE	0	CTSEL		0	0	0	0	VALID	OVERW	RESULT	
	W																
	R	0	0	0	0	CDATA											
	W																

7.6.2.1.2 SARADC_0 register definitions

Table 50. SARADC_0 register definitions

Name		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
		16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
MCR	R	OWREN	WLSIDE	MODE	0	NSTAR	0	0	0	JSTART	0	0	0	0	0	0	0
	W																
	R	WTRIGOUT	0	0	0	0	0	0	0	ABORTCHAIN	ABORT	0	FRZ	0	0	0	PWDN
	W																
MSR	R	0	0	0	0	NSTART	0	0	0	JSTART	0	0	0	0	JABORTCHAIN	0	CTUSTART
	W																
	R	CHADDR								0	0	0	0	0	ADCSTATUS		
	W																
ISR	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	CTUTRGERR	EOCTU	JEOC	JECH	NEOC	NECH
	W											w1c	w1c	w1c	w1c	w1c	w1c
ICIPR0	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	EOC_CH5	EOC_CH4	EOC_CH3	EOC_CH2	EOC_CH1	EOC_CH0
	W											w1c	w1c	w1c	w1c	w1c	w1c
IMR	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	MSKEOCTU	MSKJEOC	MSKJECH	MSKNEOC	MSKNECH
	W																

Table 50. SARADC_0 register definitions (continued)

Name		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
		16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
ICIMR0	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	IM_CH5	IM_CH4	IM_CH3	IM_CH2	IM_CH1	IM_CH0
	W																
WTISR	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	WDG3H	WDG3L	WDG2H	WDG2L	WDG1H	WDG1L	WDG0H	WDG0L
	W									w1 c	w1 c	w1 c	w1 c	w1 c	w1 c	w1 c	w1 c
WTIMR	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	MSKWDG3H	MSKWDG3L	MSKWDG2H	MSKWDG2L	MSKWDG1H	MSKWDG1L	MSKWDG0H	MSKWDG0L
	W																
DMAE	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	DCLR	DMAEN
	W																
ICDSR0	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	DS_CH5	DS_CH4	DS_CH3	DS_CH2	DS_CH1	DS_CH0
	W																
WTHRHLR0 –WTHRHLR3	R	0	0	0	0	THRH											
	W																
	R	0	0	0	0	THRL											
	W																
CTR0–3	R	CRES	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	PRECHG				INPSAMP							
	W																

Table 50. SARADC_0 register definitions (continued)

Name		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
		16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
ICNCMR0	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	NCE_CH5	NCE_CH4	NCE_CH3	NCE_CH2	NCE_CH1	NCE_CH0
	W																
ICJCMR0	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	JCE_CH5	JCE_CH4	JCE_CH3	JCE_CH2	JCE_CH1	JCE_CH0
	W																
PDEDR	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	PDED							
	W																
ICDR0–15	R	0	REFSEL	0	0	PCE	0	CTSEL		0	0	0	0	VALID	OVERW	RESULT	
	W																
	R	0	0	0	0	CDATA											
	W																
ICWSELR0	R	0	0	0	0	0	0	0	0	0	0	WSEL_CH5		0	0	WSEL_CH4	
	W																
	R	0	0	WSEL_CH3		0	0	WSEL_CH2		0	0	WSEL_CH1		0	0	WSEL_CH0	
	W																
ICWENR0	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	WEN_CH5	WEN_CH4	WEN_CH3	WEN_CH2	WEN_CH1	WEN_CH0
	W																

Table 50. SARADC_0 register definitions (continued)

Name		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
		16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
ICAWORR0	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	AWOR_CH5	AWOR_CH4	AWOR_CH3	AWOR_CH2	AWOR_CH1	AWOR_CH0
	W											w1c	w1c	w1c	w1c	w1c	w1c

Table 51. SARADC_0 test channel register definitions

Name		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
		16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
TCIPR	R	EOC_CH127	EOC_CH126	EOC_CH125	EOC_CH124	0	0	EOC_CH121	EOC_CH120	0	0	0	0	0	0	0	0
	W	w1c	w1c	w1c	w1c			w1c	w1c								
	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
TCIMR	R	IM_CH127	IM_CH126	IM_CH125	IM_CH124	0	0	IM_CH121	IM_CH120	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
TCDSR	R	DS_CH127	DS_CH126	DS_CH125	DS_CH124	0	0	DS_CH121	DS_CH120	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
TCNCMR	R	NCE_CH127	NCE_CH126	NCE_CH125	NCE_CH124	0	0	NCE_CH121	NCE_CH120	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																

Table 51. SARADC_0 test channel register definitions (continued)

Name		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
		16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
TCJCMR	R	JCE_CH127	JCE_CH126	JCE_CH125	JCE_CH124	0	0	JCE_CH121	JCE_CH120	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
TCWSELR3	R	0	0	WSEL_CH127		0		WSEL_CH126		0		WSEL_CH125		0		WSEL_CH124	
	W																
	R	0	0	0	0	0	0	0	0	0	0	WSEL_CH121		0		WSEL_CH120	
	W																
TCWENR	R	WEN_CH127	WEN_CH126	WEN_CH125	WEN_CH124	0	0	WEN_CH121	WEN_CH120	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
TCAWORR	R	AWOR_CH127	AWOR_CH126	AWOR_CH125	AWOR_CH124	0	0	AWOR_CH121	AWOR_CH120	0	0	0	0	0	0	0	0
	W	w1c	w1c	w1c	w1c			w1c	w1c								
	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
TCCAPR7	R	ESIC_TCH127	ICSEL_TCH127							ESIC_TCH126	ICSEL_TCH126						
	W																
	R	ESIC_TCH125	ICSEL_TCH125							ESIC_TCH124	ICSEL_TCH124						
	W																

Table 51. SARADC_0 test channel register definitions (continued)

Name		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
		16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
TCDR120–121, TCDR124–127	R	0	0	0	0	PCE	0	CTSEL		0	0	0	0	VALID	OVERW	RESULT	
	W																
	R	0	0	0	0	CDATA											
	W																

7.6.2.1.3 SAR ADC_1 register definitions

Table 52. SARADC_1 register definitions

Name		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
		16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
MCR	R	OWREN	WLSIDE	MODE	0	NSTAR	0	0	0	JSTART	0	0	0	0	0	0	0
	W																
	R	WTRIGOUT	0	0	0	0	0	0	0	ABORTCHAIN	ABORT	0	FRZ	0	0	0	PWDN
	W																
MSR	R	0	0	0	0	NSTART	0	0	0	JSTART	0	0	0	0	JABORTCHAIN	0	CTUSTART
	W																
	R	CHADDR								0	0	0	0	0	ADCSTATUS		
	W																
ISR	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	CTUTRGERR	EOCTU	JEOC	JECH	NEOC	NECH
	W											w1c	w1c	w1c	w1c	w1c	w1c

Table 52. SARADC_1 register definitions (continued)

Name		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
		16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
ICIPR0	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	EOC_CH15	EOC_CH14	EOC_CH13	EOC_CH12	EOC_CH11	EOC_CH10	EOC_CH9	EOC_CH8	EOC_CH7	EOC_CH6	EOC_CH5	EOC_CH4	EOC_CH3	EOC_CH2	EOC_CH1	EOC_CH0
	W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
IMR	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	MSKEOCTU	MSKJEOC	MSKJECH	MSKNEOC	MSKNECH
	W																
ICIMR0	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	IM_CH15	IM_CH14	IM_CH13	IM_CH12	IM_CH11	IM_CH10	IM_CH9	IM_CH8	IM_CH7	IM_CH6	IM_CH5	IM_CH4	IM_CH3	IM_CH2	IM_CH1	IM_CH0
	W																
WTISR	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	WDG3H	WDG3L	WDG2H	WDG2L	WDG1H	WDG1L	WDG0H	WDG0L
	W									w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
WTIMR	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	MSKWDG3H	MSKWDG3L	MSKWDG2H	MSKWDG2L	MSKWDG1H	MSKWDG1L	MSKWDG0H	MSKWDG0L
	W																
DMAE	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	DCLR	DMAEN
	W																

Table 52. SARADC_1 register definitions (continued)

Name		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
		16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
ICDSR0	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	DS_CH15	DS_CH14	DS_CH13	DS_CH12	DS_CH11	DS_CH10	DS_CH9	DS_CH8	DS_CH7	DS_CH6	DS_CH5	DS_CH4	DS_CH3	DS_CH2	DS_CH1	DS_CH0
	W																
WTHRHLR0 –WTHRHLR3	R	0	0	0	0	THRH											
	W																
	R	0	0	0	0	THRL											
	W																
CTR0–3	R	CRES	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	PRECHG				INPSAMP							
	W																
ICNCMR0	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	NCE_CH15	NCE_CH14	NCE_CH13	NCE_CH12	NCE_CH11	NCE_CH10	NCE_CH9	NCE_CH8	NCE_CH7	NCE_CH6	NCE_CH5	NCE_CH4	NCE_CH3	NCE_CH2	NCE_CH1	NCE_CH0
	W																
ICJCMR0	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	JCE_CH15	JCE_CH14	JCE_CH13	JCE_CH12	JCE_CH11	JCE_CH10	JCE_CH9	JCE_CH8	JCE_CH7	JCE_CH6	JCE_CH5	JCE_CH4	JCE_CH3	JCE_CH2	JCE_CH1	JCE_CH0
	W																
PDEDR	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	PDED							
	W																
ICDR0–15	R	0	REFSEL	0	0	PCE	0	CTSEL		0	0	0	0	VALID	OVERW	RESULT	
	W																
	R	0	0	0	0	CDATA											
	W																

Table 52. SARADC_1 register definitions (continued)

Name		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
		16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
ICWSELR0	R	0	0	WSEL_CH7		0	0	WSEL_CH6		0	0	WSEL_CH5		0	0	WSEL_CH4	
	W																
	R	0	0	WSEL_CH3		0	0	WSEL_CH2		0	0	WSEL_CH1		0	0	WSEL_CH0	
	W																
ICWSELR1	R	0	0	WSEL_CH15		0	0	WSEL_CH14		0	0	WSEL_CH13		0	0	WSEL_CH12	
	W																
	R	0	0	WSEL_CH11		0	0	WSEL_CH10		0	0	WSEL_CH9		0	0	WSEL_CH8	
	W																
ICWENR0	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	WEN_CH15	WEN_CH14	WEN_CH13	WEN_CH12	WEN_CH11	WEN_CH10	WEN_CH9	WEN_CH8	WEN_CH7	WEN_CH6	WEN_CH5	WEN_CH4	WEN_CH3	WEN_CH2	WEN_CH1	WEN_CH0
	W																
ICAWORR0	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	AWOR_CH15	AWOR_CH14	AWOR_CH13	AWOR_CH12	AWOR_CH11	AWOR_CH10	AWOR_CH9	AWOR_CH8	AWOR_CH7	AWOR_CH6	AWOR_CH5	AWOR_CH4	AWOR_CH3	AWOR_CH2	AWOR_CH1	AWOR_CH0
	W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c

Table 53. SARADC_1 test channel register definitions

Name		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
		16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
TCIPR	R	EOC_CH127	EOC_CH126	EOC_CH125	EOC_CH124	0	0	EOC_CH121	EOC_CH120	0	0	0	0	0	0	0	0
	W	w1c	w1c	w1c	w1c			w1c	w1c								
	R	0	0	EOC_CH109	0	0	0	EOC_CH105	EOC_CH104	0	0	0	0	0	0	0	0
	W			w1c				w1c	w1c								
TCIMR	R	IM_CH127	IM_CH126	IM_CH125	IM_CH124	0	0	IM_CH121	IM_CH120	0	0	0	0	0	0	0	0
	W																
	R	0	0	IM_CH109	0	0	0	IM_CH105	IM_CH104	0	0	0	0	0	0	0	0
	W																
TCDSR	R	DS_CH127	DS_CH126	DS_CH125	DS_CH124	0	0	DS_CH121	DS_CH120	0	0	0	0	0	0	0	0
	W																
	R	0	0	DS_CH109	0	0	0	DS_CH105	DS_CH104	0	0	0	0	0	0	0	0
	W																
TCNCMR	R	NCE_CH127	NCE_CH126	NCE_CH125	NCE_CH124	0	0	NCE_CH121	NCE_CH120	0	0	0	0	0	0	0	0
	W																
	R	0	0	NCE_CH109	0	0	0	NCE_CH105	NCE_CH104	0	0	0	0	0	0	0	0
	W																

Table 53. SARADC_1 test channel register definitions (continued)

Name		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
		16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
TCJCMR	R	JCE_CH127	JCE_CH126	JCE_CH125	JCE_CH124	0	0	JCE_CH121	JCE_CH120	0	0	0	0	0	0	0	0
	W																
	R	0	0	JCE_CH109	0	0	0	JCE_CH105	JCE_CH104	0	0	0	0	0	0	0	0
	W																
TCWSELR1	R	0	0	0	0	0	0	0	0	0	0	WSEL_CH109		0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	WSEL_CH105		0	0	WSEL_CH104	
	W																
TCWSELR3	R	0	0	WSEL_CH127		0		WSEL_CH126		0		WSEL_CH125		0		WSEL_CH124	
	W																
	R	0	0	0	0	0	0	0	0	0	0	WSEL_CH121		0		WSEL_CH120	
	W																
TCWENR	R	WEN_CH127	WEN_CH126	WEN_CH125	WEN_CH124	0	0	WEN_CH121	WEN_CH120	0	0	0	0	0	0	0	0
	W																
	R	0	0	WEN_CH109	0	0	0	WEN_CH105	WEN_CH104	0	0	0	0	0	0	0	0
	W																
TCAWORR	R	AWOR_CH127	AWOR_CH126	AWOR_CH125	AWOR_CH124	0	0	AWOR_CH121	AWOR_CH120	0	0	0	0	0	0	0	0
	W	w1c	w1c	w1c	w1c			w1c	w1c								
	R	0	0	AWOR_CH109	0	0	0	AWOR_CH105	AWOR_CH104	0	0	0	0	0	0	0	0
	W			w1c				w1c	w1c								

Table 53. SARADC_1 test channel register definitions (continued)

Name		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
		16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
TCCAPR7	R	ESIC_TCH127	ICSEL_TCH127							ESIC_TCH126	ICSEL_TCH126						
	W																
	R	ESIC_TCH125	ICSEL_TCH125							ESIC_TCH124	ICSEL_TCH124						
	W																
TCDR104–105, TCDR109, TCDR120–121, TCDR124–127	R	0	0	0	0	PCE	0	CTSEL		0	0	0	0	VALID	OVERW	RESULT	
	W																
	R	0	0	0	0	CDATA											
	W																

7.6.2.1.4 SAR ADC_2 register definitions

Table 54. SARADC_2 register definitions

Name		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
		16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
MCR	R	OWREN	WLSIDE	MODE	0	NSTAR	0	0	0	JSTART	0	0	0	0	0	0	0
	W																
	R	WTRIGOUT	0	0	0	0	0	0	0	ABORTCHAIN	ABORT	0	FRZ	0	0	0	PWDN
	W																
MSR	R	0	0	0	0	NSTART	0	0	0	JSTART	0	0	0	0	JABORTCHAIN	0	CTUSTART
	W																
	R	CHADDR								0	0	0	0	0	ADCSTATUS		
	W																

Table 54. SARADC_2 register definitions (continued)

Name		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
		16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
ISR	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	CTUTRGERR	EOCTU	JEOC	JECH	NEOC	NECH
	W											w1c	w1c	w1c	w1c	w1c	w1c
ICIPR0	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	EOC_CH5	EOC_CH4	EOC_CH3	EOC_CH2	EOC_CH1	EOC_CH0
	W											w1c	w1c	w1c	w1c	w1c	w1c
IMR	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	MSKEOCTU	MSKJEOC	MSKJECH	MSKNEOC	MSKNECH
	W																
ICIMR0	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	IM_CH5	IM_CH4	IM_CH3	IM_CH2	IM_CH1	IM_CH0
	W																
WTISR	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	WDG3H	WDG3L	WDG2H	WDG2L	WDG1H	WDG1L	WDG0H	WDG0L
	W									w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c

Table 54. SARADC_2 register definitions (continued)

Name		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
		16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
WTIMR	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	MSKWDG3H	MSKWDG3L	MSKWDG2H	MSKWDG2L	MSKWDG1H	MSKWDG1L	MSKWDG0H	MSKWDG0L
	W																
DMAE	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	DCLR	DMAEN
	W																
ICDSR0	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	DS_CH5	DS_CH4	DS_CH3	DS_CH2	DS_CH1	DS_CH0
	W																
WTHRHLR0 –WTHRHLR3	R	0	0	0	0	THRH											
	W																
	R	0	0	0	0	THRL											
	W																
CTR0–3	R	CRES	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	PRECHG				INPSAMP							
	W																
ICNCMR0	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	NCE_CH5	NCE_CH4	NCE_CH3	NCE_CH2	NCE_CH1	NCE_CH0
	W																
ICJCMR0	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	JCE_CH5	JCE_CH4	JCE_CH3	JCE_CH2	JCE_CH1	JCE_CH0
	W																

Table 54. SARADC_2 register definitions (continued)

Name		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
		16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
PDEDR	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	PDED							
	W																
ICDR0–15	R	0	REFSEL	0	0	PCE	0	CTSEL		0	0	0	0	VALID	OVERW	RESULT	
	W																
	R	0	0	0	0	CDATA											
	W																
ICWSELR0	R	0	0	0	0	0	0	0	0	0	0	WSEL_CH5		0	0	WSEL_CH4	
	W																
	R	0	0	WSEL_CH3		0	0	WSEL_CH2		0	0	WSEL_CH1		0	0	WSEL_CH0	
	W																
ICWENR0	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	WEN_CH5	WEN_CH4	WEN_CH3	WEN_CH2	WEN_CH1	WEN_CH0
	W																
ICAWORR0	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	AWOR_CH5	AWOR_CH4	AWOR_CH3	AWOR_CH2	AWOR_CH1	AWOR_CH0
	W											w1c	w1c	w1c	w1c	w1c	w1c

Table 55. SARADC_2 test channel register definitions

Name		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
		16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
TCIPR	R	EOC_CH127	EOC_CH126	EOC_CH125	EOC_CH124	0	0	EOC_CH121	EOC_CH120	0	0	0	0	0	0	0	0
	W	w1c	w1c	w1c	w1c			w1c	w1c								
	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
TCIMR	R	IM_CH127	IM_CH126	IM_CH125	IM_CH124	0	0	IM_CH121	IM_CH120	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
TCDSR	R	DS_CH127	DS_CH126	DS_CH125	DS_CH124	0	0	DS_CH121	DS_CH120	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
TCNCMR	R	NCE_CH127	NCE_CH126	NCE_CH125	NCE_CH124	0	0	NCE_CH121	NCE_CH120	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
TCJCMR	R	JCE_CH127	JCE_CH126	JCE_CH125	JCE_CH124	0	0	JCE_CH121	JCE_CH120	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
TCWSELR3	R	0	0	WSEL_CH127			0	WSEL_CH126			0	WSEL_CH125			0	WSEL_CH124	
	W																
	R	0	0	0	0	0	0	0	0	0	0	WSEL_CH121			0	WSEL_CH120	
	W																

Table 55. SARADC_2 test channel register definitions (continued)

Name		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
		16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
TCWENR	R	WEN_CH127	WEN_CH126	WEN_CH125	WEN_CH124	0	0	WEN_CH121	WEN_CH120	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
TCAWORR	R	AWOR_CH127	AWOR_CH126	AWOR_CH125	AWOR_CH124	0	0	AWOR_CH121	AWOR_CH120	0	0	0	0	0	0	0	0
	W	w1c	w1c	w1c	w1c			w1c	w1c								
	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
TCCAPR7	R	ESIC_TCH127	ICSEL_TCH127							ESIC_TCH126	ICSEL_TCH126						
	W																
	R	ESIC_TCH125	ICSEL_TCH125							ESIC_TCH124	ICSEL_TCH124						
	W																
TCDR120–121, TCDR124–127	R	0	0	0	0	PCE	0	CTSEL		0	0	0	0	VALID	OVERW	RESULT	
	W																
	R	0	0	0	0	CDATA											
	W																

7.6.2.1.5 SAR ADC_B1 register definitions

Table 56. SARADC_B1 register definitions

Name		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
		16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
MCR	R	OWREN	WLSIDE	MODE	0	NSTAR	0	0	0	JSTART	0	0	0	0	0	0	0
	W																
	R	WTRIGOUT	0	0	0	0	0	0	0	ABORTCHAIN	ABORT	0	FRZ	0	0	0	PWDN
	W																
MSR	R	0	0	0	0	NSTART	0	0	0	JSTART	0	0	0	0	JABORTCHAIN	0	CTUSTART
	W																
	R	CHADDR								0	0	0	0	0	ADCSTATUS		
	W																
ISR	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	CTUTRGERR	EOCTU	JEOC	JECH	NEOC	NECH
	W											w1c	w1c	w1c	w1c	w1c	w1c
ICIPR0	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	EOC_CH15	EOC_CH14	EOC_CH13	EOC_CH12	EOC_CH11	EOC_CH10	EOC_CH9	EOC_CH8	EOC_CH7	EOC_CH6	EOC_CH5	EOC_CH4	EOC_CH3	EOC_CH2	EOC_CH1	EOC_CH0
	W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
IMR	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	MSKEOCTU	MSKJEOC	MSKJECH	MSKNEOC	MSKNECH
	W																

Table 56. SARADC_B1 register definitions (continued)

Name		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
		16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
ICIMR0	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	IM_CH15	IM_CH14	IM_CH13	IM_CH12	IM_CH11	IM_CH10	IM_CH9	IM_CH8	IM_CH7	IM_CH6	IM_CH5	IM_CH4	IM_CH3	IM_CH2	IM_CH1	IM_CH0
	W																
WTISR	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	WDG3H	WDG3L	WDG2H	WDG2L	WDG1H	WDG1L	WDG0H	WDG0L
	W									w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
WTIMR	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	MSKWDG3H	MSKWDG3L	MSKWDG2H	MSKWDG2L	MSKWDG1H	MSKWDG1L	MSKWDG0H	MSKWDG0L
	W																
DMAE	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	DCLR	DMAEN
	W																
ICDSR0	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	DS_CH15	DS_CH14	DS_CH13	DS_CH12	DS_CH11	DS_CH10	DS_CH9	DS_CH8	DS_CH7	DS_CH6	DS_CH5	DS_CH4	DS_CH3	DS_CH2	DS_CH1	DS_CH0
	W																
WTHRHLR0 –WTHRHLR3	R	0	0	0	0	THRH											
	W																
	R	0	0	0	0	THRL											
	W																
CTR0–3	R	CRES	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	PRECHG				INPSAMP							
	W																

Table 56. SARADC_B1 register definitions (continued)

Name		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
		16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
ICNCMR0	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	NCE_CH15	NCE_CH14	NCE_CH13	NCE_CH12	NCE_CH11	NCE_CH10	NCE_CH9	NCE_CH8	NCE_CH7	NCE_CH6	NCE_CH5	NCE_CH4	NCE_CH3	NCE_CH2	NCE_CH1	NCE_CH0
	W																
ICJCMR0	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	JCE_CH15	JCE_CH14	JCE_CH13	JCE_CH12	JCE_CH11	JCE_CH10	JCE_CH9	JCE_CH8	JCE_CH7	JCE_CH6	JCE_CH5	JCE_CH4	JCE_CH3	JCE_CH2	JCE_CH1	JCE_CH0
	W																
PDEDR	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	PDED							
	W																
ICDR0–15	R	0	REFSEL	0	0	PCE	0	CTSEL		0	0	0	0	VALID	OVERW	RESULT	
	W																
	R	0	0	0	0	CDATA											
	W																
ICWSELR0	R	0	0	WSEL_CH7		0	0	WSEL_CH6		0	0	WSEL_CH5		0	0	WSEL_CH4	
	W																
	R	0	0	WSEL_CH3		0	0	WSEL_CH2		0	0	WSEL_CH1		0	0	WSEL_CH0	
	W																
ICWSELR1	R	0	0	WSEL_CH15		0	0	WSEL_CH14		0	0	WSEL_CH13		0	0	WSEL_CH12	
	W																
	R	0	0	WSEL_CH11		0	0	WSEL_CH10		0	0	WSEL_CH9		0	0	WSEL_CH8	
	W																
ICWENR0	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	WEN_CH15	WEN_CH14	WEN_CH13	WEN_CH12	WEN_CH11	WEN_CH10	WEN_CH9	WEN_CH8	WEN_CH7	WEN_CH6	WEN_CH5	WEN_CH4	WEN_CH3	WEN_CH2	WEN_CH1	WEN_CH0
	W																

Table 56. SARADC_B1 register definitions (continued)

Name		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
		16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
ICAWORR0	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	AWOR_CH15	AWOR_CH14	AWOR_CH13	AWOR_CH12	AWOR_CH11	AWOR_CH10	AWOR_CH9	AWOR_CH8	AWOR_CH7	AWOR_CH6	AWOR_CH5	AWOR_CH4	AWOR_CH3	AWOR_CH2	AWOR_CH1	AWOR_CH0
	W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c

Table 57. SARADC_B1 test channel register definitions

Name		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
		16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
TCIPR	R	EOC_CH127	EOC_CH126	EOC_CH125	EOC_CH124	0	0	EOC_CH121	EOC_CH120	0	0	0	0	0	0	0	0
	W	w1c	w1c	w1c	w1c			w1c	w1c								
	R	0	0	EOC_CH109	0	0	0	EOC_CH105	EOC_CH104	0	0	0	0	0	0	0	0
	W			w1c				w1c	w1c								
TCIMR	R	IM_CH127	IM_CH126	IM_CH125	IM_CH124	0	0	IM_CH121	IM_CH120	0	0	0	0	0	0	0	0
	W																
	R	0	0	IM_CH109	0	0	0	IM_CH105	IM_CH104	0	0	0	0	0	0	0	0
	W																
TCDSR	R	DS_CH127	DS_CH126	DS_CH125	DS_CH124	0	0	DS_CH121	DS_CH120	0	0	0	0	0	0	0	0
	W																
	R	0	0	DS_CH109	0	0	0	DS_CH105	DS_CH104	0	0	0	0	0	0	0	0
	W																

Table 57. SARADC_B1 test channel register definitions (continued)

Name		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
		16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
TCNCMR	R	NCE_CH127	NCE_CH126	NCE_CH125	NCE_CH124	0	0	NCE_CH121	NCE_CH120	0	0	0	0	0	0	0	0
	W																
	R	0	0	NCE_CH109	0	0	0	NCE_CH105	NCE_CH104	0	0	0	0	0	0	0	0
	W																
TCJCMR	R	JCE_CH127	JCE_CH126	JCE_CH125	JCE_CH124	0	0	JCE_CH121	JCE_CH120	0	0	0	0	0	0	0	0
	W																
	R	0	0	JCE_CH109	0	0	0	JCE_CH105	JCE_CH104	0	0	0	0	0	0	0	0
	W																
TCWSELR1	R	0	0	0	0	0	0	0	0	0	0	WSEL_CH109		0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	WSEL_CH105		0	0	WSEL_CH104	
	W																
TCWSELR3	R	0	0	WSEL_CH127		0		WSEL_CH126		0		WSEL_CH125		0		WSEL_CH124	
	W																
	R	0	0	0	0	0	0	0	0	0	0	WSEL_CH121		0		WSEL_CH120	
	W																
TCWENR	R	WEN_CH127	WEN_CH126	WEN_CH125	WEN_CH124	0	0	WEN_CH121	WEN_CH120	0	0	0	0	0	0	0	0
	W																
	R	0	0	WEN_CH109	0	0	0	WEN_CH105	WEN_CH104	0	0	0	0	0	0	0	0
	W																

Table 57. SARADC_B1 test channel register definitions (continued)

Name		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
		16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
TCAWORR	R	AWOR_CH127	AWOR_CH126	AWOR_CH125	AWOR_CH124	0	0	AWOR_CH121	AWOR_CH120	0	0	0	0	0	0	0	0
	W	w1c	w1c	w1c	w1c			w1c	w1c								
	R	0	0	AWOR_CH109	0	0	0	AWOR_CH105	AWOR_CH104	0	0	0	0	0	0	0	0
	W			w1c				w1c	w1c								
TCCAPR7	R	ESIC_TCH127	ICSEL_TCH127							ESIC_TCH126	ICSEL_TCH126						
	W																
	R	ESIC_TCH125	ICSEL_TCH125							ESIC_TCH124	ICSEL_TCH124						
	W																
TCDR104–105, TCDR109, TCDR120–121, TCDR124–127	R	0	0	0	0	PCE	0	CTSEL		0	0	0	0	VALID	OVERW	RESULT	
	W																
	R	0	0	0	0	CDATA											
	W																

7.6.2.1.6 SAR ADC_3 register definitions

Table 58. SARADC_3 register definitions

Name		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
		16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
MCR	R	OWREN	WLSIDE	MODE	0	NSTAR	0	0	0	JSTART	0	0	0	0	0	0	0
	W																
	R	WTRIGOUT	0	0	0	0	0	0	0	ABORTCHAIN	ABORT	0	FRZ	0	0	0	PWDN
	W																
MSR	R	0	0	0	0	NSTART	0	0	0	JSTART	0	0	0	0	JABORTCHAIN	0	CTUSTART
	W																
	R	CHADDR								0	0	0	0	0	ADCSTATUS		
	W																
ISR	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	CTUTRGERR	EOCTU	JEOC	JECH	NEOC	NECH
	W											w1c	w1c	w1c	w1c	w1c	w1c
ICIPR0	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	EOC_CH5	EOC_CH4	EOC_CH3	EOC_CH2	EOC_CH1	EOC_CH0
	W											w1c	w1c	w1c	w1c	w1c	w1c
IMR	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	MSKEOCTU	MSKJEOC	MSKJECH	MSKNEOC	MSKNECH
	W																

Table 58. SARADC_3 register definitions (continued)

Name		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
		16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
ICIMR0	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	IM_CH5	IM_CH4	IM_CH3	IM_CH2	IM_CH1	IM_CH0
	W																
WTISR	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	WDG3H	WDG3L	WDG2H	WDG2L	WDG1H	WDG1L	WDG0H	WDG0L
	W									w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
WTIMR	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	MSKWDG3H	MSKWDG3L	MSKWDG2H	MSKWDG2L	MSKWDG1H	MSKWDG1L	MSKWDG0H	MSKWDG0L
	W																
DMAE	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	DCLR	DMAEN
	W																
ICDSR0	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	DS_CH5	DS_CH4	DS_CH3	DS_CH2	DS_CH1	DS_CH0
	W																
WTHRHLR0 –WTHRHLR3	R	0	0	0	0	THRH											
	W																
	R	0	0	0	0	THRL											
	W																
CTR0–3	R	CRES	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	PRECHG				INPSAMP							
	W																

Table 58. SARADC_3 register definitions (continued)

Name		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
		16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
ICNCMR0	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	NCE_CH5	NCE_CH4	NCE_CH3	NCE_CH2	NCE_CH1	NCE_CH0
	W																
ICJCMR0	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	JCE_CH5	JCE_CH4	JCE_CH3	JCE_CH2	JCE_CH1	JCE_CH0
	W																
PDEDR	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	PDED							
	W																
ICDR0–15	R	0	REFSEL	0	0	PCE	0	CTSEL		0	0	0	0	VALID	OVERW	RESULT	
	W																
	R	0	0	0	0	CDATA											
	W																
ICWSELR0	R	0	0	0	0	0	0	0	0	0	0	WSEL_CH5		0	0	WSEL_CH4	
	W																
	R	0	0	WSEL_CH3		0	0	WSEL_CH2		0	0	WSEL_CH1		0	0	WSEL_CH0	
	W																
ICWENR0	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	WEN_CH5	WEN_CH4	WEN_CH3	WEN_CH2	WEN_CH1	WEN_CH0
	W																

Table 58. SARADC_3 register definitions (continued)

Name		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
		16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
ICAWORR0	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	AWOR_CH5	AWOR_CH4	AWOR_CH3	AWOR_CH2	AWOR_CH1	AWOR_CH0
	W											w1c	w1c	w1c	w1c	w1c	w1c

Table 59. SARADC_3 test channel register definitions

Name		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
		16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
TCIPR	R	EOC_CH127	EOC_CH126	EOC_CH125	EOC_CH124	0	0	EOC_CH121	EOC_CH120	0	0	0	0	0	0	0	0
	W	w1c	w1c	w1c	w1c			w1c	w1c								
	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
TCIMR	R	IM_CH127	IM_CH126	IM_CH125	IM_CH124	0	0	IM_CH121	IM_CH120	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
TCDSR	R	DS_CH127	DS_CH126	DS_CH125	DS_CH124	0	0	DS_CH121	DS_CH120	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
TCNCMR	R	NCE_CH127	NCE_CH126	NCE_CH125	NCE_CH124	0	0	NCE_CH121	NCE_CH120	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																

Table 59. SARADC_3 test channel register definitions (continued)

Name		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
		16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
TCJCMR	R	JCE_CH127	JCE_CH126	JCE_CH125	JCE_CH124	0	0	JCE_CH121	JCE_CH120	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
TCWSELR3	R	0	0	WSEL_CH127		0		WSEL_CH126		0		WSEL_CH125		0		WSEL_CH124	
	W																
	R	0	0	0	0	0	0	0	0	0	0	WSEL_CH121		0		WSEL_CH120	
	W																
TCWENR	R	WEN_CH127	WEN_CH126	WEN_CH125	WEN_CH124	0	0	WEN_CH121	WEN_CH120	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
TCAWORR	R	AWOR_CH127	AWOR_CH126	AWOR_CH125	AWOR_CH124	0	0	AWOR_CH121	AWOR_CH120	0	0	0	0	0	0	0	0
	W	w1c	w1c	w1c	w1c			w1c	w1c								
	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
TCCAPR7	R	ESIC_TCH127	ICSEL_TCH127							ESIC_TCH126	ICSEL_TCH126						
	W																
	R	ESIC_TCH125	ICSEL_TCH125							ESIC_TCH124	ICSEL_TCH124						
	W																

Table 59. SARADC_3 test channel register definitions (continued)

Name		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
		16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
TCDR120–121, TCDR124–127	R	0	0	0	0	PCE	0	CTSEL		0	0	0	0	VALID	OVERW	RESULT	
	W																
	R	0	0	0	0	CDATA											
	W																

7.6.2.1.7 SAR ADC_4 register definitions

Table 60. SARADC_4 register definitions

Name		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
		16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
MCR	R	OWREN	WLSIDE	MODE	0	NSTAR	0	0	0	JSTART	0	0	0	0	0	0	0
	W																
	R	WTRIGOUT	0	0	0	0	0	0	0	ABORTCHAIN	ABORT	0	FRZ	0	0	0	PWDN
	W																
MSR	R	0	0	0	0	NSTART	0	0	0	JSTART	0	0	0	0	JABORTCHAIN	0	CTUSTART
	W																
	R	CHADDR								0	0	0	0	0	ADCSTATUS		
	W																
ISR	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	CTUTRGERR	EOCTU	JEOC	JECH	NEOC	NECH
	W											w1c	w1c	w1c	w1c	w1c	w1c

Table 60. SARADC_4 register definitions (continued)

Name		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
		16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
ICIPR0	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	EOC_CH4	EOC_CH3	EOC_CH2	EOC_CH1	EOC_CH0
	W												w1c	w1c	w1c	w1c	w1c
IMR	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	MSKEOCTU	MSKJEOC	MSKJECH	MSKNEOC	MSKNECH
	W																
ICIMR0	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	IM_CH4	IM_CH3	IM_CH2	IM_CH1	IM_CH0
	W																
WTISR	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	WDG3H	WDG3L	WDG2H	WDG2L	WDG1H	WDG1L	WDG0H	WDG0L
	W									w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
WTIMR	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	MSKWDG3H	MSKWDG3L	MSKWDG2H	MSKWDG2L	MSKWDG1H	MSKWDG1L	MSKWDG0H	MSKWDG0L
	W																
DMAE	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	DCLR	DMAEN
	W																

Table 60. SARADC_4 register definitions (continued)

Name		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
		16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
ICDSR0	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	DS_CH4	DS_CH3	DS_CH2	DS_CH1	DS_CH0
	W																
WTHRHLR0 –WTHRHLR3	R	0	0	0	0	THRH											
	W																
	R	0	0	0	0	THRL											
	W																
CTR0–3	R	CRES	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	PRECHG				INPSAMP							
	W																
ICNCMR0	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	NCE_CH4	NCE_CH3	NCE_CH2	NCE_CH1	NCE_CH0
	W																
ICJCMR0	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	JCE_CH4	JCE_CH3	JCE_CH2	JCE_CH1	JCE_CH0
	W																
PDEDR	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	PDED							
	W																
ICDR0–15	R	0	REFSEL	0	0	PCE	0	CTSEL		0	0	0	0	VALID	OVERW	RESULT	
	W																
	R	0	0	0	0	CDATA											
	W																

Table 60. SARADC_4 register definitions (continued)

Name		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
		16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
ICWSELR0	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	WSEL_CH4	
	W																
	R	0	0	WSEL_CH3		0	0	WSEL_CH2		0	0	WSEL_CH1		0	0	WSEL_CH0	
	W																
ICWENR0	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	WEN_CH4	WEN_CH3	WEN_CH2	WEN_CH1	WEN_CH0
	W																
ICAWORR0	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	AWOR_CH4	AWOR_CH3	AWOR_CH2	AWOR_CH1	AWOR_CH0
	W												w1c	w1c	w1c	w1c	w1c

Table 61. SARADC_4 test channel register definitions

Name		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
		16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
TCIPR	R	EOC_CH127	EOC_CH126	EOC_CH125	EOC_CH124	0	0	EOC_CH121	EOC_CH120	0	0	0	0	0	0	0	0
	W	w1c	w1c	w1c	w1c			w1c	w1c								
	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
TCIMR	R	IM_CH127	IM_CH126	IM_CH125	IM_CH124	0	0	IM_CH121	IM_CH120	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																

Table 61. SARADC_4 test channel register definitions (continued)

Name		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
		16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
TCDSR	R	DS_CH127	DS_CH126	DS_CH125	DS_CH124	0	0	DS_CH121	DS_CH120	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
TCNCMR	R	NCE_CH127	NCE_CH126	NCE_CH125	NCE_CH124	0	0	NCE_CH121	NCE_CH120	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
TCJCMR	R	JCE_CH127	JCE_CH126	JCE_CH125	JCE_CH124	0	0	JCE_CH121	JCE_CH120	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
TCWSELR3	R	0	0	WSEL_CH127			0	WSEL_CH126			0	WSEL_CH125			0	WSEL_CH124	
	W																
	R	0	0	0	0	0	0	0	0	0	0	WSEL_CH121			0	WSEL_CH120	
	W																
TCWENR	R	WEN_CH127	WEN_CH126	WEN_CH125	WEN_CH124	0	0	WEN_CH121	WEN_CH120	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
TCAWORR	R	AWOR_CH127	AWOR_CH126	AWOR_CH125	AWOR_CH124	0	0	AWOR_CH121	AWOR_CH120	0	0	0	0	0	0	0	0
	W	w1c	w1c	w1c	w1c			w1c	w1c								
	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																

Table 61. SARADC_4 test channel register definitions (continued)

Name		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
		16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
TCCAPR7	R	ESIC_TCH127	ICSEL_TCH127							ESIC_TCH126	ICSEL_TCH126						
	W																
	R	ESIC_TCH125	ICSEL_TCH125							ESIC_TCH124	ICSEL_TCH124						
	W																
TCDR120–121, TCDR124–127	R	0	0	0	0	PCE	0	CTSEL		0	0	0	0	VALID	OVERW	RESULT	
	W																
	R	0	0	0	0	CDATA											
	W																

7.6.2.1.8 SAR ADC_5 register definitions

Table 62. SARADC_5 register definitions

Name		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
		16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
MCR	R	OWREN	WLSIDE	MODE	0	NSTAR	0	0	0	JSTART	0	0	0	0	0	0	0
	W																
	R	WTRIGOUT	0	0	0	0	0	0	0	ABORTCHAIN	ABORT	0	FRZ	0	0	0	PWDN
	W																
MSR	R	0	0	0	0	NSTART	0	0	0	JSTART	0	0	0	0	JABORTCHAIN	0	CTUSTART
	W																
	R	CHADDR								0	0	0	0	0	ADCSTATUS		
	W																

Table 62. SARADC_5 register definitions (continued)

Name		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
		16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
ISR	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	CTUTRGERR	EOCTU	JEOC	JECH	NEOC	NECH
	W											w1c	w1c	w1c	w1c	w1c	w1c
ICIPR0	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	EOC_CH4	EOC_CH3	EOC_CH2	EOC_CH1	EOC_CH0
	W												w1c	w1c	w1c	w1c	w1c
IMR	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	MSKEOCTU	MSKJEOC	MSKJECH	MSKNEOC	MSKNECH
	W																
ICIMR0	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	IM_CH4	IM_CH3	IM_CH2	IM_CH1	IM_CH0
	W																
WTISR	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	WDG3H	WDG3L	WDG2H	WDG2L	WDG1H	WDG1L	WDG0H	WDG0L
	W									w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c

Table 62. SARADC_5 register definitions (continued)

Name		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
		16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
WTIMR	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	MSKWDG3H	MSKWDG3L	MSKWDG2H	MSKWDG2L	MSKWDG1H	MSKWDG1L	MSKWDG0H	MSKWDG0L
	W																
DMAE	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	DCLR	DMAEN
	W																
ICDSR0	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	DS_CH4	DS_CH3	DS_CH2	DS_CH1	DS_CH0
	W																
WTHRHLR0 –WTHRHLR3	R	0	0	0	0	THRH											
	W																
	R	0	0	0	0	THRL											
	W																
CTR0–3	R	CRES	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	PRECHG				INPSAMP							
	W																
ICNCMR0	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	NCE_CH4	NCE_CH3	NCE_CH2	NCE_CH1	NCE_CH0
	W																
ICJCMR0	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	JCE_CH4	JCE_CH3	JCE_CH2	JCE_CH1	JCE_CH0
	W																

Table 62. SARADC_5 register definitions (continued)

Name		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
		16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
PDEDR	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	PDED							
	W																
ICDR0–15	R	0	REFSEL	0	0	PCE	0	CTSEL		0	0	0	0	VALID	OVERW	RESULT	
	W																
	R	0	0	0	0	CDATA											
	W																
ICWSELR0	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	WSEL_CH4	
	W																
	R	0	0	WSEL_CH3		0	0	WSEL_CH2		0	0	WSEL_CH1		0	0	WSEL_CH0	
	W																
ICWENR0	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	WEN_CH4	WEN_CH3	WEN_CH2	WEN_CH1	WEN_CH0
	W																
ICAWORR0	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	AWOR_CH4	AWOR_CH3	AWOR_CH2	AWOR_CH1	AWOR_CH0
	W												w1c	w1c	w1c	w1c	w1c

Table 63. SARADC_5 test channel register definitions

Name		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
		16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
TCIPR	R	EOC_CH127	EOC_CH126	EOC_CH125	EOC_CH124	0	0	EOC_CH121	EOC_CH120	0	0	0	0	0	0	0	0
	W	w1c	w1c	w1c	w1c			w1c	w1c								
	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
TCIMR	R	IM_CH127	IM_CH126	IM_CH125	IM_CH124	0	0	IM_CH121	IM_CH120	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
TCDSR	R	DS_CH127	DS_CH126	DS_CH125	DS_CH124	0	0	DS_CH121	DS_CH120	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
TCNCMR	R	NCE_CH127	NCE_CH126	NCE_CH125	NCE_CH124	0	0	NCE_CH121	NCE_CH120	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
TCJCMR	R	JCE_CH127	JCE_CH126	JCE_CH125	JCE_CH124	0	0	JCE_CH121	JCE_CH120	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
TCWSELR3	R	0	0	WSEL_CH127			0	WSEL_CH126			0	WSEL_CH125			0	WSEL_CH124	
	W																
	R	0	0	0	0	0	0	0	0	0	0	WSEL_CH121			0	WSEL_CH120	
	W																

Table 63. SARADC_5 test channel register definitions (continued)

Name		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
		16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
TCWENR	R	WEN_CH127	WEN_CH126	WEN_CH125	WEN_CH124	0	0	WEN_CH121	WEN_CH120	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
TCAWORR	R	AWOR_CH127	AWOR_CH126	AWOR_CH125	AWOR_CH124	0	0	AWOR_CH121	AWOR_CH120	0	0	0	0	0	0	0	0
	W	w1c	w1c	w1c	w1c			w1c	w1c								
	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
TCCAPR7	R	ESIC_TCH127	ICSEL_TCH127							ESIC_TCH126	ICSEL_TCH126						
	W																
	R	ESIC_TCH125	ICSEL_TCH125							ESIC_TCH124	ICSEL_TCH124						
	W																
TCDR120–121, TCDR124–127	R	0	0	0	0	PCE	0	CTSEL		0	0	0	0	VALID	OVERW	RESULT	
	W																
	R	0	0	0	0	CDATA											
	W																

7.7 Motor control modules

7.7.1 ECTU

This device contains two ECTU, one per each ADC group, modules with the following features:

- Up to 4 ADCs can be controlled by each single CTU
- 8 sequential triggers based on trigger CMP registers (double buffered)
- Command list (64 descriptors double buffered)
- Up to 128 ADC channels per unit
- Single/multiple conversion
- Up to 8 FIFOs for data buffering
- Up to 8 + 1 DMA channels for data transfer and control

7.7.1.1 Cross triggering unit configuration status register 2

Offset: 0x051C Access: User Read

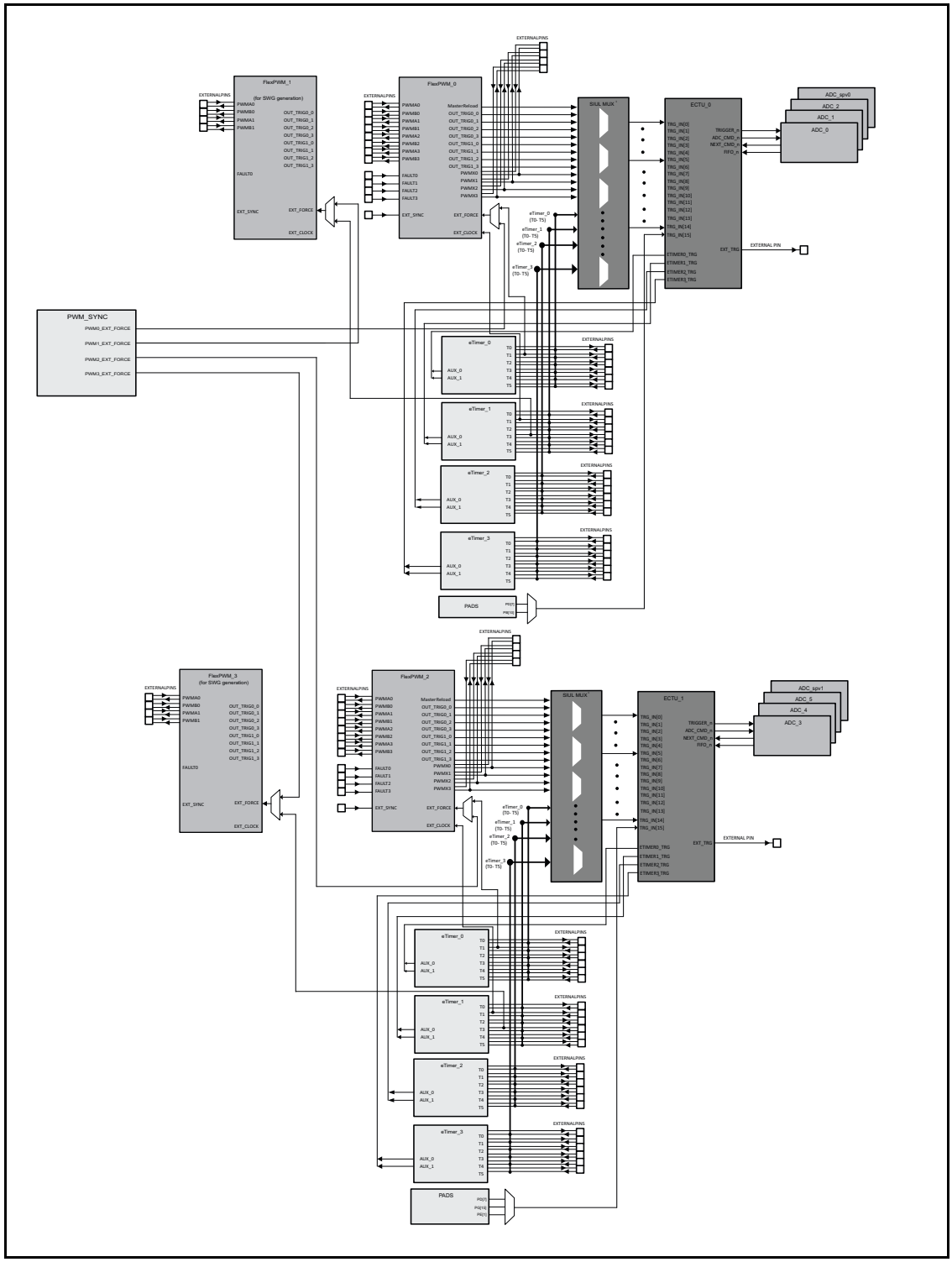
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	FIFO15_DEPTH		FIFO14_DEPTH		FIFO13_DEPTH		FIFO12_DEPTH		FIFO11_DEPTH		FIFO10_DEPTH		FIFO9_DEPTH		FIFO8_DEPTH	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	FIFO7_DEPTH		FIFO6_DEPTH		FIFO5_DEPTH		FIFO4_DEPTH		FIFO3_DEPTH		FIFO2_DEPTH		FIFO1_DEPTH		FIFO0_DEPTH	
W																
Reset	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0

Figure 32. CTU_CFGSR2 register

7.7.1.2 Interaction with other peripherals

Figure 33. Module to module connections



1. For further details, refer to *Table 13: Peripheral multiplexing table description*.

7.7.2 eTimer

7.7.2.1 eTimer instantiation

SPC574Sx includes four 6-channel eTimer units. The following table provides the instantiation information for each module.

Table 64. eTimer instantiation

eTimer module	CH_NUM	AUX_NUM	FAULT_NUM	DMA_NUM	WDOG
eTimer_0	6	2	0	2	Yes
eTimer_1	6	2	0	2	Yes
eTimer_2	6	2	0	2	Yes
eTimer_3	6	2	0	2	Yes

The eTimer fault inputs are not connected to external pins but to ETC outputs. Refer to *Table 13: Peripheral multiplexing table description* for more details.

7.7.3 FlexPWM

7.7.3.1 FlexPWM configurable parameters

This device contains four FlexPWM modules with the following features:

- FlexPWM_0 and FlexPWM_2.
  - 4 Submodules
  - Support of PWM output capabilitites of third output channel per submodule (PWMX)
  - 4 Fault Inputs for each instantiation
- FlexPWM_1 and FlexPWM_3.
  - 2 Submodules
  - Support of PWM output capabilitites of first and second output channels per submodule (PWMA and PWMB)

Table 65 shows the parameters for all FlexPWM instantiations.

Table 65. FlexPWM Device Specific Parameters

Parameter Type	Description	FlexPWM_0 / FlexPWM_2	FlexPWM_1 / FlexPWM_3
SUB_NUM	Number of submodules	4	2
FNUM	Number of fault inputs	4	0
FIFO_DEPTH	Input capture FIFO depth	4	4
INPCAPT_AB	Include input capture on PWMA and PWMB pins	No	Yes
INPCAPT_X	Include input capture on PWMX pins	Yes	No
CYC_WIDTH	Capture value cycles reg width	3	3
FRAC	Include fine control logic	No	No

7.7.3.2 FlexPWM Interrupts

Table 66. Interrupt Summary

Core Interrupt	Interrupt Flag	Interrupt Enable	Name	Description
ipi_int_comp0	CMPF_0	CMPIE_0	Submodule 0 compare interrupt	Compare event has occurred
ipi_int_capt0	CFX1_0, CFX0_0	CFX1IE_0, CFX0IE_0	Submodule 0 input capture interrupt	Input capture event has occurred
ipi_int_reload0	RF_0	RIE_0	Submodule 0 reload interrupt	Reload event has occurred
ipi_int_comp1	CMPF_1	CMPIE_1	Submodule 1 compare interrupt	Compare event has occurred
ipi_int_capt1	CFA1_1, CFA0_1, CFB1_1, CFB0_1,	CFA1IE_1, CFA0IE_1, CFB1IE_1, CFB0IE_1,	Submodule 1 input capture interrupt	Input capture event has occurred
ipi_int_reload1	RF_1	RIE_1	Submodule 1 reload interrupt	Reload event has occurred
ipi_int_comp2	CMPF_2	CMPIE_2	Submodule 2 compare interrupt	Compare event has occurred
ipi_int_capt2	CFA1_2, CFA0_2, CFB1_2, CFB0_2,	CFA1IE_2, CFA0IE_2, CFB1IE_2, CFB0IE_2,	Submodule 2 input capture interrupt	Input capture event has occurred
ipi_int_reload2	RF_2	RIE_2	Submodule 2 reload interrupt	Reload event has occurred
ipi_int_comp3	CMPF_3	CMPIE_3	Submodule 3 compare interrupt	Compare event has occurred
ipi_int_capt3	CFX1_3, CFX0_3	CFX1IE_3, CFX0IE_3	Submodule 3 input capture interrupt	Input capture event has occurred
ipi_int_reload3	RF_3	RIE_3	Submodule 3 reload interrupt	Reload event has occurred
ipi_int_rerr	REF_0	REIE_0	Submodule 0 reload error interrupt	Reload error has occurred
	REF_1	REIE_1	Submodule 1 reload error interrupt	
	REF_2	REIE_2	Submodule 2 reload error interrupt	
	REF_3	REIE_3	Submodule 3 reload error interrupt	
ipi_int_fault	FFLAG	FIE	Fault input interrupt	Fault condition has been detected

7.7.3.3 FlexPWM implemented registers

Each of the following sections shows register definitions for one FlewPWM instance:

- *Section 7.7.3.3.1: FlexPWM_0 register definitions*
- *Section 7.7.3.3.2: FlexPWM_1 register definitions*
- *Section 7.7.3.3.3: FlexPWM_2 register definitions*
- *Section 7.7.3.3.4: FlexPWM_3 register definitions*

7.7.3.3.1 FlexPWM_0 register definitions

FlexPWM_0 implements four submodules then the index n is from 0 to 3.

Table 67. FlexPWM_0 register definitions

Name		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
CNTn	R	CNT															
	W																
INITn	R	INIT															
	W																
CTRL2n	R	DBGEN	WAITEN	INDEP	PWM23_INIT	PWM45_INIT	PWMX_INIT	INIT_SEL		FRCEN	0	FORCE_SEL			RELOAD_SEL	CLK_SEL	
	W										FORCE						
CTRL1n	R	LDFQ				HALF	FULL	DT		0	PRSC			PHSSHFT	LDMOD	0	DBLEN
	W																
VAL0n	R	VAL0															
	W																
VAL1n	R	VAL1															
	W																
VAL2n	R	VAL2															
	W																
VAL3n	R	VAL3															
	W																
VAL4n	R	VAL4															
	W																
VAL5n	R	VAL5															
	W																
OCTRLn	R	PWMA_IN	PWMB_IN	PWMX_IN	0	0	POLA	POLB	POLX	0	0	PWMAFS		PWMBFS		PWMXFS	
	W																

Table 67. FlexPWM_0 register definitions (continued)

Name		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
STSn	R	0	RUF	REF	RF	0	0	0	0	CFX1	CFX0	CMPF					
	W																
INTENn	R	0	0	REIE	RIE	0	0	0	0	CX1	CX0	CMPIE					
	W																
DMAENn	R	0	0	0	0	0	0	VALDE	FAND	CAPTDE		0	0	0	0	CX1	CX0
	W																
TCTRLn	R	0	0	0	0	0	0	0	0	0	0	OUT_TRIG_EN					
	W																
DISMAPn	R	1	1	1	1	DISX				DISB				DISA			
	W																
DTCNT0n	R	0	0	0	0	0	DTCNT0										
	W																
DTCNT1n	R	0	0	0	0	0	DTCNT1										
	W																
CAPTCTRLXn	R	CX1CNT			CX0CNT			CFXWM	EDG	INP_	EDGX1		EDGX0		ONE	ARMX	
	W																
CAPTCMPXn	R	EDGCNTX								EDGCMPX							
	W																
CVAL0n	R	CAPTVAL0															
	W																
CVAL0CYCn	R	0	0	0	0	0	0	0	0	0	0	0	0	0	CVAL0CYC		
	W																
CVAL1n	R	CAPTVAL1															
	W																
CVAL1CYCn	R	0	0	0	0	0	0	0	0	0	0	0	0	0	CVAL1CYC		
	W																
OUTEN	R	0	0	0	0	0	0	0	0	0	0	0	0	PWMX_EN			
	W																
MASK	R	0	0	0	0	0	0	0	0	0	0	0	0	MASKX			
	W	UPDATE_MASK															
SWCOUT	R	0	0	0	0	0	0	0	0	OUT23_3	OUT45_3	OUT23_2	OUT45_2	OUT23_1	OUT45_1	OUT23_0	OUT45_0
	W																

Table 67. FlexPWM_0 register definitions (continued)

Name		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
DTSRCSEL	R	SEL23_3		SEL45_3		SEL23_2		SEL45_2		SEL23_1		SEL45_1		SEL23_0		SEL45_0	
	W																
MCTRL	R	IPOL				RUN				0	0	0	0	LDOK			
	W									CLDOK							
FCTRL	R	FLVL				FAUTO				FSAFE				FIE			
	W																
FSTS	R	0	0	0	FTEST	FFPIN				FFULL				FFLAG			
	W																
FFILT	R	GSTR	0	0	0	0	FILT_CNT			FILT_PER							
	W																

7.7.3.3.2 FlexPWM_1 register definitions

FlexPWM_1 implements two submodules then the index n is from 0 to 1.

Table 68. FlexPWM_1 register definitions

Name		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
CNTn	R	CNT															
	W																
INITn	R	INIT															
	W																
CTRL2n	R	DBGEN	WAITEN	INDEP	PWM23_INIT	PWM45_INIT	PWMX_INIT	INIT_SEL		FRCEN	0	FORCE_SEL			RELOAD_SEL	CLK_SEL	
	W										FORCE						
CTRL1n	R	LDFQ				HALF	FULL	DT		0	PRSC			PHSSHFT	LDMOD	0	DBLEN
	W																
VAL0n	R	VAL0															
	W																
VAL1n	R	VAL1															
	W																
VAL2n	R	VAL2															
	W																
VAL3n	R	VAL3															
	W																

Table 68. FlexPWM_1 register definitions (continued)

Name		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
VAL4n	R	VAL4															
	W																
VAL5n	R	VAL5															
	W																
OCTRLn	R	PWMA_IN	PWMB_IN	0(1)	0	0	POLA	POLB	0(1)	0	0	PWMAFS		PWMBFS		0(1)	0(1)
	W																
STSn	R	0	RUF	REF	RF	CFA1	CFA0	CFB1	CFB0	0	0	CMPF					
	W																
INTENn	R	0	0	REIE	RIE	CA1IE	CA0IE	CB1IE	CB0IE	0	0	CMPIE					
	W																
DMAENn	R	0	0	0	0	0	0	VALDE	FAND	CAPTDE		CA1	CA0	CB1	CB0	0	0
	W																
TCTRLn	R	0	0	0	0	0	0	0	0	0	0	OUT_TRIG_EN					
	W																
DISMAPn	R	1	1	1	1	DISX				DISB				DISA			
	W																
DTCNT0n	R	0	0	0	0	0	DTCNT0										
	W																
DTCNT1n	R	0	0	0	0	0	DTCNT1										
	W																
CAPTCTRLAn	R	CA1CNT			CA0CNT			CFAWM		EDGCNTA_EN	INP_SELA	EDGA1		EDGA0		ONESHOTA	ARMA
	W																
CAPTCMPAn	R	EDGCNTA								EDGCMPA							
	W																
CAPTCTRLBn	R	CB1CNT			CB0CNT			CFBWM		EDGCNTB_EN	INP_SELB	EDGB1		EDGB0		ONESHOTB	ARMB
	W																
CAPTCMPBn	R	EDGCNTB								EDGCMPB							
	W																

Table 68. FlexPWM_1 register definitions (continued)

Name		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
CVAL2n	R	CAPTVAL2															
	W																
CVAL2CYCn	R	0	0	0	0	0	0	0	0	0	0	0	0	0	CVAL2CYC		
	W																
CVAL3n	R	CAPTVAL3															
	W																
CVAL3CYCn	R	0	0	0	0	0	0	0	0	0	0	0	0	0	CVAL3CYC		
	W																
CVAL4n	R	CAPTVAL4															
	W																
CVAL4CYCn	R	0	0	0	0	0	0	0	0	0	0	0	0	0	CVAL4CYC		
	W																
CVAL5n	R	CAPTVAL5															
	W																
CVAL5CYCn	R	0	0	0	0	0	0	0	0	0	0	0	0	0	CVAL5CYC		
	W																
OUTEN	R	0	0	0	0	PWMA_EN				PWMB_EN				0	0	0	0
	W																
MASK	R	0	0	0	0	MASKA				MASKB				0	0	0	0
	W	UPDATE_MASK															
SWCOUT	R	0	0	0	0	0	0	0	0	0	0	0	0	OUT23_1	OUT45_1	OUT23_0	OUT45_0
	W																
DTSRCSEL	R	0	0	0	0	0	0	0	0	SEL23_1		SEL45_1		SEL23_0		SEL45_0	
	W																
MCTRL	R	0	0	IPOL		0	0	RUN		0	0	0	0	0	0	LDOK	
	W											CLDOK					
FCTRL	R	0	0	0	FLVL	0	0	0	FAUTO	0	0	0	FSAFE	0	0	0	FIE
	W																
FSTS	R	0	0	0	FTEST	0	0	0	FFPIN	0	0	0	FFULL	0	0	0	FFLAG
	W																
FFILT	R	GSTR	0	0	0	0	FILT_CNT			FILT_PER							
	W																

1. This bit is writable but has no impact.

7.7.3.3.3 FlexPWM_2 register definitions

FlexPWM_2 implements four submodules then the index n is from 0 to 3.

Table 69. FlexPWM_2 register definitions

Name		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
CNTn	R	CNT															
	W																
INITn	R	INIT															
	W																
CTRL2n	R	DBGEN	WAITEN	INDEP	PWM23_INIT	PWM45_INIT	PWMX_INIT	INIT_SEL		FRCEN	0	FORCE_SEL			RELOAD_SEL	CLK_SEL	
	W										FORCE						
CTRL1n	R	LDFQ				HALF	FULL	DT		0	PRSC			PHSSHFT	LDMOD	0	DBLEN
	W																
VAL0n	R	VAL0															
	W																
VAL1n	R	VAL1															
	W																
VAL2n	R	VAL2															
	W																
VAL3n	R	VAL3															
	W																
VAL4n	R	VAL4															
	W																
VAL5n	R	VAL5															
	W																
OCTRLn	R	PWMA_IN	PWMB_IN	PWMX_IN	0	0	POLA	POLB	POLX	0	0	PWMAFS		PWMBFS		PWMXFS	
	W																
STSn	R	0	RUF	REF	RF	0	0	0	0	CFX1	CFX0	CMPF					
	W																
INTENn	R	0	0	REIE	RIE	0	0	0	0	CX1	CX0	CMPIE					
	W																

Table 69. FlexPWM_2 register definitions (continued)

Name		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
DMAENn	R	0	0	0	0	0	0	VALDE	FAND	CAPTDE		0	0	0	0	CX1	CX0
	W																
TCTRLn	R	0	0	0	0	0	0	0	0	0	0	OUT_TRIG_EN					
	W																
DISMAPn	R	1	1	1	1	DISX				DISB				DISA			
	W																
DTCNT0n	R	0	0	0	0	0	DTCNT0										
	W																
DTCNT1n	R	0	0	0	0	0	DTCNT1										
	W																
CAPTCTRLXn	R	CX1CNT			CX0CNT			CFXWM	EDG	INP_	EDGX1			EDGX0		ONE	ARMX
	W																
CAPTCMPXn	R	EDGCNTX								EDGCMPX							
	W																
CVAL0n	R	CAPTVAL0															
	W																
CVAL0CYCn	R	0	0	0	0	0	0	0	0	0	0	0	0	0	CVAL0CYC		
	W																
CVAL1n	R	CAPTVAL1															
	W																
CVAL1CYCn	R	0	0	0	0	0	0	0	0	0	0	0	0	0	CVAL1CYC		
	W																
OUTEN	R	0	0	0	0	0	0	0	0	0	0	0	0	PWMX_EN			
	W																
MASK	R	0	0	0	0	0	0	0	0	0	0	0	0	MASKX			
	W	UPDATE_MASK															
SWCOUT	R	0	0	0	0	0	0	0	0	OUT23_3	OUT45_3	OUT23_2	OUT45_2	OUT23_1	OUT45_1	OUT23_0	OUT45_0
	W																
DTSRCSEL	R	SEL23_3		SEL45_3		SEL23_2		SEL45_2		SEL23_1		SEL45_1		SEL23_0		SEL45_0	
	W																
MCTRL	R	IPOL				RUN				0	0	0	0	LDOK			
	W									CLDOK							

Table 69. FlexPWM_2 register definitions (continued)

Name		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
FCTRL	R	FLVL				FAUTO				FSAFE				FIE			
	W																
FSTS	R	0	0	0	FTEST	FFPIN				FFULL				FFLAG			
	W																
FFILT	R	GSTR	0	0	0	0	FILT_CNT			FILT_PER							
	W																

7.7.3.3.4 FlexPWM_3 register definitions

FlexPWM_3 implements two submodules then the index n is from 0 to 1.

Table 70. FlexPWM_3 register definitions

Name		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
CNTn	R	CNT															
	W																
INITn	R	INIT															
	W																
CTRL2n	R	DBGEN	WAITEN	INDEP	PWM23_INIT	PWM45_INIT	PWMX_INIT	INIT_SEL		FRCEN	0	FORCE_SEL			RELOAD_SEL	CLK_SEL	
	W										FORCE						
CTRL1n	R	LDFQ				HALF	FULL	DT		0	PRSC			PHSSHFT	LDMOD	0	DBLEN
	W																
VAL0n	R	VAL0															
	W																
VAL1n	R	VAL1															
	W																
VAL2n	R	VAL2															
	W																
VAL3n	R	VAL3															
	W																
VAL4n	R	VAL4															
	W																
VAL5n	R	VAL5															
	W																

Table 70. FlexPWM_3 register definitions (continued)

Name		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
OCTRLn	R	PWMA_IN	PWMB_IN	0(1)	0	0	POLA	POLB	0(1)	0	0	PWMAFS		PWMBFS		0(1)	0(1)
	W																
STSn	R	0	RUF	REF	RF	CFA1	CFA0	CFB1	CFB0	0	0	CMPF					
	W																
INTENn	R	0	0	REIE	RIE	CA1IE	CA0IE	CB1IE	CB0IE	0	0	CMPIE					
	W																
DMAENn	R	0	0	0	0	0	0	VALDE	FAND	CAPTDE		CA1	CA0	CB1	CB0	0	0
	W																
TCTRLn	R	0	0	0	0	0	0	0	0	0	0	OUT_TRIG_EN					
	W																
DISMAPn	R	1	1	1	1	DISX				DISB				DISA			
	W																
DTCNT0n	R	0	0	0	0	0	DTCNT0										
	W																
DTCNT1n	R	0	0	0	0	0	DTCNT1										
	W																
CAPTCTRLAn	R	CA1CNT			CA0CNT			CFAWM	EDGCNTA_EN	INP_SELA	EDGA1		EDGA0		ONESHOTA	AR MA	
	W																
CAPTCMPAn	R	EDGCNTA								EDGCMPA							
	W																
CAPTCTRLBn	R	CB1CNT			CB0CNT			CFBWM	EDGCNTB_EN	INP_SELB	EDGB1		EDGB0		ONESHOTB	AR MB	
	W																
CAPTCMPBn	R	EDGCNTB								EDGCMPB							
	W																
CVAL2n	R	CAPTVAL2															
	W																
CVAL2CYCn	R	0	0	0	0	0	0	0	0	0	0	0	0	0	CVAL2CYC		
	W																

Table 70. FlexPWM_3 register definitions (continued)

Name		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
CVAL3n	R	CAPTVAL3															
	W																
CVAL3CYCn	R	0	0	0	0	0	0	0	0	0	0	0	0	0	CVAL3CYC		
	W																
CVAL4n	R	CAPTVAL4															
	W																
CVAL4CYCn	R	0	0	0	0	0	0	0	0	0	0	0	0	0	CVAL4CYC		
	W																
CVAL5n	R	CAPTVAL5															
	W																
CVAL5CYCn	R	0	0	0	0	0	0	0	0	0	0	0	0	0	CVAL5CYC		
	W																
OUTEN	R	0	0	0	0	PWMA_EN				PWMB_EN				0	0	0	0
	W																
MASK	R	0	0	0	0	MASKA				MASKB				0	0	0	0
	W	UPDATE_MASK															
SWCOUT	R	0	0	0	0	0	0	0	0	0	0	0	0	OUT23_1	OUT45_1	OUT23_0	OUT45_0
	W																
DTSRCSEL	R	0	0	0	0	0	0	0	0	SEL23_1		SEL45_1		SEL23_0		SEL45_0	
	W																
MCTRL	R	0	0	IPOL		0	0	RUN		0	0	0	0	0	0	LDOK	
	W											CLDOK					
FCTRL	R	0	0	0	FLVL	0	0	0	FAUTO	0	0	0	FSAFE	0	0	0	FIE
	W																
FSTS	R	0	0	0	FTEST	0	0	0	FFPIN	0	0	0	FFULL	0	0	0	FFLAG
	W																
FFILT	R	GSTR	0	0	0	0	FILT_CNT			FILT_PER							
	W																

1. This bit is writable but has no impact.

7.7.3.4 FlexPWM DMA

Table 71. DMA Summary

DMA Request	DMA Enable	Name	Description
Submodule 0 read request	CX0DE_0	Capture FIFO X0 read request	CVAL0 contains a value to be read
	CX1DE_0	Capture FIFO X1 read request	CVAL1 contains a value to be read
	CAPTDE_0	Capture FIFO read request source select	Selects source of read DMA request
Submodule 0 write request	VALDE_0	VALx write request	VALx registers need to be updated
Submodule 1 read request	CA0DE_1	Capture FIFO A0 read request	CVAL2 contains a value to be read
	CA1DE_1	Capture FIFO A1 read request	CVAL3 contains a value to be read
	CB0DE_1	Capture FIFO B0 read request	CVAL4 contains a value to be read
	CB1DE_1	Capture FIFO B1 read request	CVAL5 contains a value to be read
	CAPTDE_1	Capture FIFO read request source select	Selects source of read DMA request
Submodule 1 write request	VALDE_1	VALx write request	VALx registers need to be updated
Submodule 2 read request	CX0DE_2	Capture FIFO X0 read request	CVAL0 contains a value to be read
	CX1DE_2	Capture FIFO X1 read request	CVAL1 contains a value to be read
	CAPTDE_2	Capture FIFO read request source select	Selects source of read DMA request
Submodule 2 write request	VALDE_2	VALx write request	VALx registers need to be updated
Submodule 3 read request	CA0DE_3	Capture FIFO A0 read request	CVAL2 contains a value to be read
	CA1DE_3	Capture FIFO A1 read request	CVAL3 contains a value to be read
	CB0DE_3	Capture FIFO B0 read request	CVAL4 contains a value to be read
	CB1DE_3	Capture FIFO B1 read request	CVAL5 contains a value to be read
	CAPTDE_3	Capture FIFO read request source select	Selects source of read DMA request
Submodule 3 write request	VALDE_3	VALx write request	VALx registers need to be updated

7.8 Timers

Four types of timers and a complex timer subsystem are implemented on SPC574Sx.

7.8.1 System Timer Module (STM) configuration

SPC574Sx includes one STM.

Table 72. System Timer Module (STM) instances

Instance	Description	Number of channels	Number of registers (32-bit)
STM_0	Intended for Main Core_0 in the high-speed computational clock domain	4	14

7.8.2 Software Watchdog Timer (SWT) configuration

SPC574Sx includes one SWT.

Table 73. Software Watchdog Timer (SWT) instances

Instance	Description
SWT	Intended for Main Core in the high-speed computational clock domain.

The SWT has the following features:

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

7.8.2.1 Reset assertion

The SWT can assert a reset when the watchdog timer expires. This reset causes a system reset equivalent to assertion of the RESET pin.

7.8.2.1.1 Default configuration

On SPC574Sx, the SWT comes out of reset with the default reset values summarized in *Table 74*. Minimum timeout value is 256 cycles.

Table 74. Register reset values

SWT instance	Register	Reset value
SWT2	SWT Control Register (SWT_CR)(1)	0xFF00_011B
SWT2	SWT Time-out (SWT_TO)	0x0003_FDE0

1. The SWT can be disabled by accepting a DCI-driven signal to clear the SWT_CR[WEN] bit. For SWT2, the default reset value for that bit is ‘1’, but the DCI can override that default value by setting it to ‘0’. If the signal is de-asserted, there is no change to the SWT_CR[WEN] value. (The DCI can only disable the SWT; it cannot enable it.) See the DCI chapter for details on the system watchdog control.

7.8.3 Periodic Interupt Timer (PIT) configuration

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

Figure 34. PIT configuration

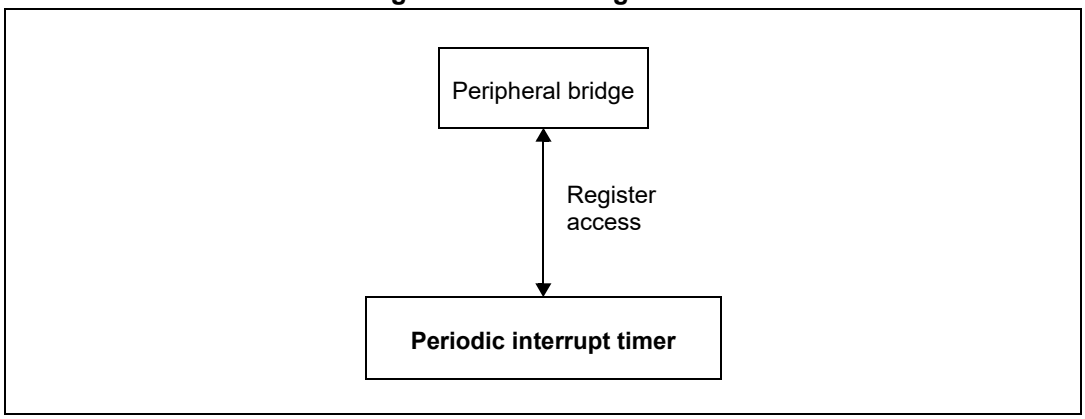


Table 75. Reference links to related information

Topic	Related module	Reference
Full description	PIT	*Chapter 41: Periodic Interrupt Timer (PIT)*
System memory map	—	*Chapter 5: Memory map*
Clocking	—	*Chapter 24: Clocking*
Power management	—	*Chapter 10: Power management*
DMA channel assignments	Direct memory access channel multiplexer	*Chapter 21: DMA channel multiplexer (DMACHMUX)*

7.8.3.1 PIT instantiation

Table 76. PIT information

PIT module instances	PIT channels	Description
1	8	The PIT has eight standard channels. All PIT registers are cleared by any reset.

7.8.3.2 PIT/DMA periodic trigger assignments

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

7.8.3.3 PIT clocking

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

7.9 Communication interfaces

7.9.1 CAN subsystem configuration

7.9.1.1 CAN nodes

Table 77 lists the different CAN nodes.

Table 77. CAN bus controllers

Subsystem	Type	Instance	CAN FD support up to 64 data byte (M_CAN Rev 3.x)	Debug on CAN support	Clock calibration(1)
CAN_SUBSYS_0	M_CAN	MCAN_0	Yes	Yes	Yes
		MCAN_2	Yes	Yes	Yes
CAN_SUBSYS_1	M_CAN	MCAN_1	Yes	Yes	Yes

1. Clock calibration based on M_CAN_0 messages for CAN_SUBSYS_0.
Clock calibration based on M_CAN_1 messages for CAN_SUBSYS_1.

7.9.2 DSPI configuration

This section summarizes how the module has been configured in the chip.

DSPI module features:

- Transmission speed: 14 MHz
- Frame size: Programmable from 4-bits up to 32-bits in extended mode. Frame size is set inside the DSPI_CTARn descriptor. Extended DSPI mode is set inside the DSPI_MCR register
- Chip selects: 8
- FIFO depths: 4

The DSPI module operates in SPI configuration. The DSPI module cannot be configured to operate in DSI or CSI modes.

Figure 35. DSPI configuration

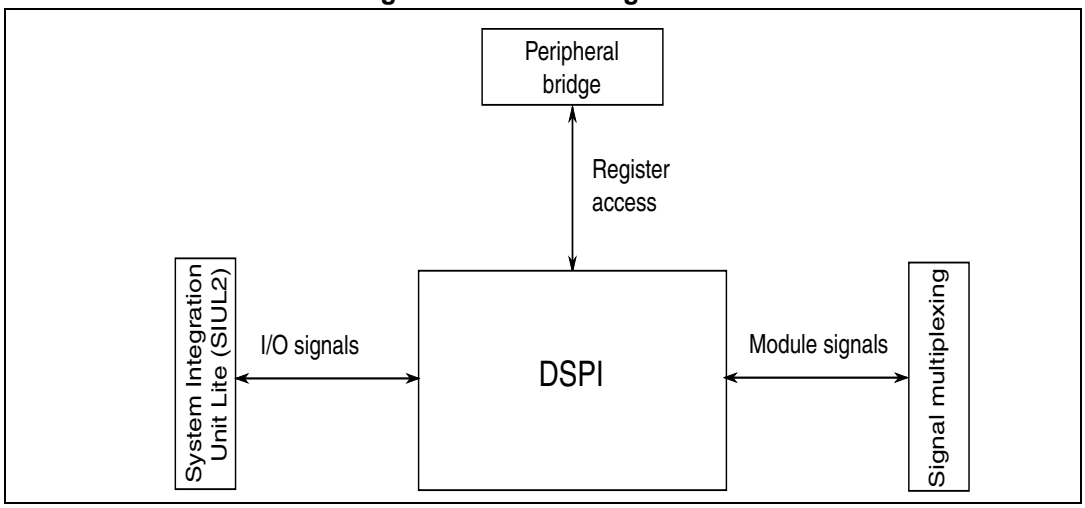


Table 78. Reference links to related information

Topic	Related module	Reference
Full description	DSPI	*Chapter 43: Deserial Serial Peripheral Interface (DSPI)*
System memory map	—	*Chapter 5: Memory map*
Clocking	—	*Chapter 24: Clocking*
Signal multiplexing	Port control	*Chapter 4: Signal description*
System module	System Integration Unit Lite	*Chapter 14: System Integration Unit Lite2 (SIUL2)*
DMA channel assignments	Direct memory access channel multiplexer	*Chapter 21: DMA channel multiplexer (DMACHMUX)*

7.9.2.1 DSPI instantiation

This device contains four DSPI modules. The following table provides the instantiation information for each module.

Table 79. DSPI instantiation

DSPI module	CTARs	TX FIFO depth	RX FIFO depth	PCS signals	Data Serialization Support
DSPI_0	8	4	4	PCS[7:0]	—
DSPI_1	8	4	4	PCS[7:0]	—
DSPI_2	8	4	4	PCS[7:0]	—
DSPI_3	8	4	4	PCS[7:0]	—

7.9.2.2 DSPI half duplex operation

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

7.9.2.3 DSPI registers

The following table shows the absolute addresses of DSPI module registers in this MCU.

Table 80. DSPI registers memory map

Absolute address (hex)	Module
FBE7_0000	DSPI_2
FBE7_4000	DSPI_3
FFE7_0000	DSPI_0
FFE7_4000	DSPI_1

The following registers are **not** implemented on this MCU:

- DSPI DSI Configuration Register 0 (DSPIx_DSICR0)
- DSPI DSI Serialization Data Register 0 (DSPIx_SDR0)
- DSPI DSI Alternate Serialization Data Register 0 (DSPIx_ASDR0)
- DSPI DSI Transmit Comparison Register 0 (DSPIx_COMPR0)
- DSPI DSI Deserialization Data Register 0 (DSPIx_DDR0)
- DSPI DSI Configuration Register 1 (DSPIx_DSICR1)
- DSPI DSI Serialization Source Select Register 0 (DSPIx_SSR0)
- DSPI DSI Parallel Input Select Register 0 (DSPIx_PISR0)
- DSPI DSI Parallel Input Select Register 1 (DSPIx_PISR1)
- DSPI DSI Parallel Input Select Register 2 (DSPIx_PISR2)
- DSPI DSI Parallel Input Select Register 3 (DSPIx_PISR3)
- DSPI DSI Deserialized Data Interrupt Mask Register 0 (DSPIx_DIMR0)
- DSPI DSI Deserialized Data Polarity Interrupt Register 0 (DSPIx_DPIR0)
- DSPI DSI Serialization Data Register 1 (DSPIx_SDR1)
- DSPI DSI Alternate Serialization Data Register 1 (DSPIx_ASDR1)
- DSPI DSI Transmit Comparison Register 1 (DSPIx_COMPR1)
- DSPI DSI Deserialization Data Register 1 (DSPIx_DDR1)
- DSPI DSI Serialization Source Select Register 1 (DSPIx_SSR1)
- DSPI DSI Parallel Input Select Register 4 (DSPIx_PISR4)
- DSPI DSI Parallel Input Select Register 5 (DSPIx_PISR5)
- DSPI DSI Parallel Input Select Register 6 (DSPIx_PISR6)
- DSPI DSI Parallel Input Select Register 7 (DSPIx_PISR7)
- DSPI DSI Deserialized Data Interrupt Mask Register 1 (DSPIx_DIMR1)
- DSPI DSI Deserialized Data Polarity Interrupt Register 1 (DSPIx_DPIR1)

7.9.3 FlexRay Configuration

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

7.9.3.1 FlexRay signals pin assignment

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

Table 81. Flexray pin names

Signal description	Pin name
FR_A_RX (Receive Data Channel A)	RXDA
FR_A_TX (Transmit Data Channel A)	CA_TX
$\overline{\text{FR_A_TX_EN}}$ (Transmit Enable Channel A)	CA_TR_EN
FR_B_RX (Receive Data Channel B)	RXDB

Table 81. Flexray pin names (continued)

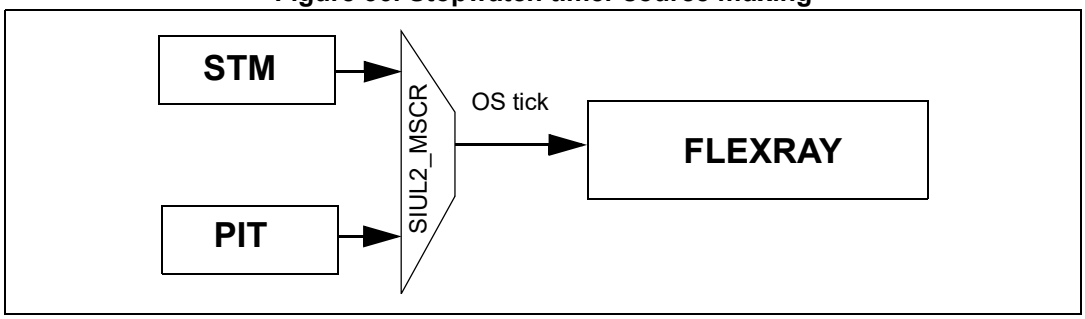
Signal description	Pin name
FR_B_TX (Transmit Data Channel B)	CB_TX
$\overline{\text{FR_B_TX_EN}}$ (Transmit Enable Channel B)	CB_TR_EN
FR_DBG[0] (Debug Strobe Signal 0)	DBG[0]
FR_DBG[1] (Debug Strobe Signal 1)	DBG[1]
FR_DBG[2] (Debug Strobe Signal 2)	DBG[2]
FR_DBG[3] (Debug Strobe Signal 3)	DBG[3]

7.9.3.2 FlexRay Stopwatch timer source muxing

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

- STM0_CMP0
- PIT1_CH0

Figure 36. Stopwatch timer source muxing



7.9.4 SENT Receiver (SRX) configuration

7.9.4.1 SRX channels

Table 82 shows the SENT receiver channels for each instance configured on this chip.

Table 82. SENT channel configuration

Instance	Channels (CH)
SENT 0	3
SENT 1	3

7.9.4.2 SRX registers

Table 83 lists the address offset and memory map for both SENT receiver (SRX) instances for this chip.

Table 83. SENT receiver memory map

Offset	Register name	SENT 0	SENT 1
0x0000	GBL_CTRL— Global Control Register	X	X
0x0004	CHNL_EN—Channel Enable Register	X	X
0x0008	GBL_STATUS—Global Status Register	X	X
0x000C	FMSG_RDY—Fast Message Ready Register	X	X
0x0010	SMSG_RDY—Slow Serial Message Ready Register	X	X
0x0014	Reserved	—	—
0x0018	DATA_CTRL1— Data Control Register 1	X	X
0x001C	Reserved	—	—
0x0020	Reserved	—	—
0x0024	Reserved	—	—
0x0028	FDMA_CTRL—Fast Message DMA Control Register	X	X
0x002C	SDMA_CTRL—Slow Serial Message DMA Control Register	X	X
0x0030	Reserved	—	—
0x0034	FRDY_IE—Fast Message Ready Interrupt Control Register	X	X
0x0038	SRDY_IE—Slow Serial Message Ready Interrupt Control Register	X	X
0x003C	Reserved	—	—
0x0040	DMA_FMSG_DATA—DMA Fast Message Read Register	X	X
0x0044	DMA_FMSG_CRC—DMA Fast Message CRC Read Register	X	X
0x0048	DMA_FMSG_TS—DMA Fast Message Time Stamp Read Register	X	X
0x004C	Reserved	—	—
0x0050	DMA_SMSG_BIT3—DMA Serial Message Read Register (Bit 3)	X	X
0x0054	DMA_SMSG_BIT2—DMA Serial Message Read Register (Bit 2)	X	X
0x0058	DMA_SMSG_TS—DMA Serial Message Time Stamp Read Register	X	X
0x005C	Reserved	—	—
0x0060	CH0_CLK_CTRL—Channel ‘n’ Clock Control Register	X	X
0x0064	CH0_STATUS—Channel ‘n’ Status Register	X	X
0x0068	CH0_CONFIG—Channel ‘n’ Configuration Register	X	X
0x006C	Reserved	—	—
0x0070	CH1_CLK_CTRL—Channel ‘n’ Clock Control Register	X	X
0x0074	CH1_STATUS—Channel ‘n’ Status Register	X	X
0x0078	CH1_CONFIG—Channel ‘n’ Configuration Register	X	X
0x007C	Reserved	—	—
0x0080	CH2_CLK_CTRL—Channel ‘n’ Clock Control Register	X	X
0x0084	CH2_STATUS—Channel ‘n’ Status Register	X	X
0x0088	CH2_CONFIG—Channel ‘n’ Configuration Register	X	X

Table 83. SENT receiver memory map (continued)

Offset	Register name	SENT 0	SENT 1
0x008C – 0x015F	Reserved	—	—
0x0160	CH0_FMSG_DATA—Channel ‘n’ Fast Message Register	X	X
0x0164	CH0_FMSG_CRC—Channel ‘n’ Fast Message CRC Register	X	X
0x0168	CH0_FMSG_TS—Channel ‘n’ Fast Message Time Stamp Register	X	X
0x016C	CH0_SMSG_BIT3—Channel ‘n’ Serial Message Register (Bit 3)	X	X
0x0170	CH0_SMSG_BIT2—Channel ‘n’ Serial Message Register (Bit 2)	X	X
0x0174	CH0_SMSG_TS—Channel ‘n’ Serial Message Time Stamp Register	X	X
0x0178	CH1_FMSG_DATA—Channel ‘n’ Fast Message Register	X	X
0x017C	CH1_FMSG_CRC—Channel ‘n’ Fast Message CRC Register	X	X
0x0180	CH1_FMSG_TS—Channel ‘n’ Fast Message Time Stamp Register	X	X
0x0184	CH1_SMSG_BIT3—Channel ‘n’ Serial Message Register (Bit 3)	X	X
0x0188	CH1_SMSG_BIT2—Channel ‘n’ Serial Message Register (Bit 2)	X	X
0x018C	CH1_SMSG_TS—Channel ‘n’ Serial Message Time Stamp Register	X	X
0x0190	CH2_FMSG_DATA—Channel ‘n’ Fast Message Register	X	X
0x0194	CH2_FMSG_CRC—Channel ‘n’ Fast Message CRC Register	X	X
0x0198	CH2_FMSG_TS—Channel ‘n’ Fast Message Time Stamp Register	X	X
0x019C	CH2_SMSG_BIT3—Channel ‘n’ Serial Message Register (Bit 3)	X	X
0x01A0	CH2_SMSG_BIT2—Channel ‘n’ Serial Message Register (Bit 2)	X	X
0x01A4	CH2_SMSG_TS—Channel ‘n’ Serial Message Time Stamp Register	X	X
0x01A8 – 0x02E0	Reserved	—	—

7.9.5 LinFlexD

7.9.5.1 LINFlexD – Configurations

SPC574Sx has four instances of LINFlexD. *Table 84* lists the configuration for each instance.

Table 84. LINFlexD configurations

Description	LINFlexD_0	LINFlexD_1	LINFlexD_2	LINFlexD_3
Number of filters implemented	16	0		
Number of Tx DMA channels	1(1)			
Number of Rx DMA channels	1(2)			
LIN operation mode	Master/Slave (slave = 1)	Master (slave = 0)		
Value of autosynchronization	1	0		

1. TX_CH_NUM = 0, Number of TX DMA channels = $2^{TX_CH_NUM}$
2. RX_CH_NUM = 0, Number of RX DMA channels = $2^{RX_CH_NUM}$

7.9.5.2 LINFlexD implemented registers

Table 85 shows the registers implemented for each LINFlexD instance on the SPC574Sx chip.

Table 85. LINFlexD register comparison map

Register description	LINFlexD_0	LINFlexD_1	LINFlexD_2	LINFlexD_3
	Address offset (hex)			
LIN Control register 1 (LINCR1)	00	00	00	00
LIN Interrupt Enable register (LINIER)	04	04	04	04
LIN Status register (LINSR)	08	08	08	08
LIN Error Status register (LINESR)	0C	0C	0C	0C
UART Mode Control register (UARTCR)	10	10	10	10
UART Mode Status register (UARTSR)	14	14	14	14
LIN Timeout Control Status register (LINTCSR)	18	18	18	18
LIN Output Compare register (LINOCR)	1C	1C	1C	1C
LIN Timeout Control register (LINTOCR)	20	20	20	20
LIN Fractional Baud Rate register (LINFBRR)	24	24	24	24
LIN Integer Baud Rate register (LINIBRR)	28	28	28	28
LIN Checksum Field register (LINCFR)	2C	2C	2C	2C
LIN Control register 2 (LINCR2)	30	30	30	30
Buffer Identifier register (BIDR)	34	34	34	34
Buffer Data Register Least Significant (BDRL)	38	38	38	38
Buffer Data Register Most Significant (BDRM)	3C	3C	3C	3C
Identifier Filter Enable register (IFER)	40	40	40	40
Identifier Filter Match index (IFMI)	44	44	44	44
Identifier Filter Mode register (IFMR)	48	48	48	48
Identifier Filter Control registers (IFCR0)	4C	—	—	—
Identifier Filter Control registers (IFCR1)	50	—	—	—
Identifier Filter Control registers (IFCR2)	54	—	—	—
Identifier Filter Control registers (IFCR3)	58	—	—	—
Identifier Filter Control registers (IFCR4)	5C	—	—	—
Identifier Filter Control registers (IFCR5)	60	—	—	—
Identifier Filter Control registers (IFCR6)	64	—	—	—
Identifier Filter Control registers (IFCR7)	68	—	—	—
Identifier Filter Control registers (IFCR8)	6C	—	—	—
Identifier Filter Control registers (IFCR9)	70	—	—	—
Identifier Filter Control registers (IFCR10)	74	—	—	—

Table 85. LINFlexD register comparison map (continued)

Register description	LINFlexD_0	LINFlexD_1	LINFlexD_2	LINFlexD_3
	Address offset (hex)			
Identifier Filter Control registers (IFCR11)	78	—	—	—
Identifier Filter Control registers (IFCR12)	7C	—	—	—
Identifier Filter Control registers (IFCR13)	80	—	—	—
Identifier Filter Control registers (IFCR14)	84	—	—	—
Identifier Filter Control registers (IFCR15)	88	—	—	—
Global Control register (GCR)	8C	4C	4C	4C
UART Preset Timeout register (UARTPTO)	90	50	50	50
UART Current Timeout register (UARTCTO)	94	54	54	54
DMA Tx Enable register (DMATXE)	98	58	58	58
DMA Rx Enable register (DMARXE)	9C	5C	5C	5C

7.10 Reset and Boot modules

7.10.1 BAF configuration

7.10.1.1 Search boot header and boot options

Refer to *Section 8.9: SW boot record search (BAF search)* in the *Chapter 8: Reset and boot*.

7.10.1.2 LINFlexD pins configuration

Table 86 shows the device pin configurations for the LINFlexD_0 module.

Table 86. LinFlexD pin configuration

Pads	Initial Serial Boot Mode		Serial Boot Mode after a valid LINFlexD message received
	Function	Pad Configuration	Function
PA[12]	LINFlexD_0 Rx	Input:— pull-down hysteresis TTL voltage level	LINFlexD_0 Rx
PA[13]	LINFlexD_0 Tx	Output:— Push/Pull medium drive no pullup/pulldown	LINFlexD_0 Tx

7.10.2 System Status and Configuration Module (SSCM) configuration

This section summarizes the System Status and Configuration Module (SSCM) configuration in the SPC574Sx device. For a comprehensive description of the SSCM, please refer to the SSCM's dedicated chapter.

7.10.2.1 SSCM instantiation

There is one SSCM instance in the SPC574Sx device.

7.10.2.2 Device-specific features

The device-specific reset values for the SSCM_MEMCONFIG register are shown in *Table 87*.

Table 87. SSCM_MEMCONFIG reset values

Field	Field definition	Value
MREV	Minor Revision	Minor silicon version
JPIN	JTAG Part ID	0A4h

Table 88 shows the address locations that the SSCM uses to search for startup information.

Table 88. SSCM startup information addresses

Description	Value
The location at which the SSCM will look for the UTEST DCF records (UTEST_OFFSET)	0040_0000h
Location of the Boot Assist code in Flash (BAF)	0040_4000h

7.11 Safety modules

7.11.1 CRC – Number of contexts

Table 89 lists the number of contexts available for the current CRC computation of multiple data streams of CRC.

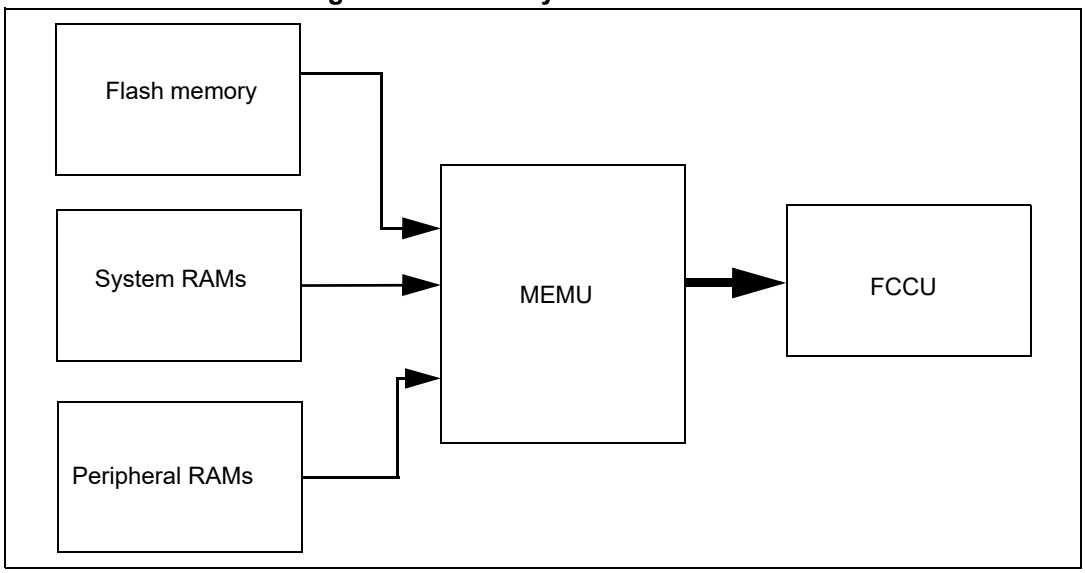
Table 89. CRC contexts

Signal name	Description
CRC_CNTX_NUM (N)	Number of contexts available for the current CRC computation of multiple data streams: 1

7.11.2 MEMU configuration

7.11.2.1 MEMU system connections

Figure 37. MEMU system connections



7.11.2.2 MEMU error sources

Table 90 shows the MEMU error sources for each instance configured on this chip.

Table 90. MEMU error sources

Instance	Value
Number of system RAM unique error sources	29
Number of peripheral RAM unique error sources	6
Number of flash memory unique error sources	2

7.11.2.3 Reporting tables implementation

There are three categories of error reporting for the MEMU:

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

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

- Correctable Error Reporting Table
- Uncorrectable Error Reporting Table

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

Table 91 describes the number of entries in each reporting table for the different error sources.

Table 91. Number of entries in each reporting table

Error source	Number of entries	
	Correctable error reporting table	Uncorrectable error reporting table
System RAM	10	1
Peripheral RAM	2	1
Flash memory	20	1

7.11.2.4 Concurrent overflow register (OFLW) implementation

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

Table 92. Corresponding error sources for system RAM OFLW0 register

OFLW0 bit-field position (bit)(1)	Error sources
0	z4a Data AHB
1	z4a Instruction AHB
2	z4a Instruction Cache
3	z4a Data Cache
4	z4a Data TCM (DMEM)
5	SRAM Memory
6	Decorated z4a TCM
7	Flexray AHB
8	MBIST z4a Data TCM Instance 1
9	MBIST z4a Data TCM Instance 2
10	MBIST z4a Instruction Cache Instance 1
11	MBIST z4a Instruction Cache Instance 2
12	MBIST z4a Instruction Cache Instance 3
13	MBIST z4a Instruction Cache Instance 4
14	MBIST z4a Instruction Cache Tag
15	MBIST z4a Data Cache Instance 1
16	MBIST z4a Data Cache Instance 2
17	MBIST z4a Data Cache Instance 3
18	MBIST z4a Data Cache Instance 4
19	MBIST z4a Data Cache Tag
20	MBIST DMA
21	MBIST Flexray Data RAM
22	MBIST Flexray LUT RAM Instance 1

Table 92. Corresponding error sources for system RAM OFLW0 register (continued)

OFLW0 bit-field position (bit)[1]	Error sources
23	MBIST Flexray LUT RAM Instance 2
24	MBIST Controller Area Network
25	MBIST Controller Area Network
26	MBIST SRAM (64KB)
27	MBIST MP Flash Krypton RAM
28	MBIST SRAM (32KB)
29	Reserved
30	Reserved
31	Reserved

1. Bit 0 in the table bit-field corresponds to bit 31 in target register.

Table 93. Corresponding error sources for peripheral RAM OFLW0 register

OFLW0 bit-field position (bit)[1]	Error sources
0	DMA TCD
1	DMA AHB
2	Flexray LUT RAM
3	Flexray Data RAM
4	CAN RAM - M_CAN0/1
5	CAN RAM - M_CAN2
6	Reserved
7	Reserved
8	Reserved
9	Reserved
10	Reserved
11	Reserved
12	Reserved
13	Reserved
14	Reserved
15	Reserved
16	Reserved
17	Reserved
18	Reserved
19	Reserved
20	Reserved

Table 93. Corresponding error sources for peripheral RAM OFLW0 register (continued)

OFLW0 bit-field position (bit)[1]	Error sources
21	Reserved
22	Reserved
23	Reserved
24	Reserved
25	Reserved
26	Reserved
27	Reserved
28	Reserved
29	Reserved
30	Reserved
31	Reserved

1. Bit 0 in the table bit-field corresponds to bit 31 in target register.

Table 94. Corresponding error sources for flash memory OFLW0 register

OFLW0 bit-field position (bit)[1]	Error sources
0	Flash 0
1	Flash 1
2	Reserved
3	Reserved
4	Reserved
5	Reserved
6	Reserved
7	Reserved
8	Reserved
9	Reserved
10	Reserved
11	Reserved
12	Reserved
13	Reserved
14	Reserved
15	Reserved
16	Reserved
17	Reserved
18	Reserved

Table 94. Corresponding error sources for flash memory OFLW0 register (continued)

OFLW0 bit-field position (bit)[1]	Error sources
19	Reserved
20	Reserved
21	Reserved
22	Reserved
23	Reserved
24	Reserved
25	Reserved
26	Reserved
27	Reserved
28	Reserved
29	Reserved
30	Reserved
31	Reserved

1. Bit 0 in the table bit-field corresponds to bit 31 in target register.

7.11.2.5 MEMU registers

Table 95 shows the address offsets and implementation of the MEMU registers in this chip.

Table 95. MEMU registers

Offset	Register
0x0000	Control register (CTRL)
0x0004	Error flag register (ERR_FLAG)
0x000C	Debug register (DEBUG)
0x0020	System RAM correctable error reporting table status register (SYS_RAM_CERR_STS0)
0x0024	System RAM correctable error reporting table address register (SYS_RAM_CERR_ADDR0)
0x0028	System RAM correctable error reporting table status register (SYS_RAM_CERR_STS1)
0x002C	System RAM correctable error reporting table address register (SYS_RAM_CERR_ADDR1)
0x0030	System RAM correctable error reporting table status register (SYS_RAM_CERR_STS2)
0x0034	System RAM correctable error reporting table address register (SYS_RAM_CERR_ADDR2)
0x0038	System RAM correctable error reporting table status register (SYS_RAM_CERR_STS3)
0x003C	System RAM correctable error reporting table address register (SYS_RAM_CERR_ADDR3)
0x0040	System RAM correctable error reporting table status register (SYS_RAM_CERR_STS4)

Table 95. MEMU registers (continued)

Offset	Register
0x0044	System RAM correctable error reporting table address register (SYS_RAM_CERR_ADDR4)
0x0048	System RAM correctable error reporting table status register (SYS_RAM_CERR_STS5)
0x004C	System RAM correctable error reporting table address register (SYS_RAM_CERR_ADDR5)
0x0050	System RAM correctable error reporting table status register (SYS_RAM_CERR_STS6)
0x0054	System RAM correctable error reporting table address register (SYS_RAM_CERR_ADDR6)
0x0058	System RAM correctable error reporting table status register (SYS_RAM_CERR_STS7)
0x005C	System RAM correctable error reporting table address register (SYS_RAM_CERR_ADDR7)
0x0060	System RAM correctable error reporting table status register (SYS_RAM_CERR_STS8)
0x0064	System RAM correctable error reporting table address register (SYS_RAM_CERR_ADDR8)
0x0068	System RAM correctable error reporting table status register (SYS_RAM_CERR_STS9)
0x006C	System RAM correctable error reporting table address register (SYS_RAM_CERR_ADDR9)
0x0070	System RAM uncorrectable error reporting table status register (SYS_RAM_UNCERR_STS)
0x0074	System RAM uncorrectable error reporting table address register (SYS_RAM_UNCERR_ADDR)
0x0078	System RAM concurrent overflow register (SYS_RAM_OFLW0)
0x0620	Peripheral RAM correctable error reporting table status register (PERIPH_RAM_CERR_STS0)
0x0624	Peripheral RAM correctable error reporting table address register (PERIPH_RAM_CERR_ADDR0)
0x0628	Peripheral RAM correctable error reporting table status register (PERIPH_RAM_CERR_STS1)
0x062C	Peripheral RAM correctable error reporting table address register (PERIPH_RAM_CERR_ADDR1)
0x0630	Peripheral RAM uncorrectable error reporting table status register (PERIPH_RAM_UNCERR_STS)
0x0634	Peripheral RAM uncorrectable error reporting table address register (PERIPH_RAM_UNCERR_ADDR)
0x0638	Peripheral RAM concurrent overflow register (PERIPH_RAM_OFLW0)
0x0C20	Flash memory correctable error reporting table status register (FLASH_CERR_STS0)
0x0C24	Flash memory correctable error reporting table address register (FLASH_CERR_ADDR0)
0x0C28	Flash memory correctable error reporting table status register (FLASH_CERR_STS1)

Table 95. MEMU registers (continued)

Offset	Register
0x0C2C	Flash memory correctable error reporting table address register (FLASH_CERR_ADDR1)
0x0C30	Flash memory correctable error reporting table status register (FLASH_CERR_STS2)
0x0C34	Flash memory correctable error reporting table address register (FLASH_CERR_ADDR2)
0x0C38	Flash memory correctable error reporting table status register (FLASH_CERR_STS3)
0x0C3C	Flash memory correctable error reporting table address register (FLASH_CERR_ADDR3)
0x0C40	Flash memory correctable error reporting table status register (FLASH_CERR_STS4)
0x0C44	Flash memory correctable error reporting table address register (FLASH_CERR_ADDR4)
0x0C48	Flash memory correctable error reporting table status register (FLASH_CERR_STS5)
0x0C4C	Flash memory correctable error reporting table address register (FLASH_CERR_ADDR5)
0x0C50	Flash memory correctable error reporting table status register (FLASH_CERR_STS6)
0x0C54	Flash memory correctable error reporting table address register (FLASH_CERR_ADDR6)
0x0C58	Flash memory correctable error reporting table status register (FLASH_CERR_STS7)
0x0C5C	Flash memory correctable error reporting table address register (FLASH_CERR_ADDR7)
0x0C60	Flash memory correctable error reporting table status register (FLASH_CERR_STS8)
0x0C64	Flash memory correctable error reporting table address register (FLASH_CERR_ADDR8)
0x0C68	Flash memory correctable error reporting table status register (FLASH_CERR_STS9)
0x0C6C	Flash memory correctable error reporting table address register (FLASH_CERR_ADDR9)
0x0C70	Flash memory correctable error reporting table status register (FLASH_CERR_STS10)
0x0C74	Flash memory correctable error reporting table address register (FLASH_CERR_ADDR10)
0x0C78	Flash memory correctable error reporting table status register (FLASH_CERR_STS11)
0x0C7C	Flash memory correctable error reporting table address register (FLASH_CERR_ADDR11)
0x0C80	Flash memory correctable error reporting table status register (FLASH_CERR_STS12)
0x0C84	Flash memory correctable error reporting table address register (FLASH_CERR_ADDR12)
0x0C88	Flash memory correctable error reporting table status register (FLASH_CERR_STS13)
0x0C8C	Flash memory correctable error reporting table address register (FLASH_CERR_ADDR13)
0x0C90	Flash memory correctable error reporting table status register (FLASH_CERR_STS14)

Table 95. MEMU registers (continued)

Offset	Register
0x0C94	Flash memory correctable error reporting table address register (FLASH_CERR_ADDR14)
0x0C98	Flash memory correctable error reporting table status register (FLASH_CERR_STS15)
0x0C9C	Flash memory correctable error reporting table address register (FLASH_CERR_ADDR15)
0x0CA0	Flash memory correctable error reporting table status register (FLASH_CERR_STS16)
0x0CA4	Flash memory correctable error reporting table address register (FLASH_CERR_ADDR16)
0x0CA8	Flash memory correctable error reporting table status register (FLASH_CERR_STS17)
0x0CAC	Flash memory correctable error reporting table address register (FLASH_CERR_ADDR17)
0x0CB0	Flash memory correctable error reporting table status register (FLASH_CERR_STS18)
0x0CB4	Flash memory correctable error reporting table address register (FLASH_CERR_ADDR18)
0x0CB8	Flash memory correctable error reporting table status register (FLASH_CERR_STS19)
0x0CBC	Flash memory correctable error reporting table address register (FLASH_CERR_ADDR19)
0x0CC0	Flash memory uncorrectable error reporting table status register (FLASH_UNCERR_STS)
0x0CC4	Flash memory uncorrectable error reporting table address register (FLASH_UNCERR_ADDR)
0x0CC8	Flash memory concurrent overflow register (FLASH_OFLW0)

7.11.3 FCCU configuration

7.11.3.1 Available FCCU output signals

Table 96 shows the available FCCU output signals implemented on this chip.

Table 96. FCCU output signals

FCCU output signal name	Chip-top signal name
EOUT[0](1), FI[0], ERROR0	FCCU_F0
EOUT[1](1), FI[1], ERROR1	FCCU_F1

1. FCCU_EOUT[0] and FCCU_EOUT[1] are both output and input pins.

7.11.3.2 FCCU register reset values

Table 97 shows the reset values of selected FCCU registers.

Table 97. Reset values of selected FCCU registers

Register	Reset value
FCCU_RF_CFG0	0xFFFF_FFFF
FCCU_RF_CFG1	0xFFFF_FFFF
FCCU_RF_CFG2	0x0000_0FFF
FCCU_RF_CFG3	0x0000_0000
FCCU_RFS_CFG0	0x0200_2000
FCCU_RFS_CFG1	0x0000_0000
FCCU_RFS_CFG2	0x0000_0000
FCCU_RFS_CFG3	0x0000_0000
FCCU_RFS_CFG4	0x0000_0000
FCCU_RFS_CFG5	0x0000_0000
FCCU_RFS_CFG6	0x0000_0000
FCCU_RFS_CFG7	0x0000_0000
FCCU_RF_S0	0x0000_0000
FCCU_RF_S1	0x0000_0000
FCCU_RF_S2	0x0000_0000
FCCU_RF_S3	0x0000_0000
FCCU_RF_E0	0x0000_1040
FCCU_RF_E1	0x0000_0000
FCCU_RF_E2	0x0000_0000
FCCU_RF_E3	0x0000_0000
FCCU_RF_TOE0	0x0000_0000
FCCU_RF_TOE1	0x0000_0000
FCCU_RF_TOE2	0x0000_0000
FCCU_RF_TOE3	0x0000_0000
FCCU_EOUT_SIG_EN0	0x0000_0000
FCCU_EOUT_SIG_EN1	0x0000_0000
FCCU_EOUT_SIG_EN2	0x0000_0000
FCCU_EOUT_SIG_EN3	0x0000_0000

7.11.3.3 FOSU parameters

The FOSU_COUNT, or FOSU timeout response time, is 1000 safe clock cycles.

7.11.4 Register protection (REG_PROT) configuration

Some modules on this device include a built-in register protection mechanism that can be used to override the normal write permissions associated with individual registers in the

module. This mechanism is software controlled and can be dynamically reconfigured at runtime or made effective until device reset. For a detailed explanation, see the *Chapter 67: Register Protection (REG_PROT)*.

Table 98. Reference links to related information

Topic	Related module	Reference
System memory map	—	*Chapter 5: Memory map*
CMU registers	CMU	*Chapter 26: Clock Monitor Unit (CMU)*
JTAG Master (JTAGM) registers	JTAGM	*Chapter 58: JTAG Master (JTAGM)*
Clock Generation Module (ME_CGM) registers	MC_CGM	*Chapter 27: Clock Generation Module (MC_CGM)*
Mode Entry (MC_ME) registers	MC_ME	*Chapter 52: Mode Entry Module (MC_ME)*
Reset Generation Module (MC_RGM) registers	MC_RGM	*Chapter 47: Reset Generation Module (MC_RGM)*
Memory Error Management Unit (MEMU) registers	MEMU	*Chapter 64: Memory Error Management Unit (MEMU)*
Oscillator Digital Interface (OSC_DIG) registers	OSC_DIG	*Chapter 28: OSC Digital Interface (XOSC)*
PLL Digital Interface (PLL_DIG) registers	PLL_DIG	*Chapter 25: Dual PLL Digital Interface (PLLDIG)*
PMC Digital Interface (PMC_DIG) registers	PMC_DIG	*Chapter 50: Power management controller digital interface (PMC_Dig)*
SIUL2 registers	SIUL2	*Chapter 14: System Integration Unit Lite2 (SIUL2)*
System Status and Control Module (SSCM) registers	SSCM	*Chapter 49: System Status and Configuration Module (SSCM)*

7.11.4.1 SIUL2 protected registers

Table 99 lists the SIUL2 registers that can be protected.

Table 99. Protected SIUL2 registers

Register	Register size (bits)	Offset from module base address	Protected size (bits)
SIUL2_DIRER0	32	0x0018	32
SIUL2_DIRSR0	32	0x0020	32
SIUL2_IREER0	32	0x0028	32
SIUL2_IFEER0	32	0x0030	32
MSCR 0	32	0x0240	32
MSCR 1–131	32	—(1)	32
MSCR 480–485	32	—(1)	32
MSCR 500–503	32	—(1)	32
MSCR 512–513	32	—(1)	32
MSCR 520	32	0X0A60	32
MSCR 560–561	32	—(1)	32
MSCR 624–635	32	—(1)	32
MSCR 688–719	32	—(1)	32
MSCR 752–786	32	—(1)	32
MSCR 788–790	32	—(1)	32
MSCR 792–794	32	—(1)	32
MSCR 816–821	32	—(1)	32
MSCR 944	32	0x1100	32
MSCR 948	32	0X1110	32
MSCR 952–999	32	—(1)	32
SIUL2_GPDO0–131	8	—(1)	8
SIUL2_PGPDO0–8	16	—(1)	16
SIUL2_MPGPDO0–8	32	—(1)	32

1. See the module memory map in *Chapter 14: System Integration Unit Lite2 (SIUL2)* for address offset.

7.11.4.2 Clock Monitor Unit (CMU) protected registers

Certain CMU registers can be write-protected. A soft lock can be unlocked by software after being previously locked. A hard lock is only unlocked by a reset once locked. *Table 100* lists the CMU registers that can be protected.

Table 100. Protected CMU registers

Module	Register(1)	Register size (bits)	Offset from module base address	Protected size (bits)
CMU_0	CMU_CSR	8	0x0203	8
	CMU_HFREFR	32	0x0208	32
	CMU_LFREFR	32	0x020C	32
	CMU_MDR	32	0x0218	32
CMU_1	CMU_CSR	8	0x0243	8
	CMU_HFREFR	32	0x0248	32
	CMU_LFREFR	32	0x024C	32
	CMU_MDR	32	0x0258	32
CMU_2	CMU_CSR	8	0x0283	8
	CMU_HFREFR	32	0x0288	32
	CMU_LFREFR	32	0x028C	32
	CMU_MDR	32	0x0298	32
CMU_3	CMU_CSR	8	0x02C3	8
	CMU_HFREFR	32	0x02C8	32
	CMU_LFREFR	32	0x02CC	32
	CMU_MDR	32	0x02D8	32

1. See *Chapter 67: Register Protection (REG_PROT)* for bit field details.

7.11.4.3 PMC Digital Interface (PMC_DIG) protected registers

Table 101 lists the PMC_DIG registers that can be protected.

Table 101. Protected PMC_DIG registers

Register	Register size (bits)	Offset from module base address	Protected size (bits)
REE_LV0	32	0x0404	32
RES_LV0	32	0x0408	32
FEE_LV0	32	0x0410	32
REE_LV1	32	0x0424	32
RES_LV1	32	0x0428	32
FEE_LV1	32	0x0430	32
REE_HV0	32	0x0444	32
RES_HV0	32	0x0448	32
FEE_HV0	32	0x0450	32
REE_HV1	32	0x0464	32
RES_HV1	32	0x0468	32

Table 101. Protected PMC_DIG registers (continued)

Register	Register size (bits)	Offset from module base address	Protected size (bits)
FEE_HV1	32	0x0470	32
VSIO	32	0x0504	32
MREG_CTRL	8	0X0643	8
REE_TD	32	0x0704	32
RES_TD	32	0x0708	32
CTL_TD	32	0x070C	32
FEE_TD	32	0x0718	32

7.11.4.4 PLL Digital Interface (PLL_DIG) protected registers

Certain PLL registers can be write-protected. The protected registers have write protection with both soft and hard locking. A soft lock can be unlocked by software after being previously locked. A hard lock is only unlocked by a reset once locked. *Table 102* lists the PLL_DIG registers that can be write-protected.

Table 102. Protected PLL_DIG registers

Register(1)	Register size (bits)	Offset from module base address	Protected size (bits)
PLL0_CR	32	0x0100	32
PLL0_DIVR	32	0x0108	32
PLL1_CR	32	0x0120	32
PLL1_DIVR	32	0x0128	32
PLL1_MR	32	0x012C	32
PLL1_CSR	32	0x0130	32

1. See *Chapter 67: Register Protection (REG_PROT)* for bit field details.

7.11.4.5 Oscillator Digital Interface (XOSC) protected registers

Certain XOSC registers can be write-protected.. The protected registers have write protection with both soft and hard locking. A soft lock can be unlocked by software after being previously locked. A hard lock is only unlocked by a reset once locked. *Table 103* lists the Oscillator Digital Interface registers that can be write-protected.

Table 103. Protected XOSC registers

Register(1)	Register size (bits)	Offset from module base address	Protected size (bits)
OSC_CTL	32	0x0080	32

1. See *Chapter 67: Register Protection (REG_PROT)* for bit field details.

7.11.4.6 Memory Error Management Unit (MEMU) protected registers

Table 104 lists the MEMU registers that can be protected.

Table 104. Protected MEMU registers

Register	Register size (bits)	Offset from module base address	Protected size (bits)
CTRL	8	0x000	8
DEBUG	32	0x00C	32

7.11.4.7 Mode Entry (MC_ME) protected registers

Table 105 lists the MC_ME registers that can be protected.

Table 105. Protected MC_ME registers

Register	Register size (bits)	Offset from module base address	Protected size (bits)
ME_MCTL	32	0x004	32
ME_ME	32	0x008	32
ME_IM	32	0x010	32
ME_SAFE_MC	32	0x028	32
ME_DRUN_MC	32	0x02C	32
ME_RUN0_MC	32	0x030	32
ME_RUN1_MC	32	0x034	32
ME_RUN2_MC	32	0x038	32
ME_RUN3_MC	32	0x03C	32
ME_HALT0_MC	32	0x040	32
ME_STOP0_MC	32	0x048	32
ME_RUN_PC0	32	0x080	32
ME_RUN_PC1	32	0x084	32
ME_RUN_PC2	32	0x088	32
ME_RUN_PC3	32	0x08C	32
ME_RUN_PC4	32	0x090	32
ME_RUN_PC5	32	0x094	32
ME_RUN_PC6	32	0x098	32
ME_RUN_PC7	32	0x09C	32
ME_LP_PC0	32	0x0A0	32
ME_LP_PC1	32	0x0A4	32
ME_LP_PC2	32	0x0A8	32
ME_LP_PC3	32	0x0AC	32

Table 105. Protected MC_ME registers (continued)

Register	Register size (bits)	Offset from module base address	Protected size (bits)
ME_LP_PC4	32	0x0B0	32
ME_LP_PC5	32	0x0B4	32
ME_LP_PC6	32	0x0B8	32
ME_LP_PC7	32	0x0BC	32
ME_PCTL15	8	0x0CF	8
ME_PCTL30	8	0x0DE	8
ME_PCTL36	8	0x0E4	8
ME_PCTL38	8	0x0E6	8
ME_PCTL62	8	0x0FE	8
ME_PCTL69	8	0x105	8
ME_PCTL70	8	0x106	8
ME_PCTL74	8	0x10A	8
ME_PCTL91	8	0x11B	8
ME_PCTL92	8	0x11C	8
ME_PCTL98	8	0x122	8
ME_PCTL99	8	0x123	8
ME_PCTL104	8	0x128	8
ME_PCTL107	8	0x12B	8
ME_PCTL111	8	0x12F	8
ME_PCTL112	8	0x130	8
ME_PCTL121	8	0x139	8
ME_PCTL122	8	0x13A	8
ME_PCTL123	8	0x13B	8
ME_PCTL125	8	0x13D	8
ME_PCTL126	8	0x13E	8
ME_PCTL127	8	0x13F	8
ME_PCTL128	8	0x140	8
ME_PCTL129	8	0x141	8
ME_PCTL130	8	0x142	8
ME_PCTL131	8	0x143	8
ME_PCTL136	8	0x148	8
ME_PCTL138	8	0x14A	8
ME_PCTL139	8	0x14B	8
ME_PCTL140	8	0x14C	8

Table 105. Protected MC_ME registers (continued)

Register	Register size (bits)	Offset from module base address	Protected size (bits)
ME_PCTL141	8	0x14D	8
ME_PCTL143	8	0x14F	8
ME_PCTL190	8	0x17E	8
ME_PCTL198	8	0x186	8
ME_PCTL202	8	0x18A	8
ME_PCTL219	8	0x19B	8
ME_PCTL220	8	0x19C	8
ME_PCTL226	8	0x1A2	8
ME_PCTL227	8	0x1A3	8
ME_PCTL232	8	0x1A8	8
ME_PCTL239	8	0x1AF	8
ME_PCTL240	8	0x1B0	8
ME_PCTL244	8	0x1B4	8
ME_PCTL249	8	0x1B9	8
ME_PCTL250	8	0x1BA	8
ME_PCTL251	8	0x1BB	8
ME_PCTL253	8	0x1BD	8
ME_PCTL254	8	0x1BE	8
ME_PCTL255	8	0x1BF	8
ME_CADDR0	32	0x1E0	32

7.11.4.8 Clock Generation Module (MC_CGM) protected registers

Table 106 lists the MC_CGM registers that can be protected.

Table 106. Protected MC_CGM registers

Register	Register size (bits)	Offset from module base address	Protected size (bits)
CMU_MDR	32	0x0218	32
CGM_SC_DIV_RC	32	0x07D0	32
CGM_DIV_UPD_TYPE	32	0x07D4	32
CGM_DIV_UPD_TRIG	32	0x07D8	32
CGM_SC_DC0	32	0x07E8	32
CGM_SC_DC1	32	0x07EC	32
CGM_SC_DC2	32	0x07F0	32
CGM_SC_DC3	32	0x07F4	32

Table 106. Protected MC_CGM registers (continued)

Register	Register size (bits)	Offset from module base address	Protected size (bits)
CGM_AC0_SC	8	0X0800	8
CGM_AC0_DC0	32	0X808	32
CGM_AC0_DC1	32	0X80C	32
CGM_AC0_DC2	32	0X810	32
CGM_AC1_SC	8	0X820	8
CGM_AC1_DC0	32	0X828	32
CGM_AC2_SC	8	0X840	8
CGM_AC3_SC	8	0X860	8
CGM_AC4_SC	8	0X880	8
CGM_AC4_DC0	8	0X888	8
CGM_AC4_DC1	8	0X88C	8
CGM_AC4_DC2	8	0X890	8
CGM_AC4_DC3	8	0X894	8
CGM_AC4_DC4	8	0X898	8
CGM_AC4_DC5	8	0X89C	8
CGM_AC5_SC	8	0X8A0	8
CGM_AC5_DC0	8	0X8A8	8

7.11.4.9 Reset Generation Module (MC_RGM) protected registers

Table 107 lists the MC_RGM registers that can be protected.

Table 107. Protected MC_RGM registers

Register	Register size (bits)	Offset from module base address	Protected size (bits)
RGM_DERD	32	0x010	32
RGM_DEAR	32	0x020	32
RGM_FERD	32	0x310	32
RGM_FEAR	32	0x320	32
RGM_FESS	32	0x340	32
RGM_PRST0	32	0x610	32
RGM_PRST1	32	0x614	32
RGM_PRST2	32	0x618	32
RGM_PRST3	32	0x61C	32
RGM_PRST4	32	0x620	32
RGM_PRST5	32	0x624	32

Table 107. Protected MC_RGM registers (continued)

Register	Register size (bits)	Offset from module base address	Protected size (bits)
RGM_PRST6	32	0x628	32
RGM_PRST7	32	0x62C	32

7.11.4.10 System Status and Control Module (SSCM) protected registers

Table 108 lists the SSCM registers that can be protected.

Table 108. Protected SSCM registers

Register	Register size (bits)	Offset from module base address	Protected size (bits)
ERROR_CONFIG	16	0x006	16

7.11.4.11 JTAGM (JTAGM) protected registers

Table 109 lists the JTAGM registers that can be protected.

Table 109. Protected JTAGM registers

Register	Register size (bits)	Offset from module base address	Protected size (bits)
JTAGM_MCR	32	0x000	32

7.11.5 Password and Device Security Module (PASS) configuration

The PASS module is used to implement password-based read and write protection for flash blocks on the SPC574Sx MCU. Up to four levels of password protection can be implemented for each flash block.

Note: The registers discussed in this section are initialized via DCF records, so the reset values depend on user settings.

See the Password and Device Security Module (PASS) chapter for further details.

7.11.5.1 PASS_LOCK*n*_PG*n* register bit mapping

Each password group defined in the DCF records has a set of four LOCKn registers association with it—PASS_LOCK*n*_PG*n*. The bits of these registers are associated with specific flash blocks. The mapping is shown in the following figures.

Note: The mapping of various LOCK register bits to flash blocks in the SPC574Sx microcontroller is very similar, but not identical. The PASS module has additional mapping in the LOCK3 register and defines read locking regions. See the PASS chapter for details.

Bit	0	1	2	3	27	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Register field	TSLOCK	Reserved	LOWLOCK[13:0]														MIDLOCK[15:0]															
Flash block name	UTESTNVM Block Space 16 KB	—	—	—	—	—	—	64 KB Code Flash block1	64 KB Code Flash block0	32 KB Code Flash block1	32 KB Code Flash block0	16 KB Code Flash block4	16 KB Code Flash block3	16 KB Code Flash block2	16 KB Code Flash block1	16 KB BAF block0	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
Read lock region	Read Lock Region 0	—	—	—	—	—	—	Read Lock Region 1									—	—	—	—	—	—	—	—	—	—	—	—	—	—	—

Figure 38. PASS_LOCK0_PGn Register

Bit	0	1	2	3	27	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Register field	Reserved																HIGHLOCK[15:0]															

Figure 39. PASS_LOCK1_PGn Register

Bit	0	1	2	3	27	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Flash block name	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	16KB MP EEPROM Block3	16KB MP EEPROM Block2	16KB MP EEPROM Block1	16KB MP EEPROM Block0
Read lock region	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	Read Lock Region 2			

Figure 39. PASS_LOCK1_PGn Register (continued)

Bit	0	1	2	3	27	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Register field	A256KLOCK[31:0]																															
Flash block name	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	256KB Code Flash block4	256KB Code Flash block3	256KB Code Flash block2	256KB Code Flash block1	256KB Code Flash block0
Read lock region	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	Read Lock Region 1				

Figure 40. PASS_LOCK2_PGn Register

Bit	0	1	2	3	27	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Register field	Reserved																															
Flash block name	PGL	DBL	—	—	—	—	—	—	—	—	—	—	—	RL2	RL1	RL0	A256KLOCK[47:32]															

Figure 41. PASS_LOCK3_PGn Register

7.11.5.2 PASS Module DCF Records (includes lifecycle state and censorship)

The records required to configure the initial values for the PASS module password group registers are shown in *Table 110*. These values provide the following capabilities:

- Set flash memory to censored/uncensored state (censorship also depends on lifecycle)
- Override the basic flash write protection for flash memory blocks
- Set levels of password-based write protection (up to four 256-bit passwords can be required) for individual blocks of flash memory
- Set levels of password-based read protection (up to four 256-bit passwords can be required) for up to five flash memory Read Locking Regions
- Set levels of password-based protection (up to four 256-bit passwords can be required) for the debug part.

Table 110. PASS DCF records

DCF CS[14:0]	DCF address (binary)		DCF client description	Number of	
	[16:10]	[9:2]		Valid bits	DCF records
000_0000_0000_1000	0000000	00101000–00101011	Reserved		
000_0000_0000_1000	0000000	00101100	Censorship(1)	16	1
000_0000_0000_1000	0000000	00101101–00111111	Reserved		
000_0000_0000_1000	0000000	01000000	LOCK0_PG0(2)	32	1
000_0000_0000_1000	0000000	01000001	LOCK1_PG0(2)	32	1
000_0000_0000_1000	0000000	01000010	LOCK2_PG0(2)	32	1
000_0000_0000_1000	0000000	01000011	LOCK3_PG0(2)	32	1

Table 110. PASS DCF records (continued)

DCF CS[14:0]	DCF address (binary)		DCF client description	Number of	
	[16:10]	[9:2]		Valid bits	DCF records
000_0000_0000_1000	0000000	01000100	LOCK0_PG1(2)	32	1
000_0000_0000_1000	0000000	01000101	LOCK1_PG1(2)	32	1
000_0000_0000_1000	0000000	01000110	LOCK2_PG1(2)	32	1
000_0000_0000_1000	0000000	01000111	LOCK3_PG1(2)	32	1
000_0000_0000_1000	0000000	01001000	LOCK0_PG2(2)	32	1
000_0000_0000_1000	0000000	01001001	LOCK1_PG2(2)	32	1
000_0000_0000_1000	0000000	01001010	LOCK2_PG2(2)	32	1
000_0000_0000_1000	0000000	01001011	LOCK3_PG2(2)	32	1
000_0000_0000_1000	0000000	01001100	LOCK0_PG3(2)	32	1
000_0000_0000_1000	0000000	01001101	LOCK1_PG3(2)	32	1
000_0000_0000_1000	0000000	01001110	LOCK2_PG3(2)	32	1
000_0000_0000_1000	0000000	01001111	LOCK3_PG3(2)	32	1

1. See <Cross Refs>Section 1.6.4, Censoring/uncensoring the device for values.
2. See *Section 7.11.5.1: PASS_LOCKn_PGn register bit mapping* for values.

7.11.6 STCU2 configuration

7.11.6.1 BIST Configuration

Table 111. MBIST configuration 1 - Offline

DCF Value	STCU2 Register Name	Comment
D3FEA98B 00080008	STCU_KEY1	Key1 in offline mode
2C015674 00080008	STCU_KEY2	Key2 in offline mode
10000001 0008000C	STCU_CFG	Pointer set for Mbist0, Mbist reduced algo(PMOSEN bit set to '0'), PMOSEN and MBU can be played with to chose different algo and mbist coverage, WRP bit set to '0' to allow IPS access. Refer to STCU doc for more details.

Table 111. MBIST configuration 1 - Offline (continued)

DCF Value	STCU2 Register Name	Comment
0C02003C 00080010	STCU_PLL_CFG	PLL programmed for 40 Mhz with input frequency as RC(16 Mhz), PLL output freq = (Input_freq*LDF)/(ODF*IDF), = (Input_freq* 'h3C)/('h0C*02), play with ODF factor(04) to reduce frequency.
FFFFFFF0 00080014	Watchdog	
00000000 00080074	STCU_MBUFML	Programming '1' to respective mbist in this register generates critical fault in case of mbist failure. If the fault is permanent, then the device will never recover due to this programming. So safe to program '0' in this register and later stcu status can be read by software.
00000000 00080028	STCU_ERR_FM	Again programming '1' to any corresponding of this register can make device unrecoverable if fault is permanent.
91000000 00080600	MBIST_CTRL0	Next pointer set to Mbist1, CSM bist set to '1' to indicate parallel running of MBIST0 and MBIST1
92000000 00080604	MBIST_CTRL1	Next pointer set to Mbist2, CSM bist set to '1' to indicate parallel running of MBIST1 and MBIST2
93000000 00080608	MBIST_CTRL2	Next pointer set to Mbist3, CSM bist set to '1' to indicate parallel running of MBIST2 and MBIST3
94000000 0008060C	MBIST_CTRL3	Next pointer set to Mbist4, CSM bist set to '1' to indicate parallel running of MBIST3 and MBIST4
95000000 00080610	MBIST_CTRL4	Next pointer set to Mbist5, CSM bist set to '1' to indicate parallel running of MBIST4 and MBIST5
96000000 00080614	MBIST_CTRL5	Next pointer set to Mbist6, CSM bist set to '1' to indicate parallel running of MBIST5 and MBIST6
97000000 00080618	MBIST_CTRL6	Next pointer set to Mbist7, CSM bist set to '1' to indicate parallel running of MBIST6 and MBIST7
98000000 0008061C	MBIST_CTRL7	Next pointer set to Mbist8, CSM bist set to '1' to indicate parallel running of MBIST7 and MBIST8
99000000 00080620	MBIST_CTRL8	Next pointer set to Mbist9, CSM bist set to '1' to indicate parallel running of MBIST8 and MBIST9
9A000000 00080624	MBIST_CTRL9	Next pointer set to Mbist10, CSM bist set to '1' to indicate parallel running of MBIST9 and MBIST10
9B000000 00080628	MBIST_CTRL10	Next pointer set to Mbist11, CSM bist set to '1' to indicate parallel running of MBIST10 and MBIST11
9C000000 0008062C	MBIST_CTRL11	Next pointer set to Mbist12, CSM bist set to '1' to indicate parallel running of MBIST11 and MBIST12
9D000000 00080630	MBIST_CTRL12	Next pointer set to Mbist13, CSM bist set to '1' to indicate parallel running of MBIST12 and MBIST13
9E000000 00080634	MBIST_CTRL13	Next pointer set to Mbist14, CSM bist set to '1' to indicate parallel running of MBIST13 and MBIST14
9F000000 00080638	MBIST_CTRL14	Next pointer set to Mbist15, CSM bist set to '1' to indicate parallel running of MBIST14 and MBIST15

Table 111. MBIST configuration 1 - Offline (continued)

DCF Value	STCU2 Register Name	Comment
A0000000 0008063C	MBIST_CTRL15	Next pointer set to Mbist16, CSM bist set to '1' to indicate parallel running of MBIST15 and MBIST16
A1000000 00080640	MBIST_CTRL16	Next pointer set to Mbist17, CSM bist set to '1' to indicate parallel running of MBIST16 and MBIST17
A2000000 00080644	MBIST_CTRL17	Next pointer set to Mbist18, CSM bist set to '1' to indicate parallel running of MBIST17 and MBIST18
A3000000 00080648	MBIST_CTRL18	Next pointer set to Mbist19, CSM bist set to '1' to indicate parallel running of MBIST18 and MBIST19
A4000000 0008064C	MBIST_CTRL19	Next pointer set to Mbist20, CSM bist set to '1' to indicate parallel running of MBIST19 and MBIST20
7F000000 0008064F	MBIST_CTRL20	Next pointer set to null pointer to indicate end of selftest
00000201 00080000	STCU_RUN	Selftest triggered with MBIST run on PLL frequency

Table 112. MBIST configuration 1 - Online

Register Value	Register Name	Comment
753F924E	STCU_KEY1	Key1 in online mode
8AC06DB1	STCU_KEY2	Key2 in online mode
10000001	STCU_CFG	Pointer set for Mbist0, Mbist reduced algo(PMOSEN bit set to '0'), PMOSEN and MBU can be played with to chose different algo and mbist coverage, WRP bit set to '0' to allow IPS access. Refer to STCU doc for more details.
FFFFFFF0	Watchdog	
00000000	STCU_MBUFML	Programming '1' to respective mbist in this register generates critical fault in case of mbist failure. If the fault is permanent, then the device will never recover due to this programming. So safe to program '0' in this register and later stcu status can be read by software.
00000000	STCU_ERR_FM	Again programming '1' to any corresponding of this register can make device unrecoverable if fault is permanent.
91000000	MBIST_CTRL0	Next pointer set to Mbist1, CSM bist set to '1' to indicate parallel running of MBIST0 and MBIST1
92000000	MBIST_CTRL1	Next pointer set to Mbist2, CSM bist set to '1' to indicate parallel running of MBIST1 and MBIST2
93000000	MBIST_CTRL2	Next pointer set to Mbist3, CSM bist set to '1' to indicate parallel running of MBIST2 and MBIST3
94000000	MBIST_CTRL3	Next pointer set to Mbist4, CSM bist set to '1' to indicate parallel running of MBIST3 and MBIST4
95000000	MBIST_CTRL4	Next pointer set to Mbist5, CSM bist set to '1' to indicate parallel running of MBIST4 and MBIST5

Table 112. MBIST configuration 1 - Online (continued)

Register Value	Register Name	Comment
96000000	MBIST_CTRL5	Next pointer set to Mbist6, CSM bist set to '1' to indicate parallel running of MBIST5 and MBIST6
97000000	MBIST_CTRL6	Next pointer set to Mbist7, CSM bist set to '1' to indicate parallel running of MBIST6 and MBIST7
98000000	MBIST_CTRL7	Next pointer set to Mbist8, CSM bist set to '1' to indicate parallel running of MBIST7 and MBIST8
99000000	MBIST_CTRL8	Next pointer set to Mbist9, CSM bist set to '1' to indicate parallel running of MBIST8 and MBIST9
9A000000	MBIST_CTRL9	Next pointer set to Mbist10, CSM bist set to '1' to indicate parallel running of MBIST9 and MBIST10
9B000000	MBIST_CTRL10	Next pointer set to Mbist11, CSM bist set to '1' to indicate parallel running of MBIST10 and MBIST11
9C000000	MBIST_CTRL11	Next pointer set to Mbist12, CSM bist set to '1' to indicate parallel running of MBIST11 and MBIST12
9D000000	MBIST_CTRL12	Next pointer set to Mbist13, CSM bist set to '1' to indicate parallel running of MBIST12 and MBIST13
9E000000	MBIST_CTRL13	Next pointer set to Mbist14, CSM bist set to '1' to indicate parallel running of MBIST13 and MBIST14
9F000000	MBIST_CTRL14	Next pointer set to Mbist15, CSM bist set to '1' to indicate parallel running of MBIST14 and MBIST15
A0000000	MBIST_CTRL15	Next pointer set to Mbist16, CSM bist set to '1' to indicate parallel running of MBIST15 and MBIST16
A1000000	MBIST_CTRL16	Next pointer set to Mbist17, CSM bist set to '1' to indicate parallel running of MBIST16 and MBIST17
A2000000	MBIST_CTRL17	Next pointer set to Mbist18, CSM bist set to '1' to indicate parallel running of MBIST17 and MBIST18
A3000000	MBIST_CTRL18	Next pointer set to Mbist19, CSM bist set to '1' to indicate parallel running of MBIST18 and MBIST19
A4000000	MBIST_CTRL19	Next pointer set to Mbist20, CSM bist set to '1' to indicate parallel running of MBIST19 and MBIST20
7F000000	MBIST_CTRL20	Next pointer set to null pointer to indicate end of selftest
00000201	STCU_RUNSW	Selftest triggered with MBIST run on PLL frequency

Table 113. LBIST configuration 2 - Offline

DCF Value	STCU Register Name	Comment
D3FEA98B 00080008	STCU_KEY1	Key1 in offline mode
2C015674 00080008	STCU_KEY2	Key2 in offline mode
005A0001 0008000C	STCU_CFG	Pointer set for Lbist partition0, WRP bit set to ‘0’ to allow IPS access

Table 113. LBIST configuration 2 - Offline (continued)

DCF Value	STCU Register Name	Comment
10020046 00080010	PLL configuration	PLL programmed for 35Mhz with input frequency as RC(16 Mhz), PLL output freq = (Input_freq*LDF)/(ODF*IDF), = (Input_freq* 'h46)/('h10*02), Play with ODF factor('h10) to reduce freq
FFFFFFF0 00080014	Watchdog	
00000000 00080040	STCU_LBCFML	Programming '1' to respective lbist in this register generates critical fault in case of lbist failure. If the fault is permanent, then the device will never recover due to this programming. So safe to program '0' in this register and later stcu status can be read by software.
81001100 00080100	STCU_LB_CTRL_0	Next pointer to LBIST partition1, parallel mode, CSM bit(31st) can be played with to run in serial mode
00001388 00080104	STCU_LB_PCS_0	Number of patterns -5000 for partition0
E8BC505E 00080110	STCU_LB_MISREL_0	MISR lower 32 bits for 5000nd pattern for partition0. Will change according to the number of patterns to be executed.
D15C99CC 00080114	STCU_LB_MISREH_0	MISR upper 32 bits for 5000nd pattern for partition0. Will change according to the number of patterns to be executed.
82001100 00080140	STCU_LB_CTRL_1	Next pointer to LBIST partition2, parallel mode, CSM bit(31st) can be played with to run in serial mode
00001388 00080144	STCU_LB_PCS_1	Number of patterns -5000 for partition1
63039729 00080150	STCU_LB_MISREL_1	MISR lower 32 bits for 5000nd pattern for partition1. Will change according to the number of patterns to be executed.
F5D1D7E8 00080154	STCU_LB_MISREH_1	MISR upper 32 bits for 5000nd pattern for partition1. Will change according to the number of patterns to be executed.
83001100 00080180	STCU_LB_CTRL_2	Next pointer to LBIST partition3, parallel mode, CSM bit(31st) can be played with to run in serial mode
00001388 00080184	STCU_LB_PCS_2	Number of patterns -5000 for partition2
45DF8B4B 00080190	STCU_LB_MISREL_2	MISR lower 32 bits for 5000nd pattern for partition2. Will change according to the number of patterns to be executed.
966615B8 00080194	STCU_LB_MISREH_2	MISR upper 32 bits for 5000nd pattern for partition2. Will change according to the number of patterns to be executed.
7F001100 000801C0	STCU_LB_CTRL_3	Next pointer to null to indicate end of selftest
00001388 000801C4	STCU_LB_PCS_3	Number of patterns -5000 for partition3
B70EE894 000801D0	STCU_LB_MISREL_3	MISR lower 32 bits for 5000nd pattern for partition3. Will change according to the number of patterns to be executed.
B3E1B779 000801D4	STCU_LB_MISREH_3	MISR upper 32 bits for 5000nd pattern for partition3. Will change according to the number of patterns to be executed.
00000101 00080000	STCU_RUN	STCU triggered with LBIST programmed to run on PLL freq.

Table 114. LBIST configuration 2 - Online

Register Value	Register Name	Comment
753F924E	STCU_KEY1	Key1 in online mode
8AC06DB1	STCU_KEY2	Key2 in online mode
005A0001	STCU_CFG	Pointer set for Lbist partition0, WRP bit set to '0' to allow IPS access
FFFFFFF0	Watchdog	
00000000	STCU_LBUFML	Programming '1' to respective lbist in this register generates critical fault in case of lbist failure. If the fault is permanent, then the device will never recover due to this programming. So safe to program '0' in this register and later stcu status can be read by software.
81001100	STCU_LB_CTRL_0	Next pointer to LBIST partition1, parallel mode, CSM bit(31st) can be played with to run in serial mode
00001388	STCU_LB_PCS_0	Number of patterns -5000 for partition0
E8BC505E	STCU_LB_MISREL_0	MISR lower 32 bits for 5000nd pattern for partition0. Will change according to the number of patterns to be executed.
D15C99CC	STCU_LB_MISREH_0	MISR upper 32 bits for 5000nd pattern for partition0. Will change according to the number of patterns to be executed.
82001100	STCU_LB_CTRL_1	next pointer to LBIST partition2, parallel mode, CSM bit(31st) can be played with to run in serial mode
00001388	STCU_LB_PCS_1	number of patterns -5000 for partition1
63039729	STCU_LB_MISRELSW_1	MISR lower 32 bits for 5000nd pattern for partition1. Will change according to the number of patterns to be executed.
F5D1D7E8	STCU_LB_MISREHSW_1	MISR upper 32 bits for 5000nd pattern for partition1. Will change according to the number of patterns to be executed.
83001100	STCU_LB_CTRL_2	Next pointer to LBIST partition3, parallel mode, CSM bit(31st) can be played with to run in serial mode
00001388	STCU_LB_PCS_2	Number of patterns -5000 for partition2
45DF8B4B	STCU_LB_MISRELSW_2	MISR lower 32 bits for 5000nd pattern for partition2. Will change according to the number of patterns to be executed.
966615B8	STCU_LB_MISREHSW_2	MISR upper 32 bits for 5000nd pattern for partition2. Will change according to the number of patterns to be executed.
7F001100	STCU_LB_CTRL_3	Next pointer to null to indicate end of selftest
00001388	STCU_LB_PCS_3	Number of patterns -5000 for partition3
B70EE894	STCU_LB_MISRELSW_3	MISR lower 32 bits for 5000nd pattern for partition3. Will change according to the number of patterns to be executed.

Table 114. LBIST configuration 2 - Online (continued)

Register Value	Register Name	Comment
B3E1B779	STCU_LB_MISREHSW_3	MISR upper 32 bits for 5000nd pattern for partition3. Will change according to the number of patterns to be executed.
00001010	STCU_RUNSW	STCU triggered with LBIST programmed to run on PLL freq.

Table 115. Full Selftest configuration 3 - Offline

DCF Value	Register Name	Comment
D3FEA98B 00080008	STCU_KEY1	Key1 in offline mode
2C015674 00080008	STCU_KEY2	Key2 in offline mode
005A0001 0008000C	STCU_CFG	Pointer set for Lbist partition0, Mbist reduced algo(PMOSEN bit set to '0'), PMOSEN and MBU can be played with to chose different algo and mbist coverage, WRP bit set to '0' to allow IPS access
0C02003C 00080010	STCU_PLL_CFG	PLL programmed for 40 Mhz with input frequency as RC(16 Mhz), PLL output freq = (Input_freq*LDF)/(ODF*IDF), = (Input_freq* 'h3C)/('h0C*02), play with ODF factor(04) to reduce frequency.
FFFFFFF0 00080014	Watchdog	
81001100 00080100	STCU_LB_CTRL_0	Next pointer to LBIST partition1, parallel mode, CSM bit(31st) can be played with to run in serial mode
00001388 00080104	STCU_LB_PCS_0	Number of patterns -5000 for partition0
E8BC505E 00080110	STCU_LB_MISREL_0	MISR lower 32 bits for 5000nd pattern for partition0. Will change according to the number of patterns to be executed.
D15C99CC 00080114	STCU_LB_MISREH_0	MISR upper 32 bits for 5000nd pattern for partition0. Will change according to the number of patterns to be executed.
82001100 00080140	STCU_LB_CTRL_1	Next pointer to LBIST partition2, parallel mode, CSM bit(31st) can be played with to run in serial mode
00001388 00080144	STCU_LB_PCS_1	Number of patterns -5000 for partition1
63039729 00080150	STCU_LB_MISREL_1	MISR lower 32 bits for 5000nd pattern for partition1. Will change according to the number of patterns to be executed.
F5D1D7E8 00080154	STCU_LB_MISREH_1	MISR upper 32 bits for 5000nd pattern for partition1. Will change according to the number of patterns to be executed.
83001100 00080180	STCU_LB_CTRL_2	Next pointer to LBIST partition3, parallel mode, CSM bit(31st) can be played with to run in serial mode
00001388 00080184	STCU_LB_PCS_2	number of patterns -5000 for partition2
45DF8B4B 00080190	STCU_LB_MISREL_2	MISR lower 32 bits for 5000nd pattern for partition2. Will change according to the number of patterns to be executed.
966615B8 00080194	STCU_LB_MISREH_2	MISR upper 32 bits for 5000nd pattern for partition2. Will change according to the number of patterns to be executed.

Table 115. Full Selftest configuration 3 - Offline (continued)

DCF Value	Register Name	Comment
10001100 000801C0	STCU_LB_CTRL_3	Next pointer to mbist0 to indicate MBIST run after LBIST
00001388 000801C4	STCU_LB_PCS_3	Number of patterns -5000 for partition3
B70EE894 000801D0	STCU_LB_MISREL_3	MISR lower 32 bits for 5000nd pattern for partition3. Will change according to the number of patterns to be executed.
B3E1B779 000801D4	STCU_LB_MISREH_3	MISR upper 32 bits for 5000nd pattern for partition3. Will change according to the number of patterns to be executed.
2C015674 00080008	Key2	Key2 given again to avoid key expiry
2C015674 00080008	Key2	—
91000000 00080600	MBIST_CTRL0	Next pointer set to Mbist1, CSM bist set to '1' to indicate parallel running of MBIST0 and MBIST1
92000000 00080604	MBIST_CTRL1	Next pointer set to Mbist2, CSM bist set to '1' to indicate parallel running of MBIST1 and MBIST2
93000000 00080608	MBIST_CTRL2	Next pointer set to Mbist3, CSM bist set to '1' to indicate parallel running of MBIST2 and MBIST3
94000000 0008060C	MBIST_CTRL3	Next pointer set to Mbist4, CSM bist set to '1' to indicate parallel running of MBIST3 and MBIST4
95000000 00080610	MBIST_CTRL4	Next pointer set to Mbist5, CSM bist set to '1' to indicate parallel running of MBIST4 and MBIST5
96000000 00080614	MBIST_CTRL5	Next pointer set to Mbist6, CSM bist set to '1' to indicate parallel running of MBIST5 and MBIST6
97000000 00080618	MBIST_CTRL6	Next pointer set to Mbist7, CSM bist set to '1' to indicate parallel running of MBIST6 and MBIST7
98000000 0008061C	MBIST_CTRL7	Next pointer set to Mbist8, CSM bist set to '1' to indicate parallel running of MBIST7 and MBIST8
99000000 00080620	MBIST_CTRL8	Next pointer set to Mbist9, CSM bist set to '1' to indicate parallel running of MBIST8 and MBIST9
9A000000 00080624	MBIST_CTRL9	Next pointer set to Mbist10, CSM bist set to '1' to indicate parallel running of MBIST9 and MBIST10
9B000000 00080628	MBIST_CTRL10	Next pointer set to Mbist11, CSM bist set to '1' to indicate parallel running of MBIST10 and MBIST11
9C000000 0008062C	MBIST_CTRL11	Next pointer set to Mbist12, CSM bist set to '1' to indicate parallel running of MBIST11 and MBIST12
9D000000 00080630	MBIST_CTRL12	Next pointer set to Mbist13, CSM bist set to '1' to indicate parallel running of MBIST12 and MBIST13
9E000000 00080634	MBIST_CTRL13	Next pointer set to Mbist14, CSM bist set to '1' to indicate parallel running of MBIST13 and MBIST14
9F000000 00080638	MBIST_CTRL14	Next pointer set to Mbist15, CSM bist set to '1' to indicate parallel running of MBIST14 and MBIST15
A0000000 0008063C	MBIST_CTRL15	Next pointer set to Mbist16, CSM bist set to '1' to indicate parallel running of MBIST15 and MBIST16

Table 115. Full Selftest configuration 3 - Offline (continued)

DCF Value	Register Name	Comment
A1000000 00080640	MBIST_CTRL16	Next pointer set to Mbist17, CSM bist set to '1' to indicate parallel running of MBIST16 and MBIST17
A2000000 00080644	MBIST_CTRL17	Next pointer set to Mbist18, CSM bist set to '1' to indicate parallel running of MBIST17 and MBIST18
A3000000 00080648	MBIST_CTRL18	Next pointer set to Mbist19, CSM bist set to '1' to indicate parallel running of MBIST18 and MBIST19
A4000000 0008064C	MBIST_CTRL19	Next pointer set to Mbist20, CSM bist set to '1' to indicate parallel running of MBIST19 and MBIST20
7F000000 0008064F	MBIST_CTRL20	Next pointer set to null pointer to indicate end of selftest
00000201 00080000	STCU_RUN	Selftest triggered, Mbist run with PLL, LBIST with RC

Table 116: Full Selftest configuration 3 - Online

Register Value	Register Name	Comment
753F924E	STCU_KEY1	Key1 for Online Mode
8AC06DB1	STCU_KEY2	Key2 for Online Mode
005A0001	STCU_CFG	Pointer set for Lbist partition0, Mbist reduced algo(PMOSEN bit set to '0'), PMOSEN and MBU can be played with to chose different algo and mbist coverage, WRP bit set to '0' to allow IPS access
FFFFFFF0	Watchdog	—
81001100	STCU_LB_CTRL_0	Next pointer to LBIST partition1, parallel mode, CSM bit(31st) can be played with to run in serial mode
00001388	STCU_LB_PCS_0	Number of patterns -5000 for partition0
E8BC505E	STCU_LB_MISRELSW_0	MISR lower 32 bits for 5000nd pattern for partition0. Will change according to the number of patterns to be executed.
D15C99CC	STCU_LB_MISREHSW_0	MISR upper 32 bits for 5000nd pattern for partition0. Will change according to the number of patterns to be executed.
82001100	STCU_LB_CTRL_1	Next pointer to LBIST partition2, parallel mode, CSM bit(31st) can be played with to run in serial mode
00001388	STCU_LB_PCS_1	Number of patterns -5000 for partition1
63039729	STCU_LB_MISRELSW_1	MISR lower 32 bits for 5000nd pattern for partition1. Will change according to the number of patterns to be executed.
F5D1D7E8	STCU_LB_MISREHSW_1	MISR upper 32 bits for 5000nd pattern for partition1. Will change according to the number of patterns to be executed.
83001100	STCU_LB_CTRL_2	Next pointer to LBIST partition3, parallel mode, CSM bit(31st) can be played with to run in serial mode
00001388	STCU_LB_PCS_2	Number of patterns -5000 for partition2
45DF8B4B	STCU_LB_MISRELSW_2	MISR lower 32 bits for 5000nd pattern for partition2. Will change according to the number of patterns to be executed.

Table 116: Full Selftest configuration 3 - Online (continued)

Register Value	Register Name	Comment
966615B8	STCU_LB_MISREHSW_2	MISR upper 32 bits for 5000nd pattern for partition2. Will change according to the number of patterns to be executed.
10001100	STCU_LB_CTRL_3	Next pointer to mbist0 to indicate MBIST run after LBIST
00001388	STCU_LB_PCS_3	Number of patterns -5000 for partition3
B70EE894	STCU_LB_MISRELSW_3	MISR lower 32 bits for 5000nd pattern for partition3. Will change according to the number of patterns to be executed.
B3E1B779	STCU_LB_MISREHSW_3	MISR upper 32 bits for 5000nd pattern for partition3. Will change according to the number of patterns to be executed.
91000000	MBIST_CTRL0	Next pointer set to Mbist1, CSM bist set to '1' to indicate parallel running of MBIST0 and MBIST1
92000000	MBIST_CTRL1	Next pointer set to Mbist2, CSM bist set to '1' to indicate parallel running of MBIST1 and MBIST2
93000000	MBIST_CTRL2	Next pointer set to Mbist3, CSM bist set to '1' to indicate parallel running of MBIST2 and MBIST3
94000000	MBIST_CTRL3	Next pointer set to Mbist4, CSM bist set to '1' to indicate parallel running of MBIST3 and MBIST4
95000000	MBIST_CTRL4	Next pointer set to Mbist5, CSM bist set to '1' to indicate parallel running of MBIST4 and MBIST5
96000000	MBIST_CTRL5	Next pointer set to Mbist6, CSM bist set to '1' to indicate parallel running of MBIST5 and MBIST6
97000000	MBIST_CTRL6	Next pointer set to Mbist7, CSM bist set to '1' to indicate parallel running of MBIST6 and MBIST7
98000000	MBIST_CTRL7	Next pointer set to Mbist8, CSM bist set to '1' to indicate parallel running of MBIST7 and MBIST8
99000000	MBIST_CTRL8	Next pointer set to Mbist9, CSM bist set to '1' to indicate parallel running of MBIST8 and MBIST9
9A000000	MBIST_CTRL9	Next pointer set to Mbist10, CSM bist set to '1' to indicate parallel running of MBIST9 and MBIST10
9B000000	MBIST_CTRL10	Next pointer set to Mbist11, CSM bist set to '1' to indicate parallel running of MBIST10 and MBIST11
9C000000	MBIST_CTRL11	Next pointer set to Mbist12, CSM bist set to '1' to indicate parallel running of MBIST11 and MBIST12
9D000000	MBIST_CTRL12	Next pointer set to Mbist13, CSM bist set to '1' to indicate parallel running of MBIST12 and MBIST13
9E000000	MBIST_CTRL13	Next pointer set to Mbist14, CSM bist set to '1' to indicate parallel running of MBIST13 and MBIST14
9F000000	MBIST_CTRL14	Next pointer set to Mbist15, CSM bist set to '1' to indicate parallel running of MBIST14 and MBIST15
A0000000	MBIST_CTRL15	Next pointer set to Mbist16, CSM bist set to '1' to indicate parallel running of MBIST15 and MBIST16

Table 116: Full Selftest configuration 3 - Online (continued)

Register Value	Register Name	Comment
A1000000	MBIST_CTRL16	Next pointer set to Mbist17, CSM bist set to '1' to indicate parallel running of MBIST16 and MBIST17
A2000000	MBIST_CTRL17	Next pointer set to Mbist18, CSM bist set to '1' to indicate parallel running of MBIST17 and MBIST18
A3000000	MBIST_CTRL18	Next pointer set to Mbist19, CSM bist set to '1' to indicate parallel running of MBIST18 and MBIST19
A4000000	MBIST_CTRL19	Next pointer set to Mbist20, CSM bist set to '1' to indicate parallel running of MBIST19 and MBIST20
7F000000	MBIST_CTRL20	Next pointer set to null pointer to indicate end of selftest
00000201	STCU_RUNSW	Selftest triggered, Mbist run with PLL, LBIST with RC

Table 117. Full Selftest configuration 4 - Offline

DCF Value	Register Name	Comment
D3FEA98B 00080008	STCU_KEY1	Key1 in offline mode
2C015674 00080008	STCU_KEY2	Key2 in offline mode
005A0001 0008000C	STCU_CFG	Pointer set for Lbist partition0, Mbist reduced algo (PMOSEN bit set to '0'), PMOSEN and MBU can be played with to chose different algo and mbist coverage, WRP bit set to '0' to allow IPS access
10020046 00080010	PLL configuration	PLL programmed for 35Mhz with input frequency as RC(16 Mhz), PLL output freq = (Input_freq*LDF)/(ODF*IDF), = (Input_freq* 'h46)/('h10*02), Play with ODF factor('h10) to reduce freq
FFFFFFF0 00080014	Watchdog	—
01001100 00080100	STCU_LB_CTRL_0	Next pointer to LBIST partition1, sequential mode
00001388 00080104	STCU_LB_PCS_0	Number of patterns -5000 for partition0
E8BC505E 00080110	STCU_LB_MISREL_0	MISR lower 32 bits for 5000nd pattern for partition0. Will change according to the number of patterns to be executed.
D15C99CC 00080114	STCU_LB_MISREH_0	MISR upper 32 bits for 5000nd pattern for partition0. Will change according to the number of patterns to be executed.
02001100 00080140	STCU_LB_CTRL_1	Next pointer to LBIST partition2, sequential mode
00001388 00080144	STCU_LB_PCS_1	Number of patterns -5000 for partition1
63039729 00080150	STCU_LB_MISREL_1	MISR lower 32 bits for 5000nd pattern for partition1. Will change according to the number of patterns to be executed.
F5D1D7E8 00080154	STCU_LB_MISREH_1	MISR upper 32 bits for 5000nd pattern for partition1. Will change according to the number of patterns to be executed.
03001100 00080180	STCU_LB_CTRL_2	Next pointer to LBIST partition3, sequential mode
00001388 00080184	STCU_LB_PCS_2	Number of patterns -5000 for partition2
45DF8B4B 00080190	STCU_LB_MISREL_2	—

Table 117. Full Selftest configuration 4 - Offline (continued)

DCF Value	Register Name	Comment
966615B8 00080194	STCU_LB_MISREH_2	MISR upper 32 bits for 5000nd pattern for partition2. Will change according to the number of patterns to be executed.
10001100 000801C0	STCU_LB_CTRL_3	Next pointer to mbist0 to indicate MBIST run after LBIST
00001388 000801C4	STCU_LB_PCS_3	Number of patterns -5000 for partition3
B70EE894 000801D0	STCU_LB_MISREL_3	MISR lower 32 bits for 5000nd pattern for partition3. Will change according to the number of patterns to be executed.
B3E1B779 000801D4	STCU_LB_MISREH_3	MISR upper 32 bits for 5000nd pattern for partition3. Will change according to the number of patterns to be executed.
2C015674 00080008	Key2	Key2 given again to avoid key expiry
2C015674 00080008	Key2	—
11000000 00080600	MBIST_CTRL0	Next pointer set to Mbist1, CSM bist set to '0' to indicate sequential running of MBIST0 and MBIST1
12000000 000806004	MBIST_CTRL1	Next pointer set to Mbist2, CSM bist set to '0' to indicate sequential unning of MBIST1 and MBIST2
13000000 000806008	MBIST_CTRL2	Next pointer set to Mbist3, CSM bist set to '0' to indicate sequential running of MBIST2 and MBIST3
14000000 0008060C	MBIST_CTRL3	Next pointer set to Mbist4, CSM bist set to '0' to indicate sequential running of MBIST3 and MBIST4
15000000 00080610	MBIST_CTRL4	Next pointer set to Mbist5, CSM bist set to '0' to indicate sequential running of MBIST4 and MBIST5
16000000 00080614	MBIST_CTRL5	Next pointer set to Mbist6, CSM bist set to '0' to indicate sequential running of MBIST5 and MBIST6
17000000 00080618	MBIST_CTRL6	Next pointer set to Mbist7, CSM bist set to '0' to indicate sequential running of MBIST6 and MBIST7
18000000 0008061C	MBIST_CTRL7	Next pointer set to Mbist8, CSM bist set to '0' to indicate sequential running of MBIST7 and MBIST8
19000000 00080620	MBIST_CTRL8	Next pointer set to Mbist9, CSM bist set to '0' to indicate sequential running of MBIST8 and MBIST9
1A000000 00080624	MBIST_CTRL9	Next pointer set to Mbist10, CSM bist set to '0' to indicate sequential running of MBIST9 and MBIST10
1B000000 00080628	MBIST_CTRL10	Next pointer set to Mbist11, CSM bist set to '0' to indicate sequential running of MBIST10 and MBIST11
1C000000 0008062C	MBIST_CTRL11	Next pointer set to Mbist12, CSM bist set to '0' to indicate sequential running of MBIST11 and MBIST12
1D000000 00080630	MBIST_CTRL12	Next pointer set to Mbist13, CSM bist set to '0' to indicate sequential running of MBIST12 and MBIST13
1E000000 00080634	MBIST_CTRL13	Next pointer set to Mbist14, CSM bist set to '0' to indicate sequential running of MBIST13 and MBIST14
1F000000 00080638	MBIST_CTRL14	Next pointer set to Mbist15, CSM bist set to '0' to indicate sequential running of MBIST14 and MBIST15

Table 117. Full Selftest configuration 4 - Offline (continued)

DCF Value	Register Name	Comment
20000000 0008063C	MBIST_CTRL15	Next pointer set to Mbist16, CSM bist set to '0' to indicate sequential running of MBIST15 and MBIST16
21000000 00080640	MBIST_CTRL16	Next pointer set to Mbist17, CSM bist set to '0' to indicate sequential running of MBIST16 and MBIST17
22000000 00080644	MBIST_CTRL17	Next pointer set to Mbist18, CSM bist set to '1' to indicate sequential running of MBIST17 and MBIST18
23000000 00080648	MBIST_CTRL18	Next pointer set to Mbist19, CSM bist set to '0' to indicate sequential running of MBIST18 and MBIST19
24000000 0008064C	MBIST_CTRL19	Next pointer set to Mbist20, CSM bist set to '0' to indicate sequential running of MBIST19 and MBIST20
7F000000 00080650	MBIST_CTRL20	Next pointer set to null pointer to indicate end of selftest
00000301 00080000	STCU_RUN	Selftest triggered, Mbist,Lbist run with PLL

Table 118. Full Selftest configuration 4 - Online

Register Value	Register Name	Comment
753F924E	STCU_KEY1	Key1 for Online mode
8AC06DB1	STCU_KEY2	Key2 for Online mode
005A0001	STCU_CFG	Pointer set for Lbist partition0, Mbist reduced algo(PMOSEN bit set to '0'), PMOSEN and MBU can be played with to chose different algo and mbist coverage, WRP bit set to '0' to allow IPS access
FFFFFFF0	Watchdog	—
01001100	STCU_LB_CTRL_0	Next pointer to LBIST partition1, sequential mode
00001388	STCU_LB_PCS_0	Number of patterns -5000 for partition0
E8BC505E	STCU_LB_MISRELSW_0	MISR lower 32 bits for 5000nd pattern for partition0. Will change according to the number of patterns to be executed.
D15C99CC	STCU_LB_MISREHSW_0	MISR upper 32 bits for 5000nd pattern for partition0. Will change according to the number of patterns to be executed.
02001100	STCU_LB_CTRL_1	Next pointer to LBIST partition2, sequential mode
00001388	STCU_LB_PCS_1	Number of patterns -5000 for partition1
63039729	STCU_LB_MISRELSW_1	MISR lower 32 bits for 5000nd pattern for partition1. Will change according to the number of patterns to be executed.
F5D1D7E8	STCU_LB_MISREHSW_1	MISR upper 32 bits for 5000nd pattern for partition1. Will change according to the number of patterns to be executed.
03001100	STCU_LB_CTRL_2	Next pointer to LBIST partition3, sequential mode
00001388	STCU_LB_PCS_2	Number of patterns -5000 for partition2
45DF8B4B	STCU_LB_MISRELSW_2	—
966615B8	STCU_LB_MISREHSW_2	MISR upper 32 bits for 5000nd pattern for partition2. Will change according to the number of patterns to be executed.

Table 118. Full Selftest configuration 4 - Online (continued)

Register Value	Register Name	Comment
10001100	STCU_LB_CTRL_3	Next pointer to mbist0 to indicate MBIST run after LBIST
00001388	STCU_LB_PCS_3	Number of patterns -5000 for partition3
B70EE894	STCU_LB_MISRELSW_3	MISR lower 32 bits for 5000nd pattern for partition3. Will change according to the number of patterns to be executed.
B3E1B779	STCU_LB_MISREHSW_3	MISR upper 32 bits for 5000nd pattern for partition3. Will change according to the number of patterns to be executed.
11000000	MBIST_CTRL0	Next pointer set to Mbist1, CSM bist set to '0' to indicate sequential running of MBIST0 and MBIST1
12000000	MBIST_CTRL1	Next pointer set to Mbist2, CSM bist set to '0' to indicate sequential unning of MBIST1 and MBIST2
13000000	MBIST_CTRL2	Next pointer set to Mbist3, CSM bist set to '0' to indicate sequential running of MBIST2 and MBIST3
14000000	MBIST_CTRL3	Next pointer set to Mbist4, CSM bist set to '0' to indicate sequential running of MBIST3 and MBIST4
15000000	MBIST_CTRL4	Next pointer set to Mbist5, CSM bist set to '0' to indicate sequential running of MBIST4 and MBIST5
16000000	MBIST_CTRL5	Next pointer set to Mbist6, CSM bist set to '0' to indicate sequential running of MBIST5 and MBIST6
17000000	MBIST_CTRL6	Next pointer set to Mbist7, CSM bist set to '0' to indicate sequential running of MBIST6 and MBIST7
18000000	MBIST_CTRL7	Next pointer set to Mbist8, CSM bist set to '0' to indicate sequential running of MBIST7 and MBIST8
19000000	MBIST_CTRL8	Next pointer set to Mbist9, CSM bist set to '0' to indicate sequential running of MBIST8 and MBIST9
1A000000	MBIST_CTRL9	Next pointer set to Mbist10, CSM bist set to '0' to indicate sequential running of MBIST9 and MBIST10
1B000000	MBIST_CTRL10	Next pointer set to Mbist11, CSM bist set to '0' to indicate sequential running of MBIST10 and MBIST11
1C000000	MBIST_CTRL11	Next pointer set to Mbist12, CSM bist set to '0' to indicate sequential running of MBIST11 and MBIST12
1D000000	MBIST_CTRL12	Next pointer set to Mbist13, CSM bist set to '0' to indicate sequential running of MBIST12 and MBIST13
1E000000	MBIST_CTRL13	Next pointer set to Mbist14, CSM bist set to '0' to indicate sequential running of MBIST13 and MBIST14
1F000000	MBIST_CTRL14	Next pointer set to Mbist15, CSM bist set to '0' to indicate sequential running of MBIST14 and MBIST15
20000000	MBIST_CTRL15	Next pointer set to Mbist16, CSM bist set to '0' to indicate sequential running of MBIST15 and MBIST16
21000000	MBIST_CTRL16	Next pointer set to Mbist17, CSM bist set to '0' to indicate sequential running of MBIST16 and MBIST17
22000000	MBIST_CTRL17	Next pointer set to Mbist18, CSM bist set to '1' to indicate sequential running of MBIST17 and MBIST18

Table 118. Full Selftest configuration 4 - Online (continued)

Register Value	Register Name	Comment
23000000	MBIST_CTRL18	Next pointer set to Mbist19, CSM bist set to '0' to indicate sequential running of MBIST18 and MBIST19
24000000	MBIST_CTRL19	Next pointer set to Mbist20, CSM bist set to '0' to indicate sequential running of MBIST19 and MBIST20
7F000000	MBIST_CTRL20	Next pointer set to null pointer to indicate end of selftest
00000301	STCU_RUNSW	Selftest triggered, Mbist,Lbist run with PLL

Table 119. Full Selftest configuration 5 - Offline

DCF Value	Register Name	Comment
D3FEA98B 00080008	STCU_KEY1	Key1 in offline mode
2C015674 00080008	STCU_KEY2	Key2 in offline mode
005A0001 0008000C	STCU_CFG	Pointer set for Lbist partition0, Mbist reduced algo(PMOSEN bit set to '0'), PMOSEN and MBU can be played with to chose different algo and mbist coverage, WRP bit set to '0' to allow IPS access
10020046 00080010	PLL configuration	PLL programmed for 35Mhz with input frequency as RC(16 Mhz), PLL output freq = (Input_freq*LDF)/(ODF*IDF), = (Input_freq* 'h46)/('h10*02), Play with ODF factor('h10) to reduce freq
FFFFFFF0 00080014	Watchdog	—
81001100 00080100	STCU_LB_CTRL_0	Next pointer to LBIST partition1, parallel mode, CSM bit(31st) can be played with to run in serial mode
00001388 00080104	STCU_LB_PCS_0	Number of patterns -5000 for partition0
E8BC505E 00080110	STCU_LB_MISREL_0	MISR lower 32 bits for 5000nd pattern for partition0. Will change according to the number of patterns to be executed.
D15C99CC 00080114	STCU_LB_MISREH_0	MISR upper 32 bits for 5000nd pattern for partition0. Will change according to the number of patterns to be executed.
02001100 00080140	STCU_LB_CTRL_1	Next pointer to LBIST partition2, sequential mode, CSM bit(31st)
00001388 00080144	STCU_LB_PCS_1	Number of patterns -5000 for partition1
63039729 00080150	STCU_LB_MISREL_1	MISR lower 32 bits for 5000nd pattern for partition1. Will change according to the number of patterns to be executed.
F5D1D7E8 00080154	STCU_LB_MISREH_1	MISR upper 32 bits for 5000nd pattern for partition1. Will change according to the number of patterns to be executed.
83001100 00080180	STCU_LB_CTRL_2	Next pointer to LBIST partition3, parallel mode, CSM bit(31st) can be played with to run in serial mode
00001388 00080184	STCU_LB_PCS_2	Number of patterns -5000 for partition2
45DF8B4B 00080190	STCU_LB_MISREL_2	MISR lower 32 bits for 5000nd pattern for partition2. Will change according to the number of patterns to be executed.

Table 119. Full Selftest configuration 5 - Offline (continued)

DCF Value	Register Name	Comment
966615B8 00080194	STCU_LB_MISREH_2	MISR upper 32 bits for 5000nd pattern for partition2. Will change according to the number of patterns to be executed.
10001100 000801C0	STCU_LB_CTRL_3	Next pointer to mbist0 to indicate MBIST run after LBIST
00001388 000801C4	STCU_LB_PCS_3	Number of patterns -5000 for partition3
B70EE894 000801D0	STCU_LB_MISREL_3	MISR lower 32 bits for 5000nd pattern for partition3. Will change according to the number of patterns to be executed.
B3E1B779 000801D4	STCU_LB_MISREH_3	MISR upper 32 bits for 5000nd pattern for partition3. Will change according to the number of patterns to be executed.
2C015674 00080008	Key2	Key2 given again to avoid key expiry
2C015674 00080008	Key2	—
91000000 00080600	MBIST_CTRL0	Next pointer set to Mbist1, CSM bist set to '1' to indicate parallel running of MBIST0 and MBIST1
92000000 00080604	MBIST_CTRL1	Next pointer set to Mbist2, CSM bist set to '1' to indicate parallel running of MBIST1 and MBIST2
93000000 00080608	MBIST_CTRL2	Next pointer set to Mbist3, CSM bist set to '1' to indicate parallel running of MBIST2 and MBIST3
94000000 0008060C	MBIST_CTRL3	Next pointer set to Mbist4, CSM bist set to '1' to indicate parallel running of MBIST3 and MBIST4
95000000 00080610	MBIST_CTRL4	Next pointer set to Mbist5, CSM bist set to '1' to indicate parallel running of MBIST4 and MBIST5
96000000 00080614	MBIST_CTRL5	Next pointer set to Mbist6, CSM bist set to '1' to indicate parallel running of MBIST5 and MBIST6
97000000 00080618	MBIST_CTRL6	Next pointer set to Mbist7, CSM bist set to '1' to indicate parallel running of MBIST6 and MBIST7
98000000 0008061C	MBIST_CTRL7	Next pointer set to Mbist8, CSM bist set to '1' to indicate parallel running of MBIST7 and MBIST8
99000000 00080620	MBIST_CTRL8	Next pointer set to Mbist9, CSM bist set to '1' to indicate parallel running of MBIST8 and MBIST9
9A000000 00080624	MBIST_CTRL9	Next pointer set to Mbist10, CSM bist set to '1' to indicate parallel running of MBIST9 and MBIST10
9B000000 00080628	MBIST_CTRL10	Next pointer set to Mbist11, CSM bist set to '1' to indicate parallel running of MBIST10 and MBIST11
9C000000 0008062C	MBIST_CTRL11	Next pointer set to Mbist12, CSM bist set to '1' to indicate parallel running of MBIST11 and MBIST12
9D000000 00080630	MBIST_CTRL12	Next pointer set to Mbist13, CSM bist set to '1' to indicate parallel running of MBIST12 and MBIST13
9E000000 00080634	MBIST_CTRL13	Next pointer set to Mbist14, CSM bist set to '1' to indicate parallel running of MBIST13 and MBIST14
9F000000 00080638	MBIST_CTRL14	Next pointer set to Mbist15, CSM bist set to '1' to indicate parallel running of MBIST14 and MBIST15

Table 119. Full Selftest configuration 5 - Offline (continued)

DCF Value	Register Name	Comment
A0000000 0008063C	MBIST_CTRL15	Next pointer set to Mbist16, CSM bist set to '1' to indicate parallel running of MBIST15 and MBIST16
A1000000 00080640	MBIST_CTRL16	Next pointer set to Mbist17, CSM bist set to '1' to indicate parallel running of MBIST16 and MBIST17
A2000000 00080644	MBIST_CTRL17	Next pointer set to Mbist18, CSM bist set to '1' to indicate parallel running of MBIST17 and MBIST18
A3000000 00080648	MBIST_CTRL18	Next pointer set to Mbist19, CSM bist set to '1' to indicate parallel running of MBIST18 and MBIST19
A4000000 0008064C	MBIST_CTRL19	Next pointer set to Mbist20, CSM bist set to '1' to indicate parallel running of MBIST19 and MBIST20
7F000000 0008064F	MBIST_CTRL20	Next pointer set to null pointer to indicate end of selftest
00000301 00080000	STCU_RUN	Selftest triggered, Mbist and LBIST run with PLL

Table 120. Full Selftest configuration 5 - Online

Register Value	Register Name	Comment
753F924E	STCU_KEY1	Key1 in Online mode
8AC06DB1	STCU_KEY2	Key2 in Online mode
005A0001	STCU_CFG	Pointer set for Lbist partition0, Mbist reduced algo(PMOSEN bit set to '0'), PMOSEN and MBU can be played with to chose different algo and mbist coverage, WRP bit set to '0' to allow IPS access
FFFFFFF0	Watchdog	—
81001100	STCU_LB_CTRL_0	Next pointer to LBIST partition1, parallel mode, CSM bit(31st) can be played with to run in serial mode
00001388	STCU_LB_PCS_0	Number of patterns -5000 for partition0
E8BC505E	STCU_LB_MISRELSW_0	MISR lower 32 bits for 5000nd pattern for partition0. Will change according to the number of patterns to be executed.
D15C99CC	STCU_LB_MISREHSW_0	MISR upper 32 bits for 5000nd pattern for partition0. Will change according to the number of patterns to be executed.
02001100	STCU_LB_CTRL_1	Next pointer to LBIST partition2, sequential mode, CSM bit(31st)
00001388	STCU_LB_PCS_1	Number of patterns -5000 for partition1
63039729	STCU_LB_MISRELSW_1	MISR lower 32 bits for 5000nd pattern for partition1. Will change according to the number of patterns to be executed.
F5D1D7E8	STCU_LB_MISREHSW_1	MISR upper 32 bits for 5000nd pattern for partition1. Will change according to the number of patterns to be executed.
83001100	STCU_LB_CTRL_2	Next pointer to LBIST partition3, parallel mode, CSM bit(31st) can be played with to run in serial mode
00001388	STCU_LB_PCS_2	Number of patterns -5000 for partition2

Table 120. Full Selftest configuration 5 - Online (continued)

Register Value	Register Name	Comment
45DF8B4B	STCU_LB_MISRELSW_2	MISR lower 32 bits for 5000nd pattern for partition2. Will change according to the number of patterns to be executed.
966615B8	STCU_LB_MISREHSW_2	MISR upper 32 bits for 5000nd pattern for partition2. Will change according to the number of patterns to be executed.
10001100	STCU_LB_CTRL_3	Next pointer to mbist0 to indicate MBIST run after LBIST
00001388	STCU_LB_PCS_3	Number of patterns -5000 for partition3
B70EE894	STCU_LB_MISRELSW_3	MISR lower 32 bits for 5000nd pattern for partition3. Will change according to the number of patterns to be executed.
B3E1B779	STCU_LB_MISREHSW_3	MISR upper 32 bits for 5000nd pattern for partition3. Will change according to the number of patterns to be executed.
91000000	MBIST_CTRL0	Next pointer set to Mbist1, CSM bist set to '1' to indicate parallel running of MBIST0 and MBIST1
92000000	MBIST_CTRL1	Next pointer set to Mbist2, CSM bist set to '1' to indicate parallel running of MBIST1 and MBIST2
93000000	MBIST_CTRL2	Next pointer set to Mbist3, CSM bist set to '1' to indicate parallel running of MBIST2 and MBIST3
94000000	MBIST_CTRL3	Next pointer set to Mbist4, CSM bist set to '1' to indicate parallel running of MBIST3 and MBIST4
95000000	MBIST_CTRL4	Next pointer set to Mbist5, CSM bist set to '1' to indicate parallel running of MBIST4 and MBIST5
96000000	MBIST_CTRL5	Next pointer set to Mbist6, CSM bist set to '1' to indicate parallel running of MBIST5 and MBIST6
97000000	MBIST_CTRL6	Next pointer set to Mbist7, CSM bist set to '1' to indicate parallel running of MBIST6 and MBIST7
98000000	MBIST_CTRL7	Next pointer set to Mbist8, CSM bist set to '1' to indicate parallel running of MBIST7 and MBIST8
99000000	MBIST_CTRL8	Next pointer set to Mbist9, CSM bist set to '1' to indicate parallel running of MBIST8 and MBIST9
9A000000	MBIST_CTRL9	Next pointer set to Mbist10, CSM bist set to '1' to indicate parallel running of MBIST9 and MBIST10
9B000000	MBIST_CTRL10	Next pointer set to Mbist11, CSM bist set to '1' to indicate parallel running of MBIST10 and MBIST11
9C000000	MBIST_CTRL11	Next pointer set to Mbist12, CSM bist set to '1' to indicate parallel running of MBIST11 and MBIST12
9D000000	MBIST_CTRL12	Next pointer set to Mbist13, CSM bist set to '1' to indicate parallel running of MBIST12 and MBIST13
9E000000	MBIST_CTRL13	Next pointer set to Mbist14, CSM bist set to '1' to indicate parallel running of MBIST13 and MBIST14
9F000000	MBIST_CTRL14	Next pointer set to Mbist15, CSM bist set to '1' to indicate parallel running of MBIST14 and MBIST15

Table 120. Full Selftest configuration 5 - Online (continued)

Register Value	Register Name	Comment
A0000000	MBIST_CTRL15	Next pointer set to Mbist16, CSM bist set to '1' to indicate parallel running of MBIST15 and MBIST16
A1000000	MBIST_CTRL16	Next pointer set to Mbist17, CSM bist set to '1' to indicate parallel running of MBIST16 and MBIST17
A2000000	MBIST_CTRL17	Next pointer set to Mbist18, CSM bist set to '1' to indicate parallel running of MBIST17 and MBIST18
A3000000	MBIST_CTRL18	Next pointer set to Mbist19, CSM bist set to '1' to indicate parallel running of MBIST18 and MBIST19
A4000000	MBIST_CTRL19	Next pointer set to Mbist20, CSM bist set to '1' to indicate parallel running of MBIST19 and MBIST20
7F000000	MBIST_CTRL20	Next pointer set to null pointer to indicate end of selftest
00000301	STCU_RUN	Selftest triggered, Mbist and LBIST run with PLL

Table 121. Bypass configuration 6

BIST TYPE	Register Value	Register Name	Comment
Offline	D3FEA98B	STCU_KEY1	Key1 in offline mode
	2C015674	STCU_KEY2	Key2 in offline mode
	00000400	STCU_RUN	Bypass Enabled
Online	753F924E	STCU_KEY1	Key1 in online mode
	8AC06DB1	STCU_KEY2	Key2 in online mode
	00000400	STCU_RUN	Bypass Enabled

7.12 Debug and trace

SPC574Sx microcontrollers have a comprehensive feature set to assist debug and trace. *Chapter 55: Debug and Trace* contains full details as well as the top-level block diagram (*Figure 1132*) of the SPC574Sx debug architecture. The figure shows the general interconnection of functional debug modules.

7.12.1 NPC configuration

7.12.1.1 Parameter values

The following table provides the parameter values for this device.

Table 122. Device parameter values

Parameter	QFP100	QFP144
Number of MDO pins available in reduced-port mode	4	4
Number of MDO pins available in full-port mode	—	12

Table 122. Device parameter values (continued)

Parameter	QFP100	QFP144
Number of blocks that input an $\overline{\text{EVTO}}$	1	1
Number of $\overline{\text{MSEO}}$ pins available	2	2

7.12.1.2 Unavailable features

This device does not support an MCKO division factor of 3. Therefore, the setting PCR[MCKO_DIV] = 2 (see *Table 1130: MCKO_DIV Values*) is not available on this device.

Note: Due to package limitation of the TQFP100, the SWT enable feature driven by the EVTI external pad in DCI part is not possible. JCOMB signal is internally fix to high level.

8 Reset and boot

8.1 Introduction

The reset and boot chapter describes the device behavior from the time the external power is applied till the execution of the first instruction of the customer application.

8.1.1 Architecture

8.1.1.1 Reset and boot modules

There are several modules used to sequence through all the steps needed to properly reset the device and prepare it to fetch the first instruction of the user application code. These are the Power Management Controller (see *Chapter 50: Power management controller digital interface (PMC_Dig)*), the Reset generation Module (see *Chapter 47: Reset Generation Module (MC_RGM)*), the Mode Entry (see *Chapter 52: Mode Entry Module (MC_ME)*), the System Status and Configuration Module (see *Chapter 49: System Status and Configuration Module (SSCM)*) and the Self Test Configuration (see *Chapter 66: Self-Test Control Unit (STCU2)*). In addition, during the boot and reset phase some concepts are used. These are: the DCF record mechanism and the BAF code. This chapter gives a top-level overview about the reset functionality, while each module specific chapter goes more into the necessary details.

8.1.1.2 Reset and boot pins

Table 123 shows the pin used for the reset system functionality.

Table 123. Reset pins

Pin	Function	Direction	Active level	Notes
PORST	Destructive reset	Input	Low	—
ESR0	Reset monitor	Output	Low	This pin can be statically configured as a GPIO/AF or ESR0

8.1.2 Functionality

The boot and reset process is organized into two stages:

1. A hardware reset stage, initiated by a reset event, where the device executes the IP initialization
2. A software boot stage

The reset cause can be read from dedicated registers within the MC_RGM module. See *Chapter 47: Reset Generation Module (MC_RGM)* for further details. The next chapters describe in details the hardware and software reset and boot stages.

8.2 Hardware reset stage

Reset sources are organized into two categories: destructive and functional.

- A destructive reset source is associated with an event related to a critical error or dysfunction. A power on belongs to this category.
- A functional reset source is associated with an event related to a less critical error or dysfunction.

Table 124 shows the various destructive reset events and *Table 125* shows the various functional reset events.

The following symbols are used in the tables:

- X: Feature is disabled and cannot be enabled
- O: Feature is enabled and cannot be disabled
- P[D]: Feature is programmable; default is set to “disabled”
- P[E]: Feature is programmable; default is set to “enabled”

8.2.1 Destructive reset events

Table 124 shows the destructive reset sources.Some reset triggers can be programmed via software in order to generate a destructive reset or an alternate event. The choice is done by writing the corresponding bit in the RGM_DERD register. In case an alternate event has been programmed, then through the RGM_DEAR register is possible to define whether the trigger generates a SAFE mode change or an interrupt. After a reset event has occurred, the reset sources that triggered the event are recorded into the RGM_DES register (see chapter *Chapter 47: Reset Generation Module (MC_RGM)* for further details).

Table 124. Destructive reset events

RGM destructive reset sources					
Reset source		Configuration			
		Input	Reset or Alternate event(1)	Alternate event type(2)	
Module	Signal(s)			Safe mode	Interrupt
D_POR	Device PowerOn-Reset (POR)	POR	—	—	—
porst_gen	dest_rst_b: PORST pin reset	D[1]	P[E]	P[E]	P[D]
Reserved		D[2]	—	—	—
D_SOFT_DEST	Software destructive reset(3)	D[3]	O	X	X
fosu	fccu_failure_to_react_b: FCCU failure to react reset	D[4]	O	X	X
D_SUF	STCU2 failure	D[5]	O	X	X
Reserved		D[6:7]	—	—	—
D_EDR	Escalation destructive reset	D[8]	O	X	X
D_FIF	System watchdog timeout during Flash initialization	D[9]	O	X	X
D_JTAG	JTAG destructive reset request	D[10]	O	X	X

Table 124. Destructive reset events (continued)

RGM destructive reset sources					
Reset source		Configuration			
		Input	Reset or Alternate event(1)	Alternate event type(2)	
Module	Signal(s)			Safe mode	Interrupt
Reserved		D[11:13]	—	—	—
SSCM	SSCM destructive reset request	D[14]	O	X	X
Reserved		D[15:22]	—	—	—
D_TSR	Temperature sensor reset	D[23]	P[E]	P[E]	P[D]
D_VOR	Voltage out of range from LVDs and/or HVDs	D[24]	O	X	X
Reserved		D[25:31]	—	—	—

1. Set by RGM_DERD register
2. Set by RGM_DEAR register
3. Triggered by ME

8.2.2 Functional reset sources

Table 125 shows the functional reset sources.Some reset triggers can be programmed via software in order to generate a functional reset or an alternate event. The choice is done by writing the corresponding bit in the RGM_FERD registers. In case a functional reset event has been selected, then it is possible to select a Short or Long Functional Reset, by writing the corresponding bit in the RGM_FESS register. In case an alternate event has been programmed, then through the RGM_FEAR register is possible to define whether the trigger generates a SAFE mode change or an interrupt. After a reset event has occurred, the reset sources that triggered the event are recorded into the RGM_FES register (see chapter *Chapter 47: Reset Generation Module (MC_RGM)* for further details).

Table 125. Functional reset events

RGM functional reset sources							
Reset source		Configuration					
		Input	Reset or Alternate event(1)	Functional Reset type(2)		Alternate event type(3)	
Module	Signal(s)			Long	Short	Safe mode	Interrupt
Reserved		F[0]	—	—	—	—	—
F_WAKEUP	Wake-up unit(4)	F[1]	P[E]	P[E]	P[D]	P[E]	P[D]
F_ST_DONE	Self-test functional reset	F[2]	O	O	X	X	X
F_SOFT_FUNC	Online Self Test completion OR Software device reset(5)	F[3]	P[E]	P[E]	P[D]	X	X

Table 125. Functional reset events (continued)

RGM functional reset sources							
Reset source		Configuration					
		Input	Reset or Alternate event[1]	Functional Reset type[2]		Alternate event type[3]	
Module	Signal(s)			Long	Short	Safe mode	Interrupt
Reserved		F[4]	—	—	—	—	—
F_FCCU_HARD	Hard reaction request	F[5]	E	O	X	X	X
F_FCCU_SOFT	Soft reaction request	F[6]	P[E]	X	O	O	X
Reserved		F[7:9]	—	—	—	—	—
F_JTAG	JTAG reset	F[10]	P[E]	P[E]	P[D]	X	X
Reserved		F[11:22]	—	—	—	—	—
F_TSR_FUNC	Temperature sensor trigger	F[23]	P[E]	P[E]	P[D]	P[E]	P[D]
F_VOR_FUNC	Voltage out of range from LVDs and/or HVDs	F[24]	P[E]	P[E]	P[D]	P[E]	P[D]
Reserved		F[25:31]	—	—	—	—	—

1. Set by RGM_FERD register
2. Set by RGM_FESS register
3. Set by RGM_FEAR register
4. Triggered by NMI
5. Triggered by ME OR when ONLINE Self TEST completes

8.2.3 Functional reset escalation

'Functional' reset escalation can be used to generate a 'destructive' reset if a number of 'functional' resets, triggered by any of the functional reset input signals, have occurred between software writes to a specific RGM register. This function is enabled by writing a non-zero value a specific RGM register[d].

Once 'functional' reset escalation has been enabled, the RGM increases a counter on each 'functional' or external reset which causes a reset sequence to be initiated (that is entrance into PHASE1 or PHASE3 from the IDLE phase). This counter is cleared on a write of any value to a specific RGM register and on any power-on or 'destructive' reset. If the counter reaches the value set into a specific RGM register, the RGM starts a reset sequence at PHASE0.

d. The default value at reset for this register is 4

8.2.4 Destructive reset escalation

'Destructive' reset escalation can be used to keep the chip in the reset state until a new power-on triggers a reset sequence if a number of 'destructive' resets, triggered by any of the destructive reset input signals, has occurred between software writes to a specific RGM register. This function is enabled by writing a non-zero value to a specific RGM register [e]. Once 'destructive' reset escalation has been enabled, the RGM increases a counter on each 'destructive' reset which causes a reset sequence to be initiated (i.e., entrance into PHASE0 from the IDLE phase) or an ongoing reset sequence to restart (i.e., entrance into PHASE0 from PHASE1, PHASE2, or PHASE3). This counter is cleared on a write of any value a specific RGM register and on any power-on reset. If the counter reaches the value set into a specific RGM register, the MC_RGM enters reset PHASE0 and stays there until the next power-on reset occurs.

8.3 Reset sequence

The hardware reset stage is handled by the RGM module through a state machine. The RGM executes the device initialization through a sequence of phases. At the end of the hardware reset stage, the device starts software boot stage. The software boot stage is actually the very first code executed by the processor. This code, called BAF, is programmed by ST in a specific Flash location and cannot be modified.

e. The default value at reset for this register is 8

Figure 42. Reset sequence

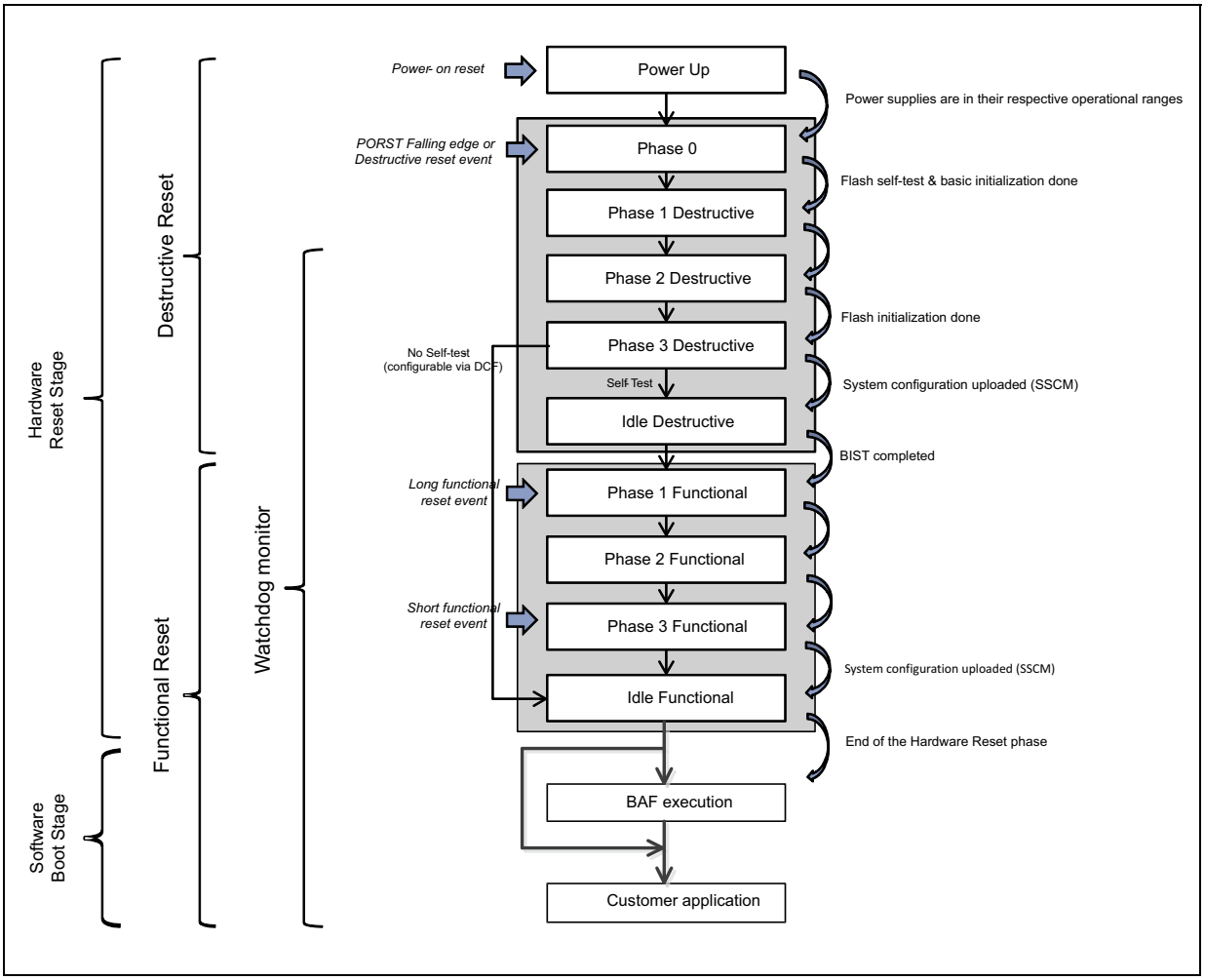


Table 126 summarizes the reset sequence phases.

Table 126. Reset sequence phases

Phase	Description
Power-up	The powerup phase can only be entered upon a power-on reset event. The device will remain in the power-up phase until the Power Management Controller module (PMC) determines that all power supplies are in their respective operational ranges. See *Chapter 10: Power management* for a detailed description of the power monitoring architecture.
Phase 0 DEST	The Reset Phase 0 Destructive is entered upon a falling edge of the PORST pin or a Destructive Reset event. During this phase, the Flash starts its initialization.
Phase 1 DEST	The Reset Phase 1 Destructive is entered after the Reset Phase 0 Destructive is completed.
Phase 2 DEST	The Reset Phase 2 Destructive is entered after the Reset Phase 1 Destructive is completed. During this phase the Flash executes its initialization. The whole operation is monitored by the Watchdog, which will trigger a destructive reset in case of Flash initialization failure.

Table 126. Reset sequence phases (continued)

Phase	Description
Phase 3 DEST	The Reset Phase 3 Destructive is entered after the Reset Phase 2 Destructive is completed. During this phase the SSCM module uploads the system configuration and dispatches to the proper clients (see *Chapter 49: System Status and Configuration Module (SSCM)* for more details). During this phase the CPU reset vector is written into the ME module and the trimming for the analog IPs (such as VR, LVD/HVD, temperature sensor, ADC, I/O, oscillator) are applied. In case no BIST is to be executed, the device can be kept into the Phase 3 DEST by keeping the PORST. This can be used for debugging purposes by allowing external test/development equipment to connect through JTAG to the internal debugger.
Idle DEST	The Idle Phase Destructive is entered after the Reset Phase 3 Destructive is completed in case a BIST execution is requested (see *Chapter 6: Functional safety* for further details). The BIST execution is monitored by a watchdog. In case of BIST failure, a destructive reset will be triggered.
Phase 1 FUNC	The Idle Phase Functional is entered after the Idle Phase Destructive is completed or when a Functional reset event is triggered, provided that the event itself is not configured to trigger a Short-Reset sequence. The system behavior during this phase is the similar to the Phase 1 Destructive.
Phase 2 FUNC	The Reset Phase 2 Functional is entered after the Reset Phase 1 Functional is completed. The system behavior during this phase is similar to the Phase 2 Destructive. The whole operation is monitored by the watchdog, which will trigger a reset in case of Flash initialization failure.
Phase 3 FUNC	The Reset Phase 3 Functional is entered after the Reset Phase 2 Functional is completed or in case a functional reset is triggered and the reset event has been configured as a Short-Reset. During Phase 3 Functional the information retrieved during the During Phase 3 Destructive is read again and compared with the value passed to the various clients. If a mismatch occurs, an error condition is signaled to the FCCU.
Idle FUNC	The Idle Phase Destructive is entered after the Reset Phase 3 Functional, or after the Reset Destructive Phase 3 is completed in case a BIST execution is not requested.

Table 127 shows the status of the various system modules during the hardware reset phase.

Table 127. Module status during the reset phase

Module	Phase								
	Destructive					Functional			
	P0	P1	P2	P3	IDLE	P1	P2	P3[1]	IDLE
IRCOSC	ON	ON	ON	ON	ON	ON	ON	ON	ON
PMC	ON	ON	ON	ON	ON	ON	ON	ON	ON
RGM	ON[1]	ON	ON	ON	ON	ON	ON	ON	ON
System watchdog	RESET	RESET	ON	RESET	LBIST	RESET	ON	RESET	ON

Table 127. Module status during the reset phase (continued)

Module	Phase								
	Destructive					Functional			
	P0	P1	P2	P3	IDLE	P1	P2	P3[1]	IDLE
Flash	RESET	RESET	ON	ON	LBIST	RESET	ON	ON	ON
XOSC	RESET	RESE	RESET	ON	ON	ON	ON	ON	ON
Device configuration	RESET	RESET	RESET	LOAD	HOLD	HOLD	HOLD	HOLD	HOLD
SSCM	RESET	RESET	RESET	ON	ON	RESET	RESET	ON	ON
FCCU	RESET	RESET	CONFIG	CONFIG	LBIST	ON	ON	ON	ON
STCU	RESET	RESET	CONFIG	CONFIG	ON	IDLE	IDLE	IDLE	IDLE
CPU	RESET	RESET	RESET	RESET	LBIST	RESET	RESET	RESET	ON
Other modules and cores	RESET	RESET	RESET	RESET	LBIST	RESET	RESET	RESET	ON

1. RGM is reset only upon a power-on.

8.4 PORST pin functionality

The PORST pin acts as an input to initialize the device to a known start-up state. The detection of a falling edge on PORST will start the reset sequence. For safety reasons, a weak pull-down is always connected to the PORST pin. *Figure 43* shows the effect of keeping PORST low during the destructive reset phase. The effect depends on the BIST enabling status:

- In case the BIST is disabled, the reset sequence does not evolve and remains into the Phase 3 Destructive until the PORST pin is de-asserted low. Note that once the PORST pin is de-asserted low, the reset sequence evolves into the Idle Functional stage
- In case the BIST is enabled, the reset sequence evolves (notwithstanding the PORST pin level) and enters the Idle Phase Destructive once the BIST is completed

Figure 43. PORS Reset pin vs. Phase 3 Destructive

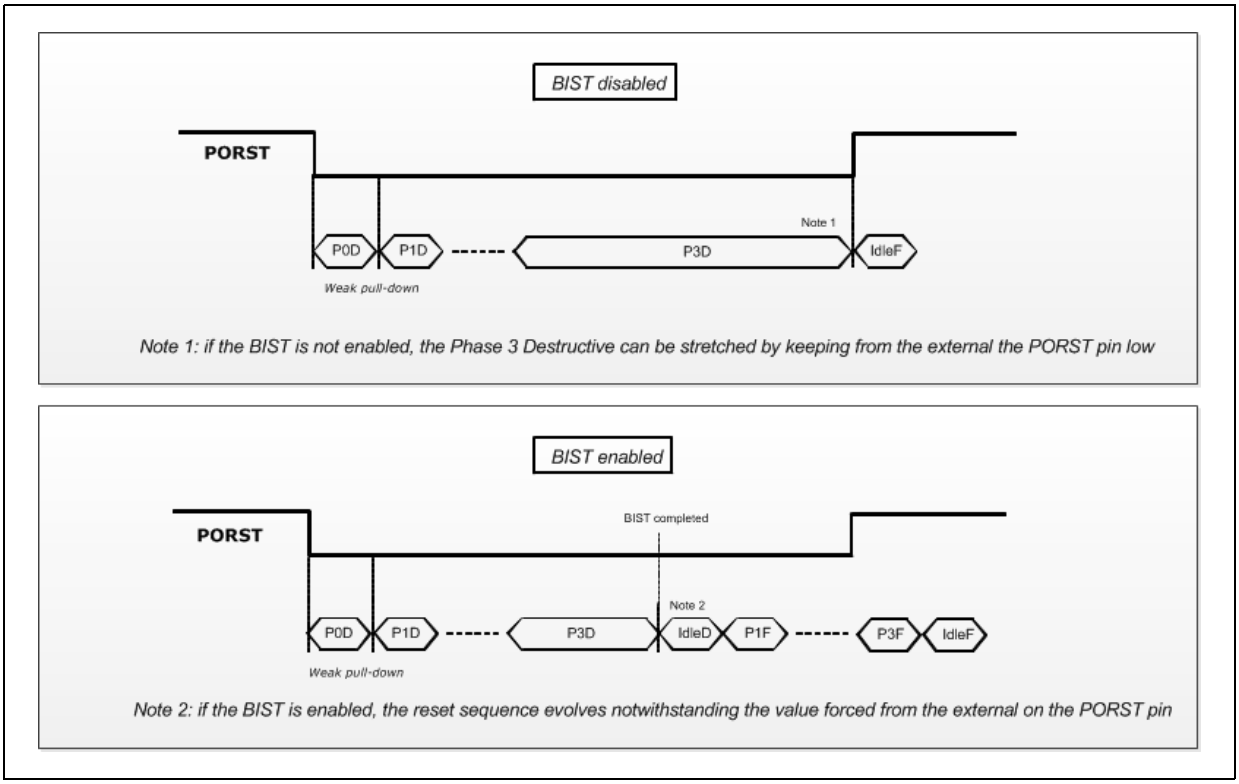


Figure 44 shows the effect of the PORST pin during the functional reset phase. It is possible to stretch the Phase 3 Functional by keeping PORST low. Note that this case is possible only when the BIST is enabled, otherwise the reset sequence would have been formerly stopped at the Phase 3 Destructive, as explained above.

Figure 44. PORS Reset pin vs. Phase 3 Functional

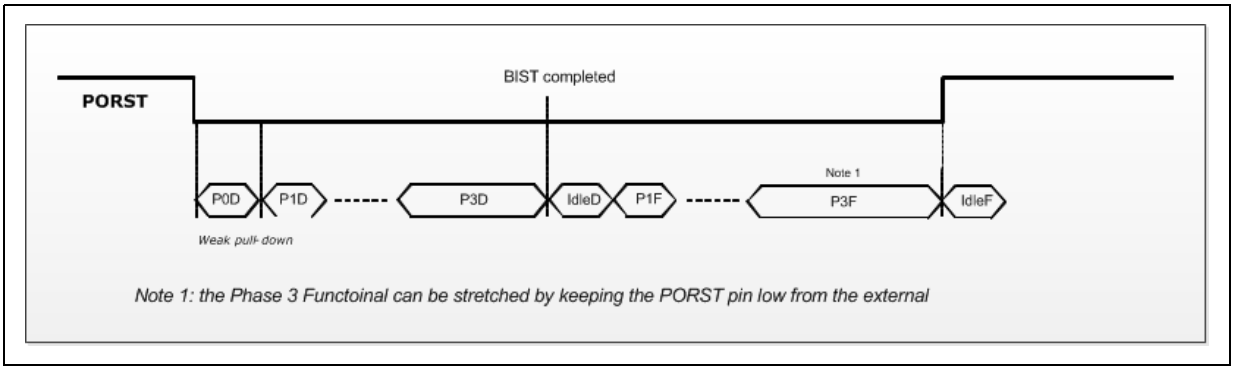


Table 128, *Table 129* and *Table 130* show PORST pin characteristics.

Table 128. PORST electrical characteristics

Type	Symbol	Description	Min	Max	Unit
TTL	VIH	High level input voltage	2.0	—	V
	VIL	Low level input voltage	—	0.8	V
	Vhyst	Input hysteresis voltage	300	—	mV

Table 129. PORST filter characteristics

Description	Min	Max	Unit
Non filtered pulses	—	500	ns
Filtered pulses	2000	—	ns

Table 130. PORST pull-down characteristics

Pull-down	Conditions	Min	Max	Unit
Weak	Vinput = VDD	30	80	uA

8.5 ESR0 pin functionality

This device does not implement a dedicated pin to output a reset signal. Nevertheless, it is possible to statically configure one pin to act as an output reset. The operation is done by reading a specific DCF record, which can be written one time only and it is assigned to a triple-voting set of flip-flops. The DCF is read during the reset phase 3 Destructive and the read value is maintained till the next power on reset. The following list summarizes the static pin configuration:

1. PAD 47 can have a double function (see *Chapter 4: Signal description* for the pad list):
   a) ESR0 reset output only
   b) GPIO/AF
2. The choice between a) or b) is done via DCF reading at every reset
3. The DCF is write-once and stored into a triple-voting set of flip-flops
4. The value read via DCF shall be available to the SW via a register read-only status bit, so that the application can always monitor the correct configuration
5. The value uploaded via DCF is maintained throughout the next Power-On-Reset
6. After a Power-On-Reset the default value is GPIO
7. After a Destructive or Functional reset, the default value is the last one being uploaded via DCF
8. In case the pin is configured as GPIO, during a reset phase the pin shall maintain the same under-reset safe configuration as for all the other GPIOs
9. In case the pin is configured as ESR0, during a reset phase the pin is driven to ‘0’ by the RGM module, while out of reset, the pin is weakly pulled-up. ESR0 is strongly driven during reset and it is weakly pulled up out of reset. So, out of reset , the rise of this pin will depend on the output load. Internal pull up resistance is 75 Kohm (worst case) , so outside resistance should be at least 10 times of this to have a output signal rising to the maximum value of IO supply.

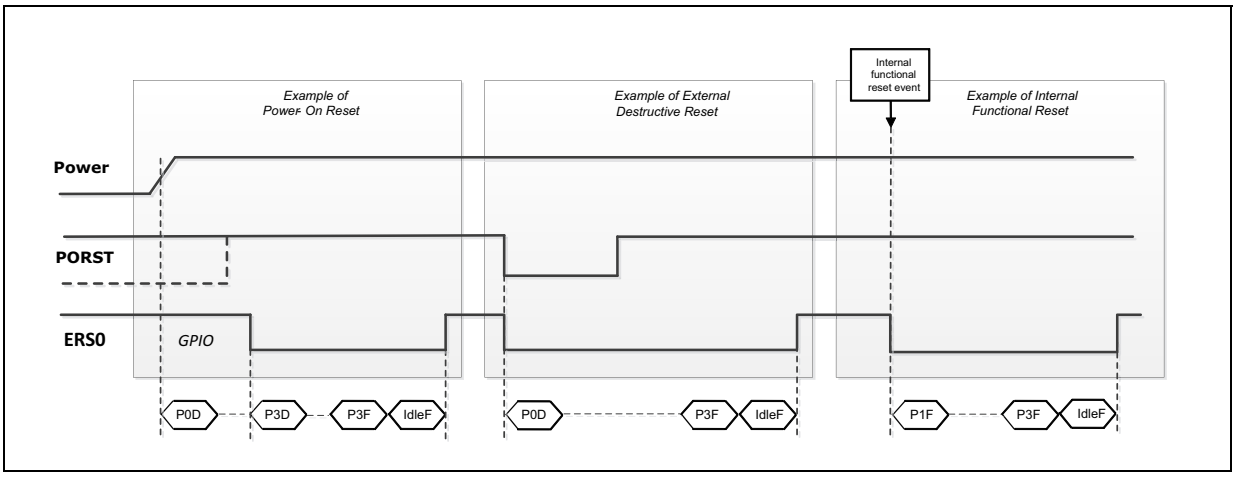
ESR0 pin characteristics: The ESR0 pin has the same electrical characteristic as a GPIO.

8.6 PORST and ESR0 pin status during power on, destructive and functional reset

Figure 45 shows the behavior of the PORST and ESR0/GPIO pins for three scenarios:

1. Power-on
2. PORST forced low from the external (the same case applies for an internal destructive reset)
3. Internal functional reset

Figure 45. PORST reset pin state during the reset sequence



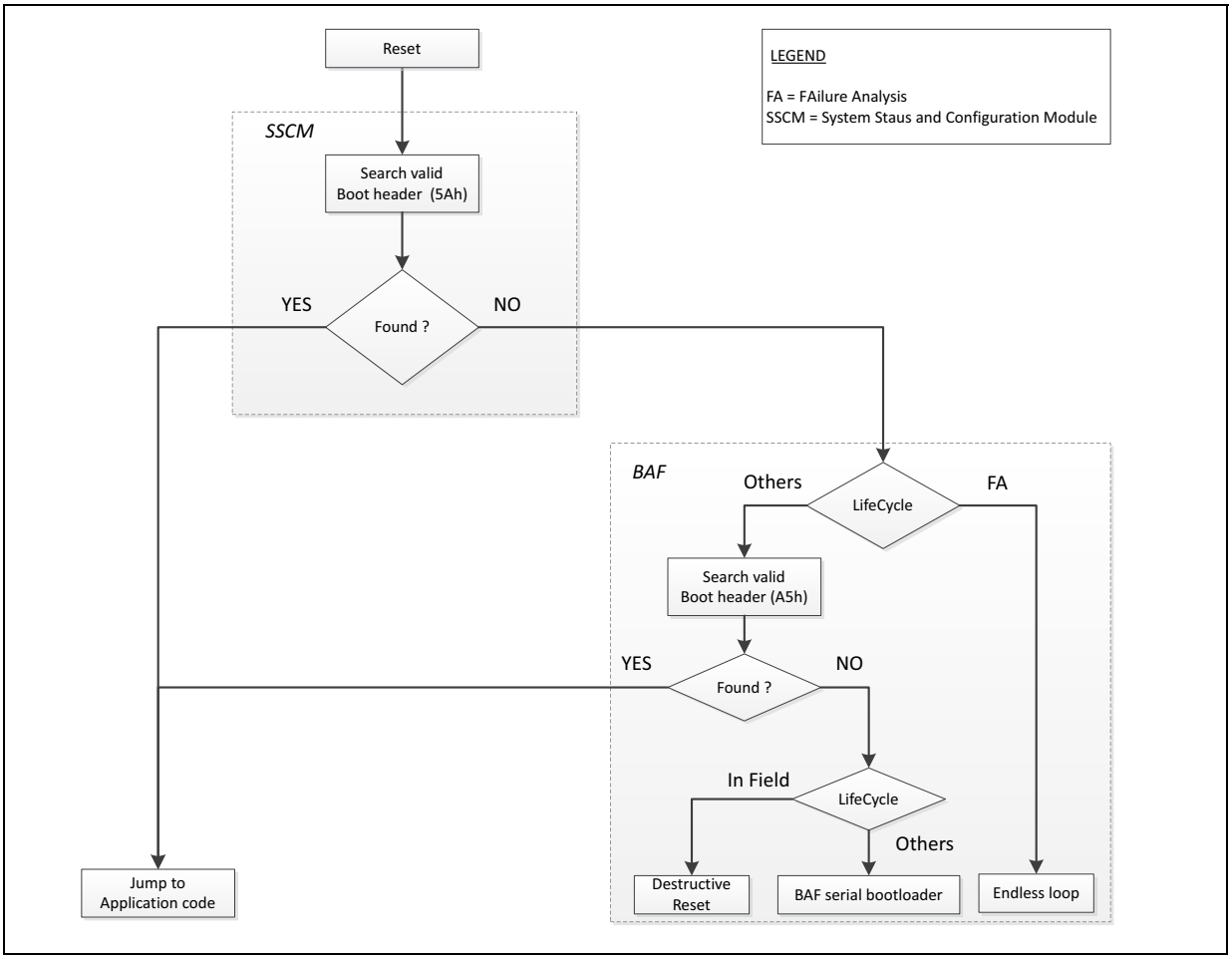
8.7 System boot

After the reset phase is completed, the CPU starts executing at the start address provided by MC_ME peripheral, according to the value stored into the CADDR register. The default value (at the reset) of the CADDR register is forced by the SSCM, according to the system configuration, which is stored in Flash and read by the SSCM from the Flash during the reset.

The system configuration is set as application boot records and is programmed by the customer. It is used to set the customer application entry point after reset or after the BAF execution (if enabled)

Figure 46 shows the the various possible system boots.The application entry point can be retrieved both in HW (by the SSCM) or in SW (by the BAF). In both cases, the application entry point is retrieved by searching for a valid Boot Record, which shall programmed by the customer in one of the pre-determined Flash memory locations looked at by the SSCM or the BAF code. Next chapters give more details about the Boot Record structure and search operation.

Figure 46. System boot sequence



8.8 HW boot record search (SSCM search)

The SSCM module looks for a valid software entry point, called SSCM boot record. The SSCM boot record is organized as shown in the *Table 131*. The “Application start address“ value will be passed to the CADDR register of MC_ME, which is used to drive the CPU start address after the reset phase.

Table 131. SSCM boot record structure

Offset	Field description	
0x0	Boot Record Tag (5Ah)	Reserved
0x4	Application start address	
0x8	Reserved	
0xC	Reserved	

The SCCM searches for a valid boot record at the addresses specified in *Table 132*. A boot record is considered valid if it contains the value 5Ah in the “Boot Record Tag” field.

The “Search Order” column specifies which location is searched first. The SSCM stops the search as soon as a valid boot record is found.

Table 132. SSCM boot record search locations

Search order	Address[1]	Location	SSCM configuration parameter
1 (First)	0x00FC_0000	Flash 16 KB Code Block 0	RCHW_LOC_0
2	0x00FC_4000	Flash 16 KB Code Block 1	RCHW_LOC_1
3	0x00FC_8000	Flash 16 KB Code Block 2	RCHW_LOC_2
4	0x00FC_C000	Flash 16 KB Code Block 3	RCHW_LOC_3
5	0x0100_0000	Flash 256 KB Code Block 0	RCHW_LOC_4
6	0x0104_0000	Flash 256 KB Code Block 1	RCHW_LOC_5
7 (Last)	0x0040_4000	Flash 16 KB BAF block	RCHW_LOC_6

1. The addresses specified in this table refer to the system memory addresses (that is as “seen” by the CPU). In reality, the SCCM directly interfaces the Flash, so the exact addresses to be used in the SSCM RTL code are specified by the Flash block guide.

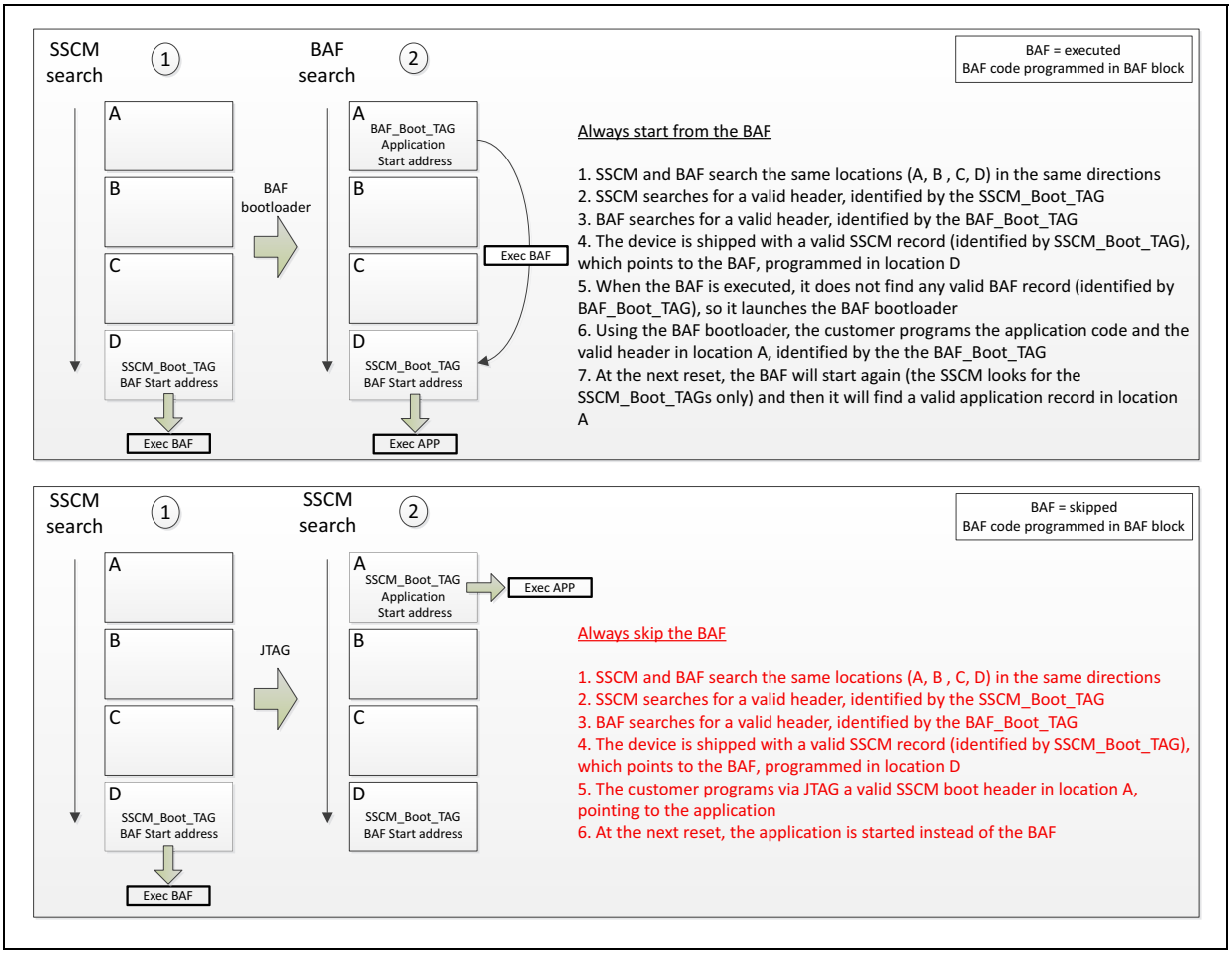
Depending upon the fact that a valid SSCM boot record is found and the content of the SSCM boot record, three scenarios can happens:

1. if a valid Boot Record is programmed by the customer, then the application is executed after the reset from the specified entry point address
2. if no valid Boot Record is programmed by the customer, then the default BAF Boot Record is retrieved and the BAF code is executed after the reset

Figure 47 shows the way to start the BAF or directly the customer application.

Note: For sake of simplicity, the picture shows only four boot record positions.

Figure 47. Valid boot header search



8.9 SW boot record search (BAF search)

The BAF is executed if no valid Boot Record (0x005A0000) is programmed by the customer. In this case a valid SSCM boot record pointing to the BAF code (programmed by factory at the location number 7) is found.

This boot record will point to the BAF entry point. Because the boot record number 7 is the last one being searched by the SSCM, this means that, if no other boot record is programmed, the software will start from the BAF. Notice that the location of the boot record nr. 7 is placed within the BAF block itself, making it not erasable (the BAF block is OTP).

When the BAF code is executed after the reset sequence is completed, a valid application Boot Record for the application entry point is searched. The mechanism is the same executed by the SSCM: a set of fixed locations are analyzed and, if a valid boot record is found, then, at the end of the BAF code, the CPU will start executing from the application entry point.

Note that the valid boot record is identified by a different tag: 5Ah for the SSCM; A5H for the BAF. This will make it possible to program the same boot record locations and launch the application directly after the reset or after the BAF code execution. *Table 133* shows the BAF boot record structure.*Table 134* shows the addresses of the location searched by the

BAF code. *Figure 46* shows a flow chart of the whole reset sequence, including the BAF search for a valid boot header. See *Chapter 48: Boot Assist Flash (BAF)* for additional information about the BAF code.

Table 133. BAF boot record structure

Offset	Field description	
0x0	BAF Boot Record Tag (A5h)	Reserved
0x4	Application start address	
0x8	Reserved	
0xC	Reserved	

Table 134. BAF boot record search locations

Search order	Address	Location
1 (First)	0x00FC_0000	Flash 16 KB Code Block 0
2	0x00FC_4000	Flash 16 KB Code Block 1
3	0x00FC_8000	Flash 16 KB Code Block 2
4	0x00FC_C000	Flash 16 KB Code Block 3
5	0x0100_0000	Flash 256 KB Code Block 0
6 (Last)	0x0104_0000	Flash 256 KB Code Block 1

8.10 Performances

Table 135 shows boot timings.

Table 135. Boot timings

Reset	BIST	BAF	Boot time
Destructive	Disabled	Skipped	500 µs

9 Device Configuration Format (DCF) Records

9.1 Introduction

Device Configuration Format (DCF) records are used to configure certain registers in the device during system boot while the reset signal is asserted. An individual DCF record consists of a pointer to the location of a register internal to the device and the data to be written to that register.

There are two broad categories of DCF records, TEST DCF Records and UTEST DCF Records. TEST DCF records are developed by the factory and are used mainly to program registers involved in trimming trip points for voltage comparators, adjusting analog to digital voltage supplies, trimming oscillator frequencies, and enabling RAM repair. However these records can also be used to write any DCF client. The TEST DCF Records are programmed into TEST flash during production and cannot be modified. TEST flash is not visible to the user.

The second set of DCF records are UTEST DCF records. Some UTEST DCF records are written by the factory and programmed during production testing (see “UTEST flash memory map” table in *Chapter 5: Memory map*). Others are written by the end user and programmed at the same time application code is programmed into the flash memory. User-supplied UTEST DCF records start at the next location in the UTEST memory map following the UTEST DCF records programmed into the flash memory during factory production. The user-defined DCF records set up the initial memory map, assign memory blocks as OTP assign flash blocks to be associated with specific password groups.

System boot is a complex process requiring a considerable amount of initialization to take place before releasing reset. Before the device can be used in an application, the user application code, reset vectors for all of the CPU and the DCF records must be properly programmed into their respective flash memories.

When power is applied to a properly programmed device, the Power Management Controller (PMC) takes control of the device. After the system power supplies have reached predefined levels, the Power Management Controller signals the Reset Generation Module (RGM) to begin the boot sequence. During the boot sequence, the RGM enables another internal module, the System Status and Control Module (SSCM), to read the device configuration records and then write the configuration information to the specified registers.

9.2 DCF clients

DCF clients are 32-bit wide hardware registers inside a module, that receive and store the data from a DCF record. This stored data is used to initialize registers and to configure features. DCF Clients have a default value before any DCF Records are written, and may have special writing constraints, such as, ‘Write Once’ or allowing bits to be written from ‘1’ to ‘0’ only or vice versa. DCF clients need not implement all 32 bits. DCF clients maybe designated ‘TEST DCF record only’, which means that only DCF Records stored in TEST flash can write to the DCF Client.

A list of DCF clients is shown in *Table 138*. (TEST DCF Record Only Clients are not shown as these are not accessible by the user)

The DCF Records select the ‘target’ DCF client by a 30-bit field in the DCF Record consisting of a 15-bit ‘Chip Select’ field and a 15-bit Address field. Each module that

includes DCF clients is assigned a Chip Select during chip definition. The address field is only relevant to the address decoding within that module and may not necessarily relate to the address of a register visible to software.

9.3 DCF records

DCF records appear as contiguous double-word (64-bit) entries programmed in a reserved area of OTP UTEST flash memory beginning at 0x0040_0200.

The structure of a DCF Record is shown in *Figure 48*.

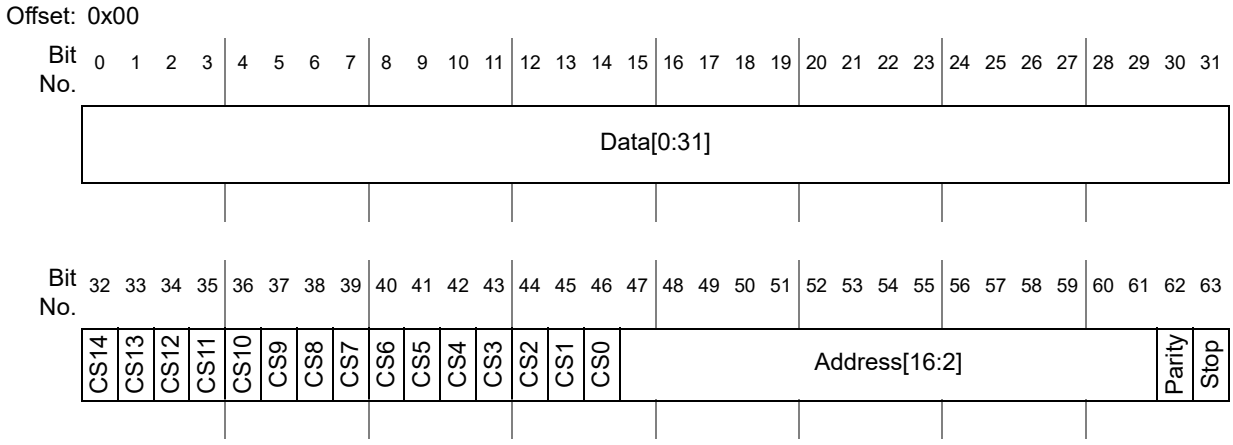


Figure 48. DCF record structure

Table 136. DCF record field description

Field	Description
0:31 Data[0:31]	32 bits of data that is to be written to the DCF Client
32:46 CSn	Chip Select n One Chip select is asserted (‘1’) per DCF record to select the target module for the DCF client. All other Chip Selects should be negated (‘0’)
47:61 Address[16:2]	Address of the DCF client within the selected module **Note:** Address decoding for DCF clients may not match the standard software address map decoding. Details of DCF Client addresses are defined in each module chapter
62 Parity	Parity Bit for the DCF Record **Note:** This bit is NOT implemented for DCF Client written from UTEST
63 Stop	Stop bit This bit indicates the end the list of DCF Records. 0 Not the end of the list 1 End of the list **Note:** The Erased state of flash is 0xFFFF_FFFF_FFFF_FFFF. Therefore the list ends with the first unprogrammed double word. This location can be programmed with a new record to extend the list

The following structure must be present:

- The first record must be a start record.
- DCF records containing configuration data must immediately follow the start record with no blank records between an unprogrammed record that is interpreted as a stop record and no DCF records following that record are processed.
- The end of the configuration records are indicated by the presence of a stop record.

Each record is 64 bits in length. *Figure 49* shows the DCF start record.

Warning: The start record must be placed at the beginning of the DCF area in UTEST flash memory to indicate to the device that the following data records must be processed.

Figure 49. DCF start record

0x00 0:31	**0x04 32:63**
0x05AA55AF	0x00000000

The stop bit designates the last device configuration record in a set of records. Normally, the stop bit is never set in a valid UTEST DCF record programmed during production, since this would prevent additional UTEST records from being appended to the factory programmed UTEST DCF records. However, the flash memory location following the last UTEST DCF record programmed at the factory is an unprogrammed location. Therefore, that location's contents is 0xFFFF_FFFF_FFFF_FFFF. Thus, the stop bit location in this unprogrammed flash memory location is a logic 1, signifying that this is the last DCF record and it is not to be acted upon.

During the boot sequence, the SSCM uses the DCF records to program various registers before reset is released. Some device configuration records are calculated during production testing and programmed into the TEST flash memory area. Other DCF records are developed by the end user and programmed into the UTEST flash area. A device configuration record simply consists of an internal register address and a data field. During system boot, while the reset signal is asserted, the SSCM reads the device configuration records and writes the data to the appropriate internal module registers. Thus, many operational aspects of the device are properly configured when the reset signal is released and normal device operation begins.

The stop record indicates the end of DCF records to be processed. The general format of the stop record is shown in *Figure 50*. Only the stop bit needs to be 1 in order to form a stop record: all other bits are ignored. An unprogrammed location in UTEST flash is interpreted as a stop record.

Figure 50. DCF stop record

0:31	32:62	63
Reserved	Reserved	1

If *n* data records are to be stored in UTEST, the data structure must be as shown in *Table 137*.

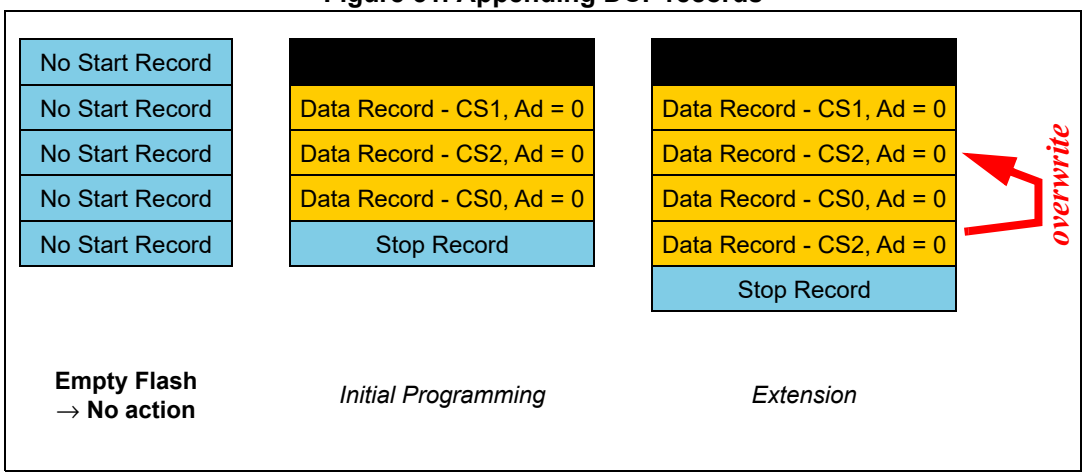
Table 137. Series of DCF records in UTEST flash memory

ADDR offset	DATA			
0x00	0x05AA55AF			
0x04	0x00000000			STOP=0
0x08	WDATA[31:0]			
0x0C	CS[14:0]	ADDR[16:2]	PRTY	STOP=0
0x10	WDATA[31:0]			
0x14	CS[14:0]	ADDR[16:2]	PRTY	STOP=0
...	...			
8n-1 + 0x0	Reserved			
8n-1 + 0x4	Reserved			**1**
8n + 0x0	—			
8n + 0x4				

Warning: There must never be an unprogrammed record in the DCF data structure, as it is interpreted as a stop record and subsequent records are ignored. This allows one to program the records in several sessions, each time appending new records at the end of the list, as shown in *Figure 51*.

It is possible for more than one DCF Record to write to the same DCF client. In this case, as shown in the “Extension” part of *Figure 51*, the later record usually overrides a DCF client value set by a previous record. However, not all DCF clients allow overwrites; this depends on the DCF client implementation.

Figure 51. Appending DCF records



9.3.1 UTEST DCF records

UTEST DCF records are located in the UTEST flash memory area (refer to UTEST flash memory map in the Memory map chapter). Some UTEST DCF records are programmed at the factory. The user may add additional USER DCF records beginning with the first unprogrammed location following the factory programmed UTEST DCF records. An unprogrammed location in flash memory contains 0xFFFF_FFFF.

When programming UTEST DCF records at the factory, the records must start at the first address in the UTEST flash memory area and be continuous; that is, one UTEST DCF record must immediately follow the previous UTEST DCF record. An unprogrammed location in the UTEST flash memory area is interpreted by the SSCM as the end of the UTEST DCF records. When this happens, the SSCM passes control of the boot sequence to the Reset Generation Module.

9.4 DCF client table

A list of DCF clients is shown in *Table 138*. Various attributes of the DCF clients are listed in these tables. The following list explains the meaning of the columns and give a description of the attributes of these DCF clients:

- DCF Chip Select

  The CS field is a 15-bit field with each bit in the field assigned to a particular module. These bits act as module select signals for the internal modules. Only one bit in this field should be set in a DCF record.
- DCF address

  The ADDR field is a 15-bit field that specifies an internal address for a DCF client. These addresses are used only by the SSCM when interpreting DCF records and writing the data to specified modules.
- DCF Client Description

  This field contains the name of the DCF clients.
- Reset value of DCF client

  This field contains the reset value of DCF client before being written by the SSCM with the associated DCF record.
- Typical value of DCF client

  This field contains a typical value that you might program into the various DCF clients. This field is for information only.
- IPS read

  This column specifies whether or not a DCF client can be read by user software after the release of reset.
- DCF client special strategy

  Though all DCF records have the same format, they can be used in different ways. As the device is being designed, registers must be designated as Device Configuration Format clients if they need to be written with a DCF record. There are other features that must be designated for DCF clients. For instance, DCF clients can have several different write strategies: Parity, Write Once, Triple Voted + Parity + Check with second write, Triple voted + Check with second write, or no strategy at all. The SSCM examines the DCF client to determine the correct special strategy to use and then

appropriately write a DCF record to that client. The following list defines the special strategies:

– None
No special DCF strategy is used.

– Parity
Not implemented for DCF clients. Only used for TEST only DCF clients not accessible by the user.

– Write Once
A register using the Write Once strategy can only be written once. The DCF client ignores subsequent writes.

– Triple Voted
DCF clients that use the Triple Voted strategy have three copies of the register. The SSCM writes to all three registers in a single write cycle. The outputs of the three registers are majority voted together to determine the correct data value. Triple voting allows for a 'bit-flip' error to occur without changing the DCF client output data.

– Triple Voted with second write
DCF clients that use the Triple Voted with 2nd write strategy have three copies of the register. The SSCM writes to all three registers in a single write cycle. The outputs of the three registers are majority voted together to determine the correct data value. During the second execution of Phase 3 of the reset sequence, the SSCM attempts to write the DCF client again. At this time, the DCF client checks to see that the register contains the same data that is being written again.

– Write 0 only
A bit in a DCF client can only be written from a logic 1 to a logic 0. An attempt to write a bit with this attribute to a logic 1 is ignored.

– Write 1 only
A bit in a DCF client can only be written from a logic 0 to a logic 1. An attempt to write a bit with this attribute to a logic 0 is ignored.

– DCF order dependency in flash memory
This column details whether or not there are special conditions on the location of a particular DCF record in flash memory.

Table 138. DCF client list(1)

DCF CS [14:0]	DCF address [9:2] (binary)	DCF client			IP read direct access	Special strategy		DCF order dependencyi in flash memory	Estimated UTEST flash memory space (bytes)
		Description	Value			DCF client	Flash		
			Reset	Typical					
STCU - Support for 4 LBIST partitions (NLBIST); 22 Memory Cuts (NMCUT); 32-bit PRPG; 64-bit MISR									
000_0000_0000_0100	00000010	STCU_SKC	—	—	yes	None	None	First record among STCU, Key1/Key2 DCF record sequence; Write Key2 every (1024-8) clock edges	16
000_0000_0000_0100	00000000	STCU_RUN	—	—	yes	None	None	None	8
000_0000_0000_0100	00000011	STCU_CFG	—	—	yes	None	None	None	8
000_0000_0000_0100	00000100	STCU_PLL_CFG	—	—	yes	None	None	None	8
000_0000_0000_0100	00000101	STCU_WDG	—	—	yes	None	None	None	8
000_0000_0000_0100	00000111	STCU_CRCE	—	—	yes	None	None	None	8
000_0000_0000_0100	00001010	STCU_ERR_FM	—	—	yes	None	None	None	8
000_0000_0000_0100	00001111	STCU_LBRMSW	—	—	yes	None	None	None	8
000_0000_0000_0100	00010000	STCU_LBUFM	—	—	yes	None	None	None	8
000_0000_0000_0100	00011101	STCU_MBUFML [31:0]	—	—	yes	None	None	None	8
000_0000_0000_0100	01000000	STCU_LB_CTRL_0	—	—	yes	None	None	None	8
000_0000_0000_0100	01000001	STCU_LB_PCS_0	—	—	yes	None	None	None	8
000_0000_0000_0100	01000010	STCU_LB_PRPGL_0	—	—	yes	None	None	None	8
000_0000_0000_0100	01000100	STCU_LB_MISREL_0	—	—	yes	None	None	None	8
000_0000_0000_0100	01000101	STCU_LB_MISREH_0	—	—	yes	None	None	None	8

Table 138. DCF client list[1] (continued)

DCF CS [14:0]	DCF address [9:2] (binary)	DCF client			IP read direct access	Special strategy		DCF order dependencyi in flash memory	Estimated UTEST flash memory space (bytes)
		Description	Value			DCF client	Flash		
			Reset	Typical					
000_0000_0000_0100	01001000	STCU_LB_MISRELSW_0	—	—	yes	None	None	None	8
000_0000_0000_0100	01001001	STCU_LB_MISREHSW_0	—	—	yes	None	None	None	8
000_0000_0000_0100	01010000	STCU_LB_CTRL_1	—	—	yes	None	None	None	8
000_0000_0000_0100	01010001	STCU_LB_PCS_1	—	—	yes	None	None	None	8
000_0000_0000_0100	01010010	STCU_LB_PRPGL_1	—	—	yes	None	None	None	8
000_0000_0000_0100	01010100	STCU_LB_MISREL_1	—	—	yes	None	None	None	8
000_0000_0000_0100	01010101	STCU_LB_MISREH_1	—	—	yes	None	None	None	8
000_0000_0000_0100	01011000	STCU_LB_MISRELSW_1	—	—	yes	None	None	None	8
000_0000_0000_0100	01011001	STCU_LB_MISREHSW_1	—	—	yes	None	None	None	8
000_0000_0000_0100	01100000	STCU_LB_CTRL_2	—	—	yes	None	None	None	8
000_0000_0000_0100	01100001	STCU_LB_PCS_2	—	—	yes	None	None	None	8
000_0000_0000_0100	01100010	STCU_LB_PRPGL_2	—	—	yes	None	None	None	8
000_0000_0000_0100	01100100	STCU_LB_MISREL_2	—	—	yes	None	None	None	8
000_0000_0000_0100	01100101	STCU_LB_MISREH_2	—	—	yes	None	None	None	8
000_0000_0000_0100	01101000	STCU_LB_MISRELSW_2	—	—	yes	None	None	None	8
000_0000_0000_0100	01101001	STCU_LB_MISREHSW_2	—	—	yes	None	None	None	8
000_0000_0000_0100	01110000	STCU_LB_CTRL_3	—	—	yes	None	None	None	8
000_0000_0000_0100	01110001	STCU_LB_PCS_3	—	—	yes	None	None	None	8
000_0000_0000_0100	01110010	STCU_LB_PRPGL_3	—	—	yes	None	None	None	8

Table 138. DCF client list[1] (continued)

DCF CS [14:0]	DCF address [9:2] (binary)	DCF client			IP read direct access	Special strategy		DCF order dependencyi in flash memory	Estimated UTEST flash memory space (bytes)
		Description	Value			DCF client	Flash		
			Reset	Typical					
000_0000_0000_0100	01110100	STCU_LB_MISREL_3	—	—	yes	None	None	None	8
000_0000_0000_0100	01110101	STCU_LB_MISREH_3	—	—	yes	None	None	None	8
000_0000_0000_0100	01111000	STCU_LB_MISRELSW_3	—	—	yes	None	None	None	8
000_0000_0000_0100	01111001	STCU_LB_MISREHSW_3	—	—	yes	None	None	None	8
000_0000_0000_0100	10000000	STCU_MB_CTRL_0	—	—	yes	None	None	None	8
000_0000_0000_0100	10000001	STCU_MB_CTRL_1	—	—	yes	None	None	None	8
000_0000_0000_0100	10000010	STCU_MB_CTRL_2	—	—	yes	None	None	None	8
000_0000_0000_0100	10000011	STCU_MB_CTRL_3	—	—	yes	None	None	None	8
000_0000_0000_0100	10000100	STCU_MB_CTRL_4	—	—	yes	None	None	None	8
000_0000_0000_0100	10000101	STCU_MB_CTRL_5	—	—	yes	None	None	None	8
000_0000_0000_0100	10000110	STCU_MB_CTRL_6	—	—	yes	None	None	None	8
000_0000_0000_0100	10000111	STCU_MB_CTRL_7	—	—	yes	None	None	None	8
000_0000_0000_0100	10001000	STCU_MB_CTRL_8	—	—	yes	None	None	None	8
000_0000_0000_0100	10001001	STCU_MB_CTRL_9	—	—	yes	None	None	None	8
000_0000_0000_0100	10001010	STCU_MB_CTRL_10	—	—	yes	None	None	None	8
000_0000_0000_0100	10001011	STCU_MB_CTRL_11	—	—	yes	None	None	None	8
000_0000_0000_0100	10001100	STCU_MB_CTRL_12	—	—	yes	None	None	None	8
000_0000_0000_0100	10001101	STCU_MB_CTRL_13	—	—	yes	None	None	None	8
000_0000_0000_0100	10001110	STCU_MB_CTRL_14	—	—	yes	None	None	None	8

Table 138. DCF client list[1] (continued)

DCF CS [14:0]	DCF address [9:2] (binary)	DCF client			IP read direct access	Special strategy		DCF order dependencyi in flash memory	Estimated UTEST flash memory space (bytes)
		Description	Value			DCF client	Flash		
			Reset	Typical					
000_0000_0000_0100	10001111	STCU_MB_CTRL_15	—	—	yes	None	None	None	8
000_0000_0000_0100	10010000	STCU_MB_CTRL_16	—	—	yes	None	None	None	8
000_0000_0000_0100	10010001	STCU_MB_CTRL_17	—	—	yes	None	None	None	8
000_0000_0000_0100	10010010	STCU_MB_CTRL_18	—	—	yes	None	None	None	8
000_0000_0000_0100	10010011	STCU_MB_CTRL_19	—	—	yes	None	None	None	8
000_0000_0000_0100	10010100	STCU_MB_CTRL_20	—	—	yes	None	None	None	8
000_0000_0000_0100	10010101	STCU_MB_CTRL_21	—	—	yes	None	None	None	8
PASS									
000_0000_0000_1000	00101001 – 00101011	Reserved							
000_0000_0000_1000	00101100	Censorship	0x0000	0x55AA	yes	None	None	None	8
000_0000_0000_1000	00101101 – 00101111	Reserved							
000_0000_0000_1000	00110000	Reserved							
000_0000_0000_1000	00110001 – 00111111	Reserved							
000_0000_0000_1000	01000000	LOCK0_PG0	0xFFF_FFFF	—	no	None	None	None	8
000_0000_0000_1000	01000001	LOCK1_PG0	0xFFF_FFFF	—	no	None	None	None	8
000_0000_0000_1000	01000010	LOCK2_PG0	0xFFF_FFFF	—	no	None	None	None	8
000_0000_0000_1000	01000011	LOCK3_PG0	0xFFF_FFFF	—	no	None	None	None	8
000_0000_0000_1000	01000100	LOCK0_PG1	0xFFF_FFFF	—	no	None	None	None	8

Table 138. DCF client list[1] (continued)

DCF CS [14:0]	DCF address [9:2] (binary)	DCF client			IP read direct access	Special strategy		DCF order dependencyi in flash memory	Estimated UTEST flash memory space (bytes)
		Description	Value			DCF client	Flash		
			Reset	Typical					
000_0000_0000_1000	01000101	LOCK1_PG1	0xFFF_FFFF	—	no	None	None	None	8
000_0000_0000_1000	01000110	LOCK2_PG1	0xFFF_FFFF	—	no	None	None	None	8
000_0000_0000_1000	01000111	LOCK3_PG1	0xFFF_FFFF	—	no	None	None	None	8
000_0000_0000_1000	01001000	LOCK0_PG2	0xFFF_FFFF	—	no	None	None	None	8
000_0000_0000_1000	01001001	LOCK1_PG2	0xFFF_FFFF	—	no	None	None	None	8
000_0000_0000_1000	01001010	LOCK2_PG2	0xFFF_FFFF	—	no	None	None	None	8
000_0000_0000_1000	01001011	LOCK3_PG2	0xFFF_FFFF	—	no	None	None	None	8
000_0000_0000_1000	01001100	LOCK0_PG3	0xFFF_FFFF	—	no	None	None	None	8
000_0000_0000_1000	01001101	LOCK1_PG3	0xFFF_FFFF	—	no	None	None	None	8
000_0000_0000_1000	01001110	LOCK2_PG3	0xFFF_FFFF	—	no	None	None	None	8
000_0000_0000_1000	01001111	LOCK3_PG3	0xFFF_FFFF	—	no	None	None	None	8
000_0000_0000_1000	01001100 – 11111111	Reserved							
Miscellaneous									
000_0000_1000_0000	00000000	UTEST Miscellaneous (See *Figure 52* and *Table 139* for bit position details)	0x00000140	—	no	Triple voted	None	None	8
000_0000_1000_0000	000000000000001 – 00011110	Reserved							

Table 138. DCF client list[1] (continued)

DCF CS [14:0]	DCF address [9:2] (binary)	DCF client			IP read direct access	Special strategy		DCF order dependencyi in flash memory	Estimated UTEST flash memory space (bytes)
		Description	Value			DCF client	Flash		
			Reset	Typical					
OTP Glue									
000_0000_0001_0000	00000000 – 00000111	Reserved							
000_0000_0001_0000	00001000	OTP_EN0	0x0000 0000	—	no	Write once	None	None	8
000_0000_0001_0000	00001001	OTP_EN1	0x0000 0000	—	no	Write once	None	None	8
000_0000_0001_0000	00001010	OTP_EN2	0x0000 0000	—	no	Write once	None	None	8
000_0000_0001_0000	00001011	OTP_EN3	0x0000 0000	—	no	Write once	None	None	8
000_0000_0001_0000	00001100 – 11111111	Reserved							
PMC_DIGITAL[2]									
000_0000_1000_0000	00011111	REE_BUS (Refer to *Figure 53* and *Table 140*)	0x0000 0000	—	yes	Triple Voted	None	None	8
000_0000_1000_0000	00100000 – 00111000	Reserved							
000_0000_1000_0000	00111001	PNREG_ENB[3]	0x0000 0000	—	yes	Triple Voted	None	1	None
000_0000_1000_0000	00111010 – 01111101	Reserved							
000_0000_1000_0000	01110000 – 01111101	Reserved							
000_0000_1000_0000	01111110	ESR0_CFG[4]	0x0000 0000	—	yes	Triple Voted +chk_wr2	None	None	8
000_0000_1000_0000	01111110 – 11111111	Reserved							

1. DCF programming should start from address 0x00400208.
2. See *Chapter 50: Power management controller digital interface (PMC_Dig)*.
3. See *Section 50.5.31: IPS Regulator Monitor Register (IREGU_MON)*.
4. See *Section 50.5.32: ESR0 ConFiGuration Register (ESR0_CFG)*.

Offset: 000000000000000 Access: DCF Client

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

Reset The reset value is described in *Table 138: DCF client list*.

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0			0	0		0									0
W		XOSC_ALC_DIS	XOSC_LF_EN			XOSC_EN_40MHZ		LOCKSTEP_EN	XOSC_EN	XOSC_EXT_CLOAD	XOSC_LOAD_CAP_SEL					

Reset The reset value is described in *Table 138: DCF client list*.

Figure 52. UTEST Miscellaneous DCF Client

Table 139. UTEST Miscellaneous DCF Client field descriptions

Field	Description
17 XOSC_ALC_DIS	XOSC ALC_DIS 0 Enable Automatic Level Controller 1 Disable Automatic Level Controller
18 XOSC_LF_EN	XOSC LF_EN 0 Oscillator is not used for low frequency (~8 MHz) 1 Oscillator is used for low frequency (~8 MHz)
21 XOSC_EN_40MHZ	XOSC EN_40MHz **Note:** This field must be configured according to the external cristal used. 0 20 MHz 1 40 MHz XOSC
24 XOSC_EN	Enable XOSC 0 XOSC disabled 1 XOSC enabled
25 XOSC_EXT_CLOAD	XOSC EXT CLOAD 0 Selects XOSC internal capacitor (20–40 MHz operation) 1 External (required for 4–16 MHz operation)
26:30 XOSC_LOAD_CAP_SEL	Five trim bits to program the load capacitance. Load capacitor values are determined by the crystal manufacturer datasheet. **Note:** Please refer to the SPC574Sx Datasheet for the value of the trimming bits.

Offset: 000000000011111 Access: DCF Client

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0													
W				LVD14_ADL1	LVD14_ADL0	LVD14_IOL1	LVD14_IOL0	HVD13_C	LVD11_ADL1	LVD11_ADL0	LVD11_FL	LVD11_IOL1	LVD11_OSC	LVD11_C	HVD_6C	LVD_3C
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 53. REE_BUS bits DCF Client

Note: LVD11_C, LVD11_FL & LVD3_C monitors once enabled in DCF cannot be cleared by DCF record.

Table 140. REE_BUS bits DCF client field descriptions

Field	Description
19 LVD14_ADL1	LVD14_ADL1 reset enable This bit defines whether an LVD assertion on the first supply of the ADC1 generates a system reset. The RES_HV1[RES14_ADL1] bit determines whether the reset is functional or destructive. 0 Disabled. LVD assertion on the first supply of the ADC1 does not cause system reset 1 Enabled. LVD assertion on the first supply of the ADC1 causes system reset
20 LVD14_ADL0	LVD14_ADL0 reset enable This bit defines whether an LVD assertion on the first supply of the ADC0 generates a system reset. The RES_HV1[RES14_ADL0] bit determines whether the reset is functional or destructive. 0 Disabled. LVD assertion on the first supply of the ADC0 does not cause system reset 1 Enabled. LVD assertion on the first supply of the ADC0 causes system reset
21 LVD14_IOL1	LVD14_IOL1 reset enable This bit defines whether an LVD assertion on the first supply of the IOL1 generates a system reset. The RES_HV1[RES14_IOL1] bit determines whether the reset is functional or destructive. 0 Disabled. LVD assertion on the first supply of the IOL1 does not cause system reset 1 Enabled. LVD assertion on the first supply of the IOL1 causes system reset

Table 140. REE_BUS bits DCF client field descriptions (continued)

Field	Description
22 LVD14_IOL0	LVD14_IOL0 reset enable This bit defines whether an LVD assertion on the first supply of the IOL0 generates a system reset. The RES_HV1[RES14_IOL0] bit determines whether the reset is functional or destructive. 0 Disabled. LVD assertion on the first supply of the IOL0 does not cause system reset 1 Enabled. LVD assertion on the first supply of the IOL0 causes system reset
23 HVD13_C	HVD13_C reset enable This bit defines whether an HVD assertion on the IOL0 supply generates a system reset. The RES_HV1[RES13_C] bit determines whether the reset is functional or destructive. 0 Disabled. HVD assertion on the IOL0 supply does not cause system reset 1 Enabled. HVD assertion on the IOL0 supply causes system reset
24 LVD11_ADL1	LVD11_ADL1 reset enable This bit defines whether an LVD assertion on the first supply of the ADC1 generates a system reset. The RES_HV0[RES11_ADL1] bit determines whether the reset is functional or destructive. 0 Disabled. LVD assertion on the first supply of the ADC1 does not cause system reset 1 Enabled. LVD assertion on the first supply of the ADC1 causes system reset
25 LVD11_ADL0	LVD11_ADL0 reset enable This bit defines whether an LVD assertion on the first supply of the ADC0 generates a system reset. The RES_HV0[RES11_ADL0] bit determines whether the reset is functional or destructive. 0 Disabled. LVD assertion on the first supply of the ADC0 does not cause system reset 1 Enabled. LVD assertion on the first supply of the ADC0 causes system reset
26 LVD11_FL	LVD11_FL reset enable This bit defines whether an LVD assertion on the supply of the flash generates a system reset. The RES_HV0[RES11_FL] bit determines whether the reset is functional or destructive. 0 Disabled. LVD assertion on the supply of the flash does not cause system reset 1 Enabled. LVD assertion on the supply of the flash causes system reset
27 LVD11_IOL1	LVD11_IOL1 reset enable This bit defines whether an LVD assertion on the supply of the IOL1 generates a system reset. The RES_HV0[RES11_IOL1] bit determines whether the reset is functional or destructive. 0 Disabled. LVD assertion on the supply of the IOL1 does not cause system reset 1 Enabled. LVD assertion on the supply of the IOL1 causes system reset
28 LVD11_OSC	LVD11_OSC reset enable This bit defines whether an LVD assertion on the supply of the XOSC generates a system reset. The RES_HV0[RES11_XOSC] bit determines whether the reset is functional or destructive. 0 Disabled. LVD assertion on the supply of the XOSC does not cause system reset 1 Enabled. LVD assertion on the supply of the XOSC causes system reset

Table 140. REE_BUS bits DCF client field descriptions (continued)

Field	Description
29 LVD11_C	This bit defines whether an LVD assertion on the IOL0 supply generates a system reset. The RES_HV0[RES11_C] bit determines whether the reset is functional or destructive. 0 Disabled. LVD assertion on the IOL0 supply does not cause system reset 1 Enabled. LVD assertion on the IOL0 supply causes system reset
30 HVD_6C	HVD_6C reset enable This bit defines whether an HVD assertion on the core supply generates a system reset. The RES_LV1[RES6_C] bit determines whether the reset is functional or destructive. 0 Disabled. HVD assertion on the core supply does not cause system reset 1 Enabled. HVD assertion on the core supply causes system reset
31 LVD_3C	LVD_3C reset enable This bit defines whether an LVD assertion on the core supply generates a system reset. The RES_LV0[RES3_C] bit determines whether the reset is functional or destructive. 0 Disabled. LVD assertion on the core supply does not cause system reset 1 Enabled. LVD assertion on the core supply causes system reset

10 Power management

10.1 Overview

SPC574Sx microcontrollers include a robust power management infrastructure for applications to select various Operational and Low-power modes and monitor internal voltages for High- and Low-voltage conditions. This monitoring is also used to ensure supply voltages and internal voltages are within the required ranges before the microcontroller leaves reset.

This chapter gives an overview of the built-in power management features.

The power management infrastructure encompasses the following modules:

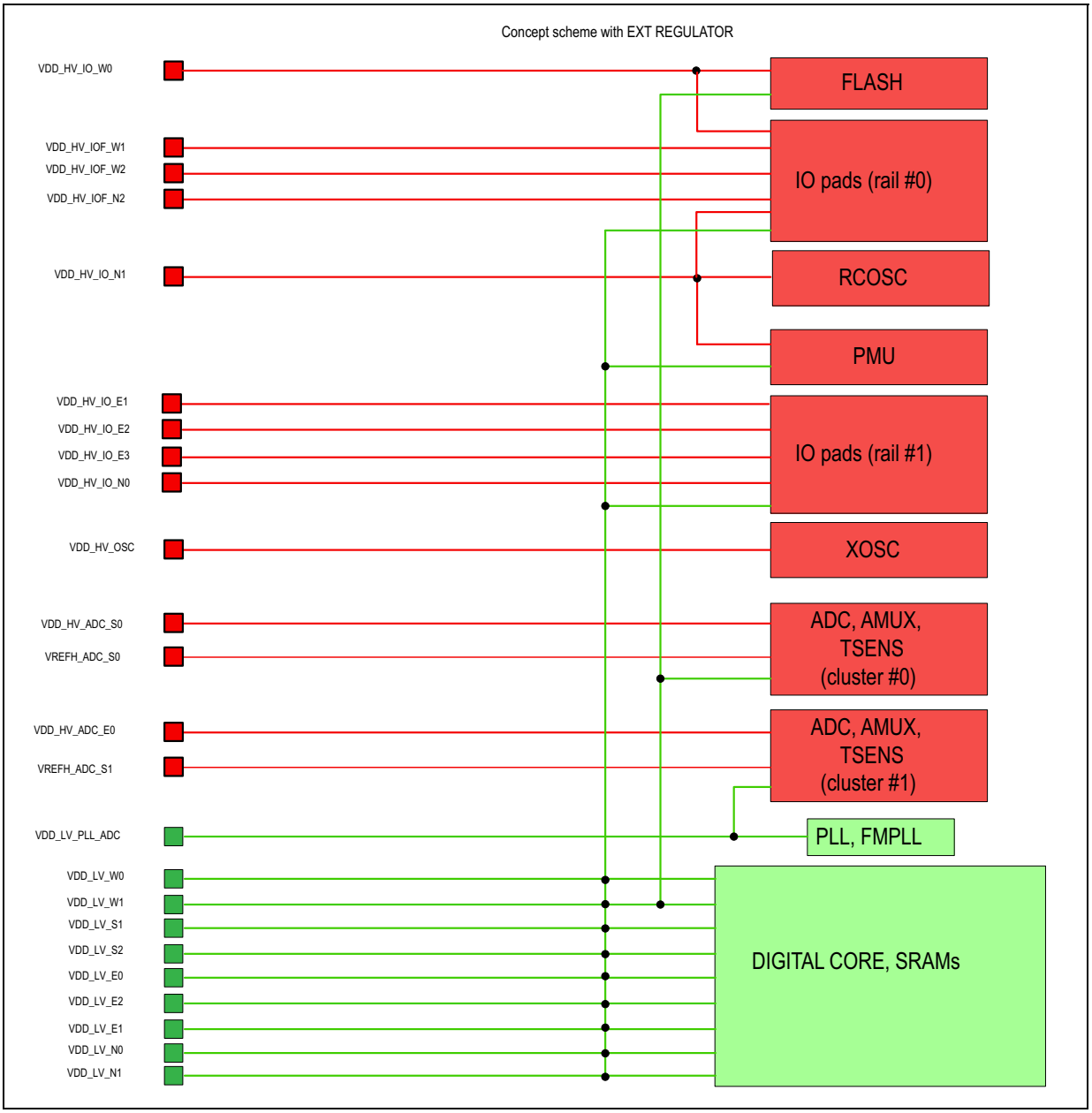
- Power Control Unit (MC_PCU): provides a status register, and the Power Management Control Digital Interface (PMC_Dig) submodule, which provides an interface to configure and control internal voltages
- Clock Generation Module (MC_CGM): generates reference clocks for all the chip blocks. It works in conjunction with internal clock gating logic to implement Low-power modes
- Mode Entry Module (MC_ME): controls chip operational modes and mode transition sequences. It also contains configuration, control and status registers accessible for the application
  - System modes RESET, SAFE, TEST and DRUN are entered automatically on system events or on application request
  - User modes RUN*x*, HALT, and STOP control performance and consumption during normal application operation. The modes are organized according to the performance provided. Each mode is entered via a dedicated key-protected register to avoid unwanted transition

See the respective module chapters for further details.

10.1.1 Power management framework

The power management framework supports a variety of configuration options (see *Figure 54*). SPC574Sx implements a unique core voltage power domain. There are no separate functional power domains.

Figure 54. Concept scheme of the power connections



10.1.2 Power management supply description

Table 141 provides an overview of the PMC supply signals.

Table 141. Device power management controller external signals

Supply	Pad	Description
VSS_HV	Exposed pad	Common High voltage ground
VSS_LV	Exposed pad	Common Low voltage ground

Table 141. Device power management controller external signals (continued)

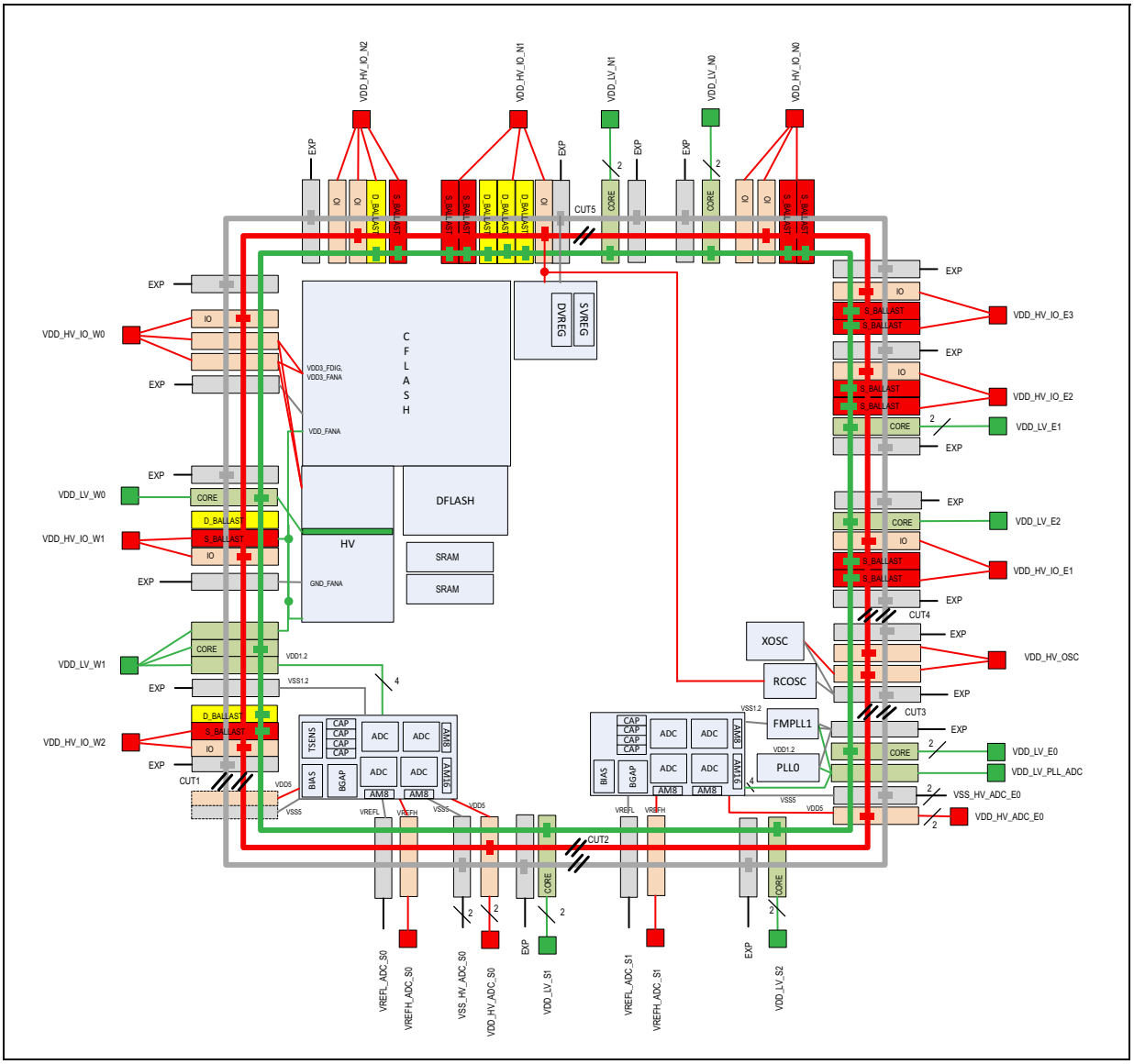
Supply	Pad	Description
VSS_HV_OSC_PMC	Exposed pad	Ground supply for the internal power management unit and the RCOSC
VDD_LV	VDD_LV_PLL_ADC VDD_LV_W0 VDD_LV_W1 VDD_LV_S1 VDD_LV_S2 VDD_LV_E0 VDD_LV_E2 VDD_LV_E1 VDD_LV_N0 VDD_LV_N1	Low voltage power supply for device
VDD_HV_IO0	VDD_HV_IO_W0 VDD_HV_IO_W1 VDD_HV_IO_W2 VDD_HV_IO_N2 VDD_HV_IO_N1	High voltage power supply for I/O rail #0, FLASH, Power Management Unit and RCOSC
VDD_HV_IO1	VDD_HV_IO_E1 VDD_HV_IO_E2 VDD_HV_IO_E3 VDD_HV_IO_N0	High voltage power supply for I/O rail #1
VDD_HV_OSC	VDD_HV_OSC	High voltage power supply for XOSC
VSS_HV_ADC_TSENS	Exposed pad	Ground supply for ADC and TSENS
VDD_HV_ADC_TSENS_0	VDD_HV_ADC_S0	High voltage supply for ADC group #0 and TSENS 0
VDD_HV_ADC_TSENS_1	VDD_HV_ADC_E0	High voltage supply for ADC group #1 and TSENS 1
VREFL_ADC	Exposed pad	Ground reference of ADCs
VREFH_ADC_0	VREFH_ADC_S0	Voltage reference of ADC group #0
VREFH_ADC_1	VREFH_ADC_S1	Voltage reference of ADC group #1

Warning: There are constraints on some input voltages relative to other input voltages. Please refer to the device data sheet for detailed information.

10.2 Architecture

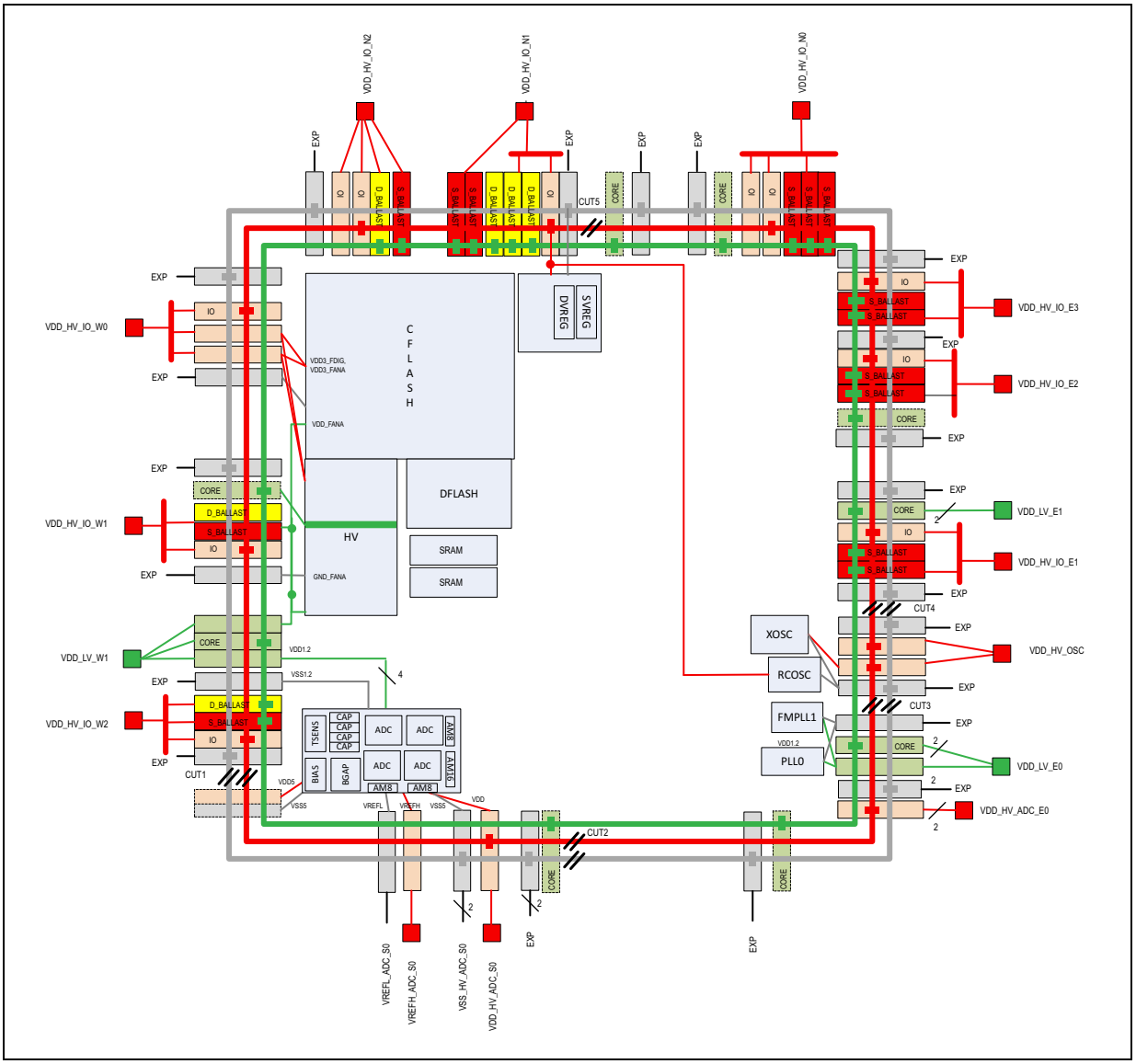
Figure 55 and *Figure 56* show the general power supply scheme for the BGA244/QFP144 package and QFP100 package respectively.

Figure 55. Power supply scheme BGA244 and QFP144(1)



1. GND is connected to GROUND BALLS for the BGA244.
   GND is connected to EXPOSED PAD for the TQFP144.

Figure 56. Power supply scheme QFP100



The HV ring is divided into 5 different physical domains:

- ADC group #0 and Temperature Sensor
- ADC group #1
- IO rail #0, Flash, PMU and RCOSC
- IO rail #1
- XOSC

From user point of view:

- 4 HV supplies are to be considered if the internal LV regulator is not used :
  - ADC group #0
  - ADC group #1
  - IO rail #0 + XOSC + PMU + Flash
  - IO rail #1
- 3 HV supplies are to be considered if the internal LV regulator is used (mandatory for the QFP100 package):
  - ADC group #0
  - ADC group #1[f]
  - IO rail #0 + XOSC + PMU + Flash + IO rail #1

The LV voltage is organized as one single domain. The LV can be generated internally (with internal ballast) or externally, depending on a pin setting. The exposed pad is used to connect the ground ring.

Figure 54 shows the concept scheme of the power connection.

10.2.1 Start-up sequence

There is no constraint on the power startup sequence for LV/HV.

10.2.2 Voltage regulator mode

The voltage regulator can be statically (bonding option) configured as internal or external.

A pin (PWMODE) determines the voltage regulator:

- if connected to GND, then external voltage regulator is used to generate the low voltage
- if connected to VDD, then internal voltage regulator is used to generate the low voltage

10.2.3 Ballast

Two types of ballast are included:

- S_BALLAST, which is a static (or digital) ballast
- D_BALLAST, which is a dynamic (or linear) ballast

The ballasts are circuits placed in the padring, similarly as other pad circuits (supply or input/output functions).

The S_BALLASTs are controlled by the digital regulator to source the static current. Total current capabily for the S_BALLAST on the LV domain is 325mA.

The D_BALLASTs are controlled by the linear regulator to source the dynamic current to support the the transient load response. Total current capability for all the D_BALLAST on the LV domain is 100mA.

Transient time for the dynamic regulation is 20 us for maximum 100mA on the LV.

The S_BALLASTs are enabled only in INTERNAL regulator mode (PWMODE = 1).

f. In case of TQFP100, ADC 1 is not available (refer to *Figure 56: Power supply scheme QFP100*)

The D_BALLASTs are always enabled in INTERNAL regulator mode (PWMODE = 1).

The D_BALLASTs are enabled in EXTERNAL regulator mode (PWMODE = 0) only if the LV supply is out of specs (+/- 5%) as tentative to avoid the device reset.

10.2.4 Reduced power modes

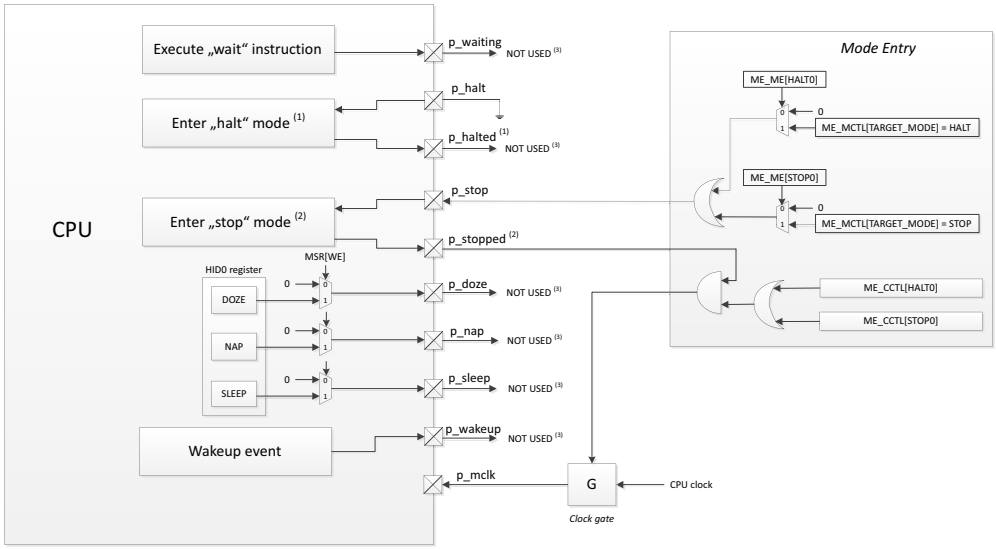
Table 142 shows which reduced power modes are available (as part of the Magic Carpet mode selection) and the corresponding effects.

Table 142. Reduced power mode summary

Mode	Enabled by	Effect
Wait	“wait” instruction execution	No effect
HALT	MC_ME mode change	– CPU enters in STOP mode – RCOSC enabled – PLL0, PLL1,XOSC clock on/off status configurable via ME_HALT0_MC register – Platform clock is stopped
STOP	MC_ME mode change	– CPU enters in STOP mode – RCOSC enabled – PLL0 and PLL1 stopped – XOSC clock on/off status configurable via ME_STOP0_MC register – System clock is forced to the internal RCOSC – Platform clock is stopped – I/O pin power can be disabled via ME_STOP0_MC register
NAP	CPU HID0[NAP] bit writing	No effect
DOZE	CPU HID0[DOZE] bit writing	No effect
SLEEP	CPU HID0[SLEEP] bit writing	No effect

Figure 57 gives a representation of the internal signals regulating the reduced power modes.

Figure 57. Reduced power modes



Note 1: HALT mode => RCOSC enabled; PLL0, PLL1, XOSC clock on/off status configurable via ME_HALT0_MC register
Note 2: STOP mode => RCOSC enabled; PLL0, PLL1 stopped; XOSC clock on/off status configurable via ME_STOP0_MC register; I/O pin power can be disabled via ME_STOP0_MC register; system clock is forced to the internal RCOSC
Note 3: Not used at SOC level but still used within the lockstep core signal comparison

10.2.5 Voltage monitors

The PMU block contains a set of monitors aimed to check the voltage and temperature operating conditions.

10.2.5.1 Nomenclature

- **POR** stands for Power On Reset. It is a circuit that manages the reset from very low voltage up to its threshold. Cannot be disabled
- **MVD** stands for Minimum Voltage Detector. It cannot be disabled by the user and generates a destructive Reset
- **LVD** stands for Low Voltage Detector. It can be disabled by the user
- **HVD** stands for High Voltage Detector. It can be disabled by the user
- **UVD** stands for Upper Voltage Detector. It cannot be disabled by the user and generates a destructive reset

Figure 58 shows the stack of coverage for the voltage monitor.

The operating ranges of the LV/HV for any static/dynamic/PVT/load conditions should be the following:

1. LV = 1.255V +/- 5%
2. HV = 4.5V to 5.5V or 3.0V to 3.6V

30mV of overshoot on the LV is allowed on the top of 1.255V + 5% in the transient time required to stabilize the supply. In case of violation of the allowed overshoot the HVD140_C and UVD145_C detectors can be triggered.

The HVD140_C reaction in case of failure is user defined.

The UVD145_C reaction in case of failure is the destructive reset.

In terms of transient load response, the recommendation is to support 100mA in 20 µs on the LV (as done for the INTERNAL regulator).

On the HV the window is +/- 10%, therefore eventual overshoot must be enclosed in 500mV.

5.5V is the HV limit to operate without reliability degradation.

Figure 58. Voltage monitor coverage stack

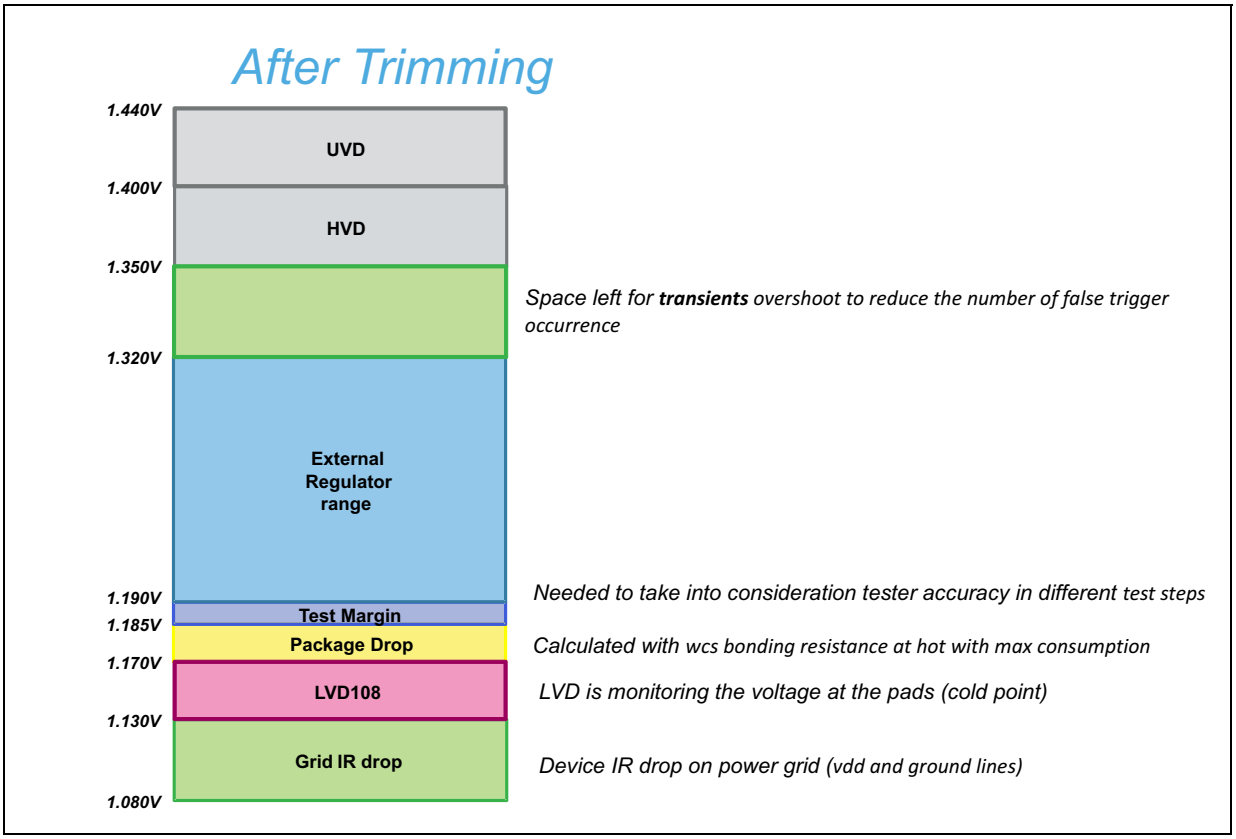
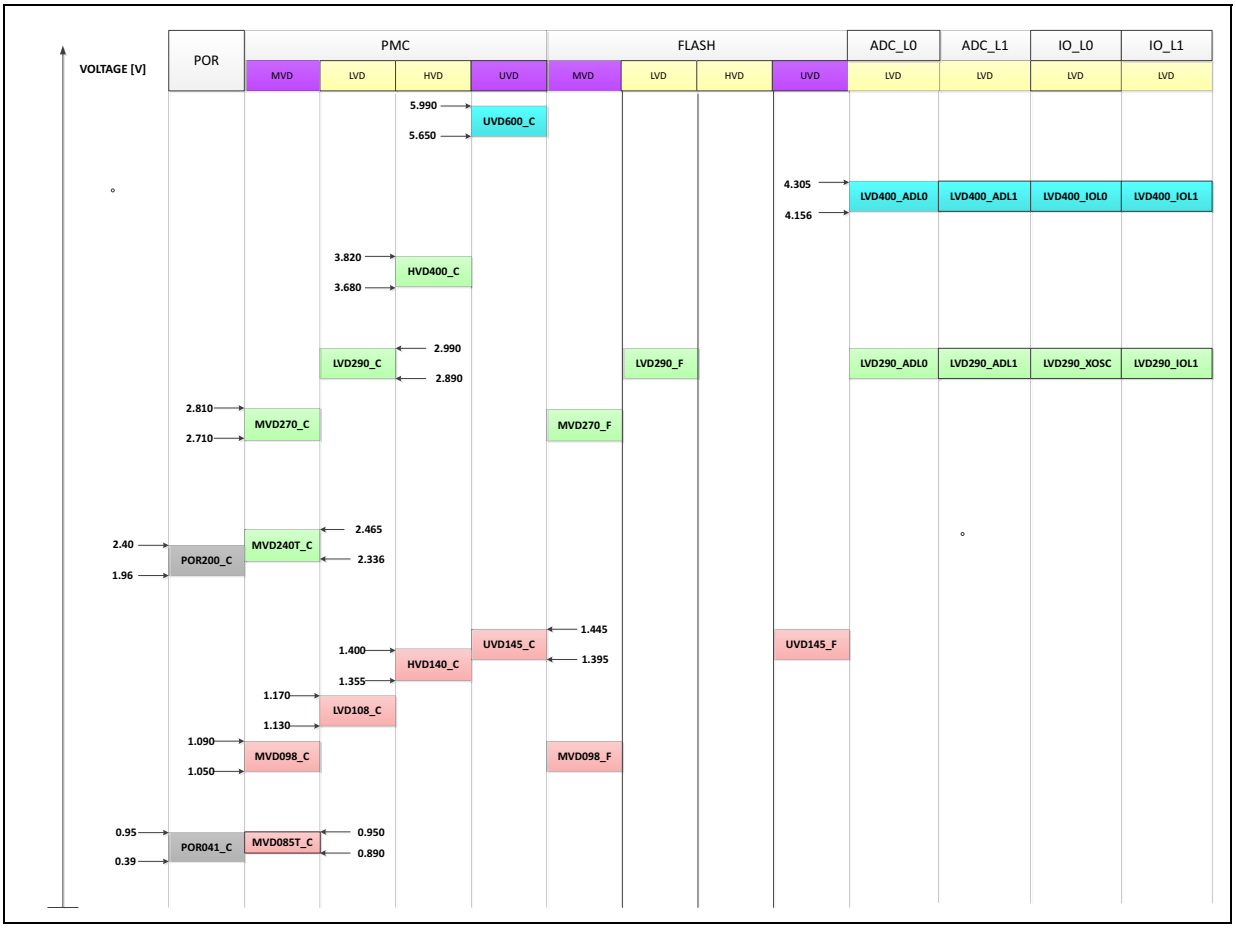


Figure 59 and *Figure 62* show the voltage monitor detailed coverage and functionality.

Figure 59. Voltage monitor coverage



Depending on the chosen power supply for the high voltage, some voltage detectors for the high voltage should be disabled by the user. *Table 143* shows which voltage monitors shall be used according to the high voltage value.

Table 143. Voltage Monitors for the High Voltage

HV = 3.3V	HV = 5V
LVD290_ADL0(1)	LVD400_ADL0(1)
LVD290_ADL1(1)	LVD400_ADL1(1)
—	LVD400_IOL0(2)
LVD290_IOL1(2)	LVD400_IOL1(2)
HVD400_C	—

1. L0 and L1 are equivalent to group #0 and group #1 respectively for the ADC
2. L0 and L1 are equivalent to rail #0 and rail #1 respectively for the IO

10.2.5.2 Voltage monitor reactions

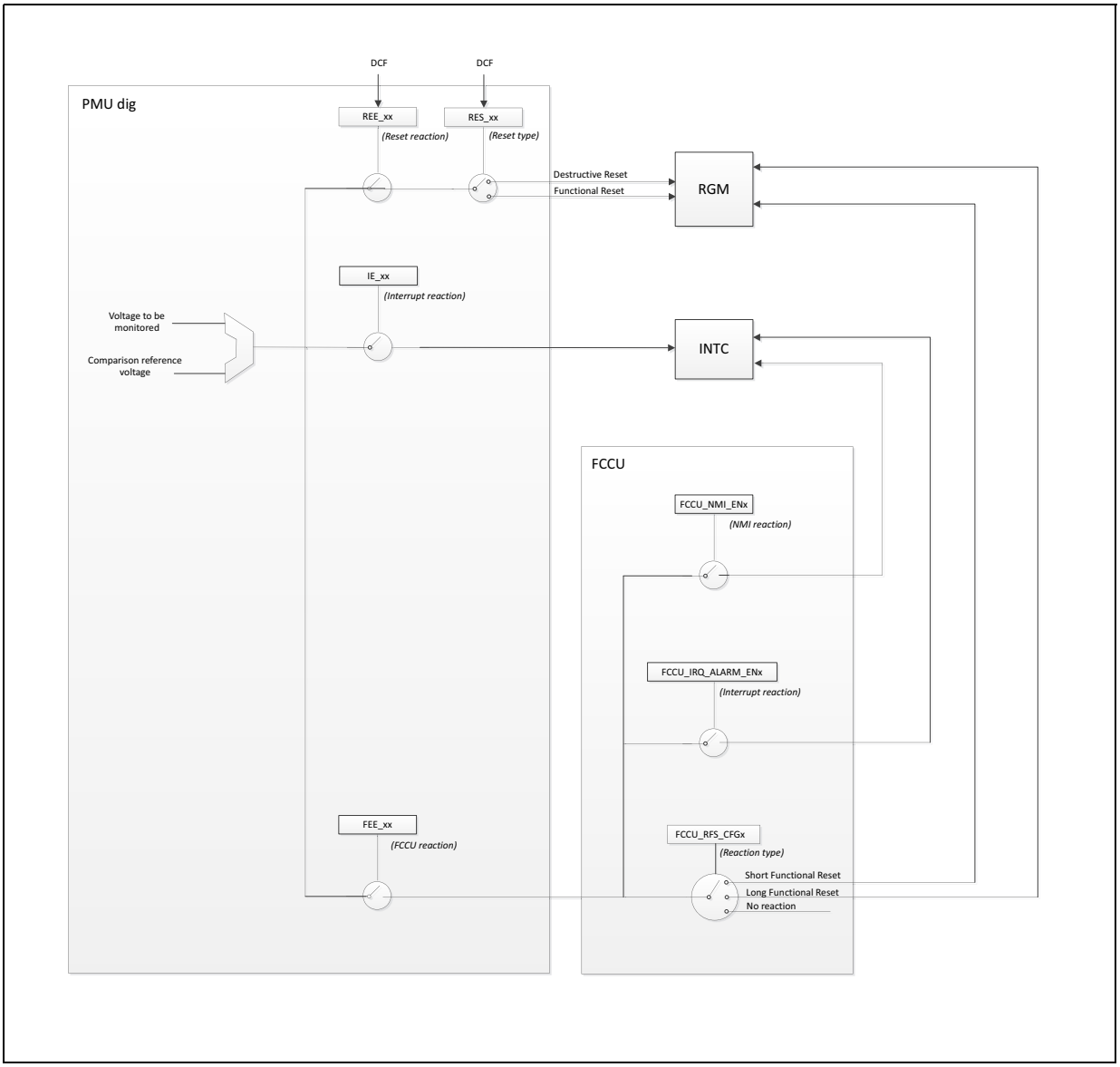
Figure 60 shows the architecture of a monitor in terms of generated trigger and system reaction.

The PMU digital interface makes it possible to enable one of the following reactions:

- destructive reset
- functional reset
- interrupt
- FCCU trigger

If the FCCU trigger reaction is enabled, them through the FCCU registers is it possible to set a system reaction as described in *Chapter 65: Fault Collection and Control Unit (FCCU)*.

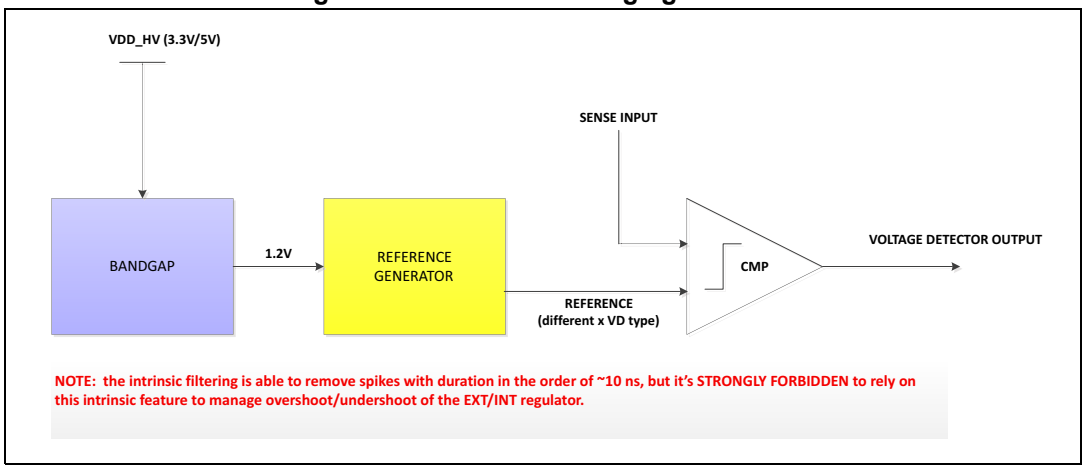
Figure 60. Monitor trigger and system reaction architecture



10.2.5.3 Reference Voltage

Figure 61 shows how is it generated the reference voltage used by the voltage monitor comparators.

Figure 61. Reference Voltage generation



10.2.5.4 Voltage Monitor activation

Figure 62 shows how the voltage monitors functionality and activation option.

Figure 62. Voltage monitor functionality

[1] : mandatory setting to be programmed in the TEST FLASH that cannot be overwritten (VD disabling) by the UTEST FLASH.
[2] : Destructive reset reaction can be set by UTEST or AIPS.
Functional reset reaction can be set by AIPS. AIPS can only modify the type of reaction (RES register -> destructive or functional) without to disable the REE setting by UTEST.
[3] IRQ/FCCU reaction type is USER defined (AIPS) according to the related application.
It's typically enabled if a destructive or functional reset reaction is disabled.

SW setting via AIPS registers init
Default setting (AIPS regs)
SW setting via DCF/AIPS regs init
Default setting w/o DCF init (user/test FLASH)

		Power ON exit	PH0D exit	PH1D-PH2D	Trimming	Post PH3D	Destr RST	Funct RST	IRQ	FCCU fault	VD disabling	Trimming	SW BIST
		PH1F-PH2F-PH3F-exit											
VD15	UVD600_C	Active (HW)									Test mode	Y	Y
VD14	LVD400_IOL1	NA				Active (SW)	user [2]	user [2]	user [3]	user [3]	Test mode or SW	Y	Y
							disabled	disabled	disabled	disabled			
VD14	LVD400_IOL0						user [2]	user [2]	user [3]	user [3]	Test mode or SW	Y	Y
							disabled	disabled	disabled	disabled			
VD14	LVD400_ADL1						user [2]	user [2]	user [3]	user [3]	Test mode or SW	Y	Y
							disabled	disabled	disabled	disabled			
VD14	LVD400_ADL0						user [2]	user [2]	user [3]	user [3]	Test mode or SW	Y	Y
							disabled	disabled	disabled	disabled			
VD13	HVD400_C						user [2]	user [2]	user [3]	user [3]	Test mode or SW	Y	Y
							disabled	disabled	disabled	disabled			
VD11	LVD290_IOL1						user [2]	user [2]	user [3]	user [3]	Test mode or SW	Y	Y
							disabled	disabled	disabled	disabled			
VD11	LVD290_XOSC						user [2]	user [2]	user [3]	user [3]	Test mode or SW	Y	Y
							disabled	disabled	disabled	disabled			
VD11	LVD290_ADL1						user [2]	user [2]	user [3]	user [3]	Test mode or SW	Y	Y
							disabled	disabled	disabled	disabled			
VD11	LVD290_ADL0						user [2]	user [2]	user [3]	user [3]	Test mode or SW	Y	Y
							disabled	disabled	disabled	disabled			
VD11	LVD290_F		Active (HW)		NA		active [1]				Test mode or SW	Y	Y
VD11	LVD290_C		Active (HW)		NA		active [1]				Test mode or SW	Y	Y
VD10	MVD270_F	Active (HW)									Test mode	Y	Y
VD10	MVD270_C	Active (HW)									Test mode	Y	Y
VD9	MVD240T_C	Active (HW)									Test mode	y	Y
VD8	POR200_C	Active (HW)									NA	NA	NA
VD7	UVD145_F	Active (HW)									Test mode	Y	Y
VD7	UVD145_C	Active (HW)									Test mode	Y	Y
VD6	HVD140_C	NA	NA			Active (SW)	user masked	user masked	user masked	user masked	Test mode or SW	Y	Y
VD3	LVD108_C		Active (HW)	NA		Active (SW)	active [1] masked	user masked	masked masked	masked masked	Test mode or SW	Y	Y
VD2	MVD098_F	Active (HW)									Test mode	Y	Y
VD2	MVD098_C	Active (HW)									Test mode	Y	Y
VD1	MVD085T_C	Active (HW)									Test mode	y	Y
VD0	POR041_C	Active (HW)									NA	NA	NA

VD = voltage detector, SW = software (option bit), NA = not available, Y = available, MVD = min VD, LVD = low VD, HVD = high VD, UVD = ultra VD

10.2.6 Device operating status versus power conditions

Table 144 refers to the scenario when the external regulator is used to generate the internal LV supply. The table shows the device operating status and possible safety reaction respect to the eventuality that one or some of the supplies are out of the operating.

Table 144. Device operating status versus Power when the external regulator is used

LV CORE supply	HV supply IO rail #0 / RCOSC	HV supply IO rail #1	ADC supply rail #0	ADC supply rail #1	FLASH HV supply (connected to IO rail #0)	XOSC HV supply (connected to IO rail #0)	Device operating status	Safety reaction
1.2V	3.3V / 5V	3.3V / 5V	3.3V / 5V	3.3V / 5V	Correctly powered by HV rail #0	Correctly powered by HV rail #0	Device FULLY WORKING at 3.3V/5V on each IO/ADC rail	-
Floating Ground Out of spec	Don´t care						DEVICE not WORKING	Destructive Reset
1.2V	Floating Ground Out of spec	Don´t care			Floating/Ground/Out of spec	Floating/Ground/Out of spec	DEVICE not WORKING	Destructive Reset
1.2V	3.3V / 5V	Floating Ground Out of spec	3.3V / 5V	3.3V / 5V	Correctly powered by HV rail #0	Correctly powered by HV rail #0	Device WORKING at 3.3V/5V on IO rail #0 and on both ADC rails. IO rail #1 not working	User defined
1.2V	3.3V / 5V	3.3V / 5V	Floating Ground Out of spec	3.3V / 5V	Correctly powered by HV rail #0	Correctly powered by HV rail #0	Device WORKING at 3.3V/5V on both IO rails and ADC rail #1. ADC rail #0 not working	User defined
1.2V	3.3V / 5V	3.3V / 5V	3.3V / 5V	Floating Ground Out of spec	Correctly powered by HV rail #0	Correctly powered by HV rail #0	Device WORKING at 3.3V/5V on both IO rails and ADC rail #0. ADC rail #1 not working	User defined
1.2V	3.3V / 5V	Floating Ground Out of spec	Floating Ground Out of spec	3.3V / 5V	Correctly powered by HV rail #0	Correctly powered by HV rail #0	Device WORKING at 3.3V/5V on IO rail #0 and ADC rail #1 ADC rail #0 and IO rail #1 not working	User defined

Table 144. Device operating status versus Power when the external regulator is used (continued)

LV CORE supply	HV supply IO rail #0 / RCOSC	HV supply IO rail #1	ADC supply rail #0	ADC supply rail #1	FLASH HV supply (connected to IO rail #0)	XOSC HV supply (connected to IO rail #0)	Device operating status	Safety reaction
1.2V	3.3V / 5V	Floating Ground Out of spec	3.3V / 5V	Floating Ground Out of spec	Correctly powered by HV rail #0	Correctly powered by HV rail #0	Device WORKING at 3.3V/5V on IO rail #0 and ADC rail #0 ADC rail #1 and IO rail #1 not working	User defined
1.2V	3.3V / 5V	3.3V / 5V	Floating Ground Out of spec	Floating Ground Out of spec	Correctly powered by HV rail #0	Correctly powered by HV rail #0	Device WORKING at 3.3V/5V on both IO rails. Both ADC rails not working	User defined
1.2V	3.3V / 5V	Floating Ground Out of spec	Floating Ground Out of spec	Floating Ground Out of spec	Correctly powered by HV rail #0	Correctly powered by HV rail #0	Device WORKING at 3.3V/5V on IO rail #0 Both ADC rails and IO rail #1 not working	User defined

Table 145 refers to the scenario when the internal regulator is used to generate the internal LV supply. The table shows the device operating status and possible safety reaction respect to the eventuality that one or some of the supplies are out of the operating range. Note that for this case it exists one single IO rail.

Table 145. Device operating status versus Power when the internal regulator is used

LV CORE supply	HV supply (IO rail #0 = IO rail #1)	ADC supply rail #0	ADC supply rail #1	FLASH HV supply (connected to IO rail #0)	XOSC/RCOSC HV supply (connected to IO rail #0)	Device operating status	Safety reaction
1.2V	Both 3.3V or both 5V	3.3V / 5V	3.3V / 5V	Correctly powered at 3.3V or 5V	Correctly powered by HV rail #0	Device FULLY WORKING at 3.3V or 5V (depending on the applied supply) on both IO rails, and at 3.3V/5V on both ADC rails	-
Floating Ground Out of spec	Don´t care					DEVICE not WORKING	Destructive Reset
1.2V	Floating Ground Out of spec	Don´t care		Floating / Ground / Out of spec	Floating / Ground / Out of spec	DEVICE not WORKING	Destructive Reset

Table 145. Device operating status versus Power when the internal regulator is used (continued)

LV CORE supply	HV supply (IO rail #0 = IO rail #1)	ADC supply rail #0	ADC supply rail #1	FLASH HV supply (connected to IO rail #0)	XOSC/RCOSC HV supply (connected to IO rail #0)	Device operating status	Safety reaction
1.2V	Both 3.3V or both 5V	Floating Ground Out of spec	3.3V / 5V	Correctly powered by HV rail #0	Correctly powered by HV rail #0	Device FULLY WORKING at 3.3V or 5V (depending on the applied supply) on both IO rails and at 3.3V/5V on ADC rail #1. ADC rail #0 not working	User defined
1.2V	Both 3.3V or both 5V	3.3V / 5V	Floating Ground Out of spec	Correctly powered by HV rail #0	Correctly powered by HV rail #0	Device FULLY WORKING at 3.3V or 5V (depending on the applied supply) on both IO rails and at 3.3V/5V on ADC rail #0. ADC rail #1 not working	User defined
1.2V	Both 3.3V or both 5V	Floating Ground Out of spec	Floating Ground Out of spec	Correctly powered by HV rail #0	Correctly powered by HV rail #0	Device FULLY WORKING at 3.3V or 5V (depending on the applied supply) on both IO rails . Both ADC rails not working	User defined

10.2.7 BIST details

To detect failures in the monitoring structure of the Power Management Unit, an internal BIST is implemented which can be run by programming PMU_Dig registers.

The default values of the registers assume a 70 MHz clock (pbridge_clk). The BIST_TIME registers should be programmed accordingly if the BIST is required to run at a different clock frequency (see *Table 147*).

The timing diagram in *Figure 63* explains the sequence for checking three different monitors: LVD_290FL, LVD_290AD and LVD_290C. All inputs and outputs are controlled inside the PMC_Dig module.

Figure 63. Monitor checking timing diagram

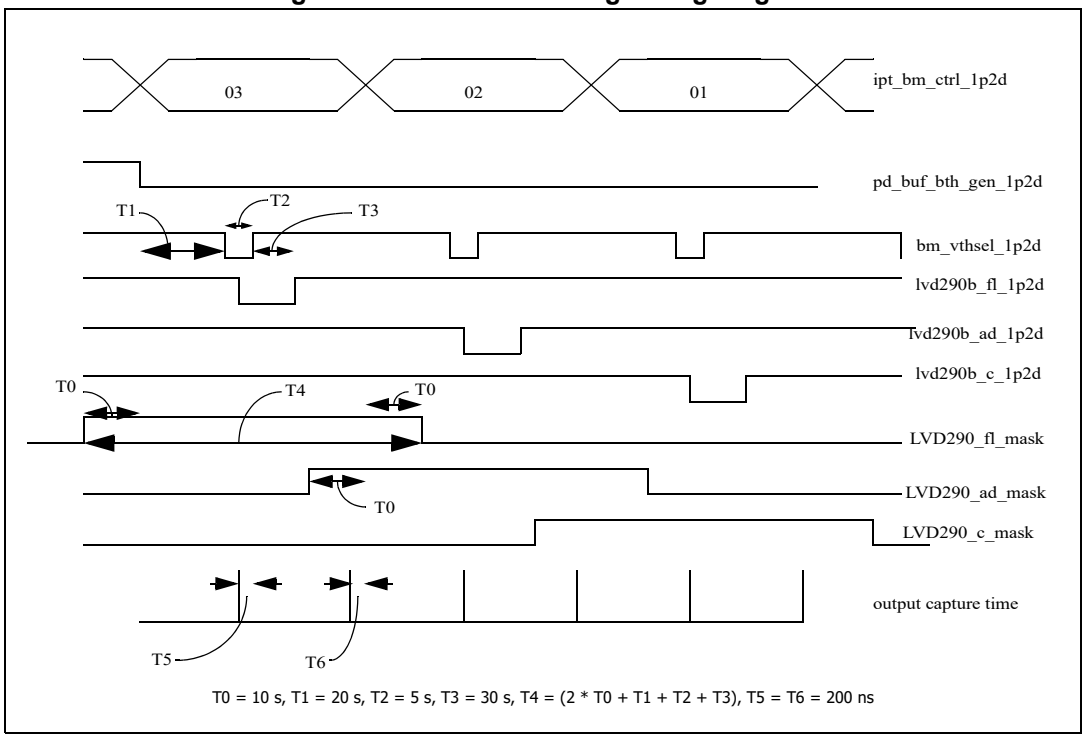


Table 146. Timing descriptions

Label	Description	Value
T0	the time that the monitor output is masked from the SoC before and after the monitor selection.	10 µs
T1	the wait time before the BIST testing for a specific LVD takes place. This is required to take care of the transients on the reference.	20 µs
T2	the time for which "bm_vthsel_1p2d" is kept logic 0.	5 µs
T3	the wait time before starting the testing of next monitor.	32 µs
T4	time for that the testing monitor output is kept masked from the SoC. T4 = 2 * T0 + T1 + T2 + T3	75 µs
T5	the time before "bm_vthsel_1p2d" becomes logic 1 such that the comparator logic 0 output can be checked	200 ns
T6	the time before the end of T3 phase to check logic 1 output of the comparator	200 ns

The BIST_TIME registers need to be programmed according to the following formula:

Equation 1 TIME_*x* = *T*x * pbridge_clk_freq - 1 where *x* is 0 to 3, 5 and 6.
See *Table 146* for the values of T*x*.

Table 147. Calculating the BIST_TIME values

PBRIDGE clock freq.	Base	TIME_0	TIME_1	TIME_2	TIME_3	TIME_5	TIME_6	BIST_TIME registers		
								TIME_10(1)	TIME_32(2)	TIME_65(3)
70 MHz	dec	699	1399	349	2099	14	14	0x057702BB	0x0833015D	0x000E000E
	hex	0x02BB	0x0577	0x015D	0x0833	0x000E	0x000E			
25 MHz	dec	249	499	124	749	5	5	0x01F300F9	0x02ED007C	0x00050005
	hex	0x00F9	0x01F3	0x007C	0x02ED	0x0008	0x0008			
40 MHz	dec	399	799	199	1199	8	8	0x031F018F	0x04AF00C7	0x00080008
	hex	0x018F	0x031F	0x00C7	0x04AF	0x0008	0x0008			

1. See *Section 50.5.36: User BIST Time1 and Time0 Register (BIST_TIME10)*
2. See *Section 50.5.37: User BIST Time3 and Time2 Register (BIST_TIME32)*
3. See *Section 50.5.38: User BIST Time6 and Time5 Register (BIST_TIME65)*

Note: When PMU BIST is being run there is a possibility that the value latched inside the EPR_HV1 register is incorrect.

Please clear the EPR_HV1 register after completion of BIST (see Section 50.5.16: Event Pending Register (EPR_HV1)).

Table 148 shows the individual monitors to check.

Table 148. Monitors to check

Monitor	BIST_FLAGS bit number(1)	
	FLAGS_P2	FLAGS_P1
UVD600_C	2	18
UVD145_C	3	19
UVD145_FL	4	20
HVD400_C	5	21
HVD140_C	6	22
LVD400_AD	7	23
LVD290_C	8	24
LVD290_AD	9	25
LVD290_FL	10	26
LVD108_C	11	27
MVD270_C	12	28
MVD270_FL	13	29
MVD098_C	14	30
MVD098_FL	15	31

1. See *Section 50.5.34: User BIST Flags Register (BIST_FLAGS)*

11 Security

This device has a comprehensive set of customer-configurable security features designed to protect code and data from unauthorized access.

11.1 Basic security

All SPC574Sx devices have the following basic set of security features.

- Device censorship based on the life cycle model with code and data access progressively more restricted as device matures through defined life cycle steps
- Memory security features:
  - NVM censorship support, password protection, one-time-programmable (OTP) flash memory areas.
- Monitoring of operating conditions
- Unique ID for each device: Each SPC574Sx device has a unique identification number stored in an OTP flash memory area which can be read by any of the cores on the device.
- Basic debugger restrictions (on/off via censorship mode)

11.2 Advanced security

Depending on the device configuration, the following features are available:

- Secure debugger interface
- Boot mode: Trusted/secure boot support

11.3 Detailed security information

Details of most of the SPC574Sx security features are published in a separate *SPC574Sx Microcontroller Security Reference Manual*, which is only available to qualified customers.

12 Core e200z420n3 Description

12.1 Overview of the e200z420n3 Core

e200z420n3 is a dual-issue 32-bit <asisitalic>PowerISA 2.06 VLE compliant design with 32-bit general-purpose registers (GPRs). The e200z420n3 core implements the VLE (variable-length encoding) ISA, providing improved code density. The VLE ISA is further documented in <asisitalic>PowerISA 2.06, a separate document.

An Embedded Floating-point (EFPU2) APU is provided to support real-time single-precision floating-point embedded numerics operations using the general-purpose registers.

The e200z420n3 processor integrates a pair of integer execution units, a branch control unit, instruction fetch unit, load/store unit, and a multi-ported register file capable of sustaining six read and three write operations per clock cycle. Most integer instructions execute in a single clock cycle. Branch target prefetching is performed by the branch unit to allow single-cycle branches in many cases.

e200z420n3 contains a 8 KB Instruction Cache, and a 4 KB Data Cache and 32KB DMEM, as well as a Nexus Class 3+ real-time debug module with support for program and data trace features as well as extensive trace controls.

Note: There is no IMEM memory support in this core (IMEM related registers are available but with no effect).

A Memory Protection Unit that protects various instruction and data memory areas is also included.

12.2 Register Model

This section describes the registers implemented in the e200z420n3 core. The <asisitalic>PowerISA 2.06 architecture defines register-to-register operations for all computational instructions.

e200z420n3 extends usage of the general purpose registers to support EFPU operations on the 32-bit GPR registers.

Figure 64 and *Figure 65* show the complete e200z420n3 register set, all of which are accessible in supervisor mode. *Figure 66* and *Figure 67* show the subset of registers accessible in user mode. The number to the right of the special-purpose registers (SPRs) is the decimal number used in the instruction syntax to access the register (for example, the integer exception register (XER) is SPR 1).

Figure 64. e200z420n3 SUPERVISOR Mode Programmer's Model SPRs

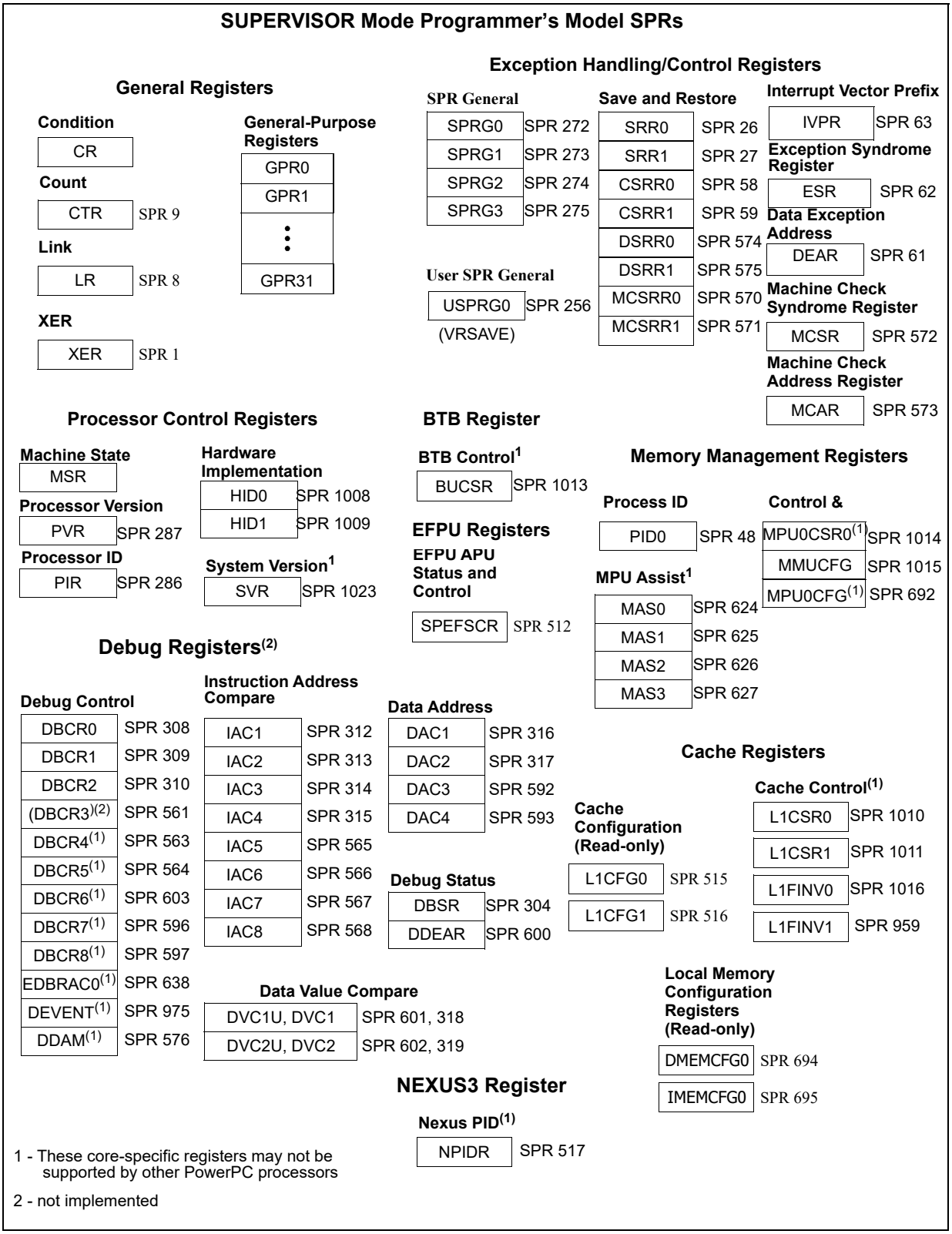


Figure 65. e200z420n3 Supervisor Mode Programmer's Model DCRs and PMRs

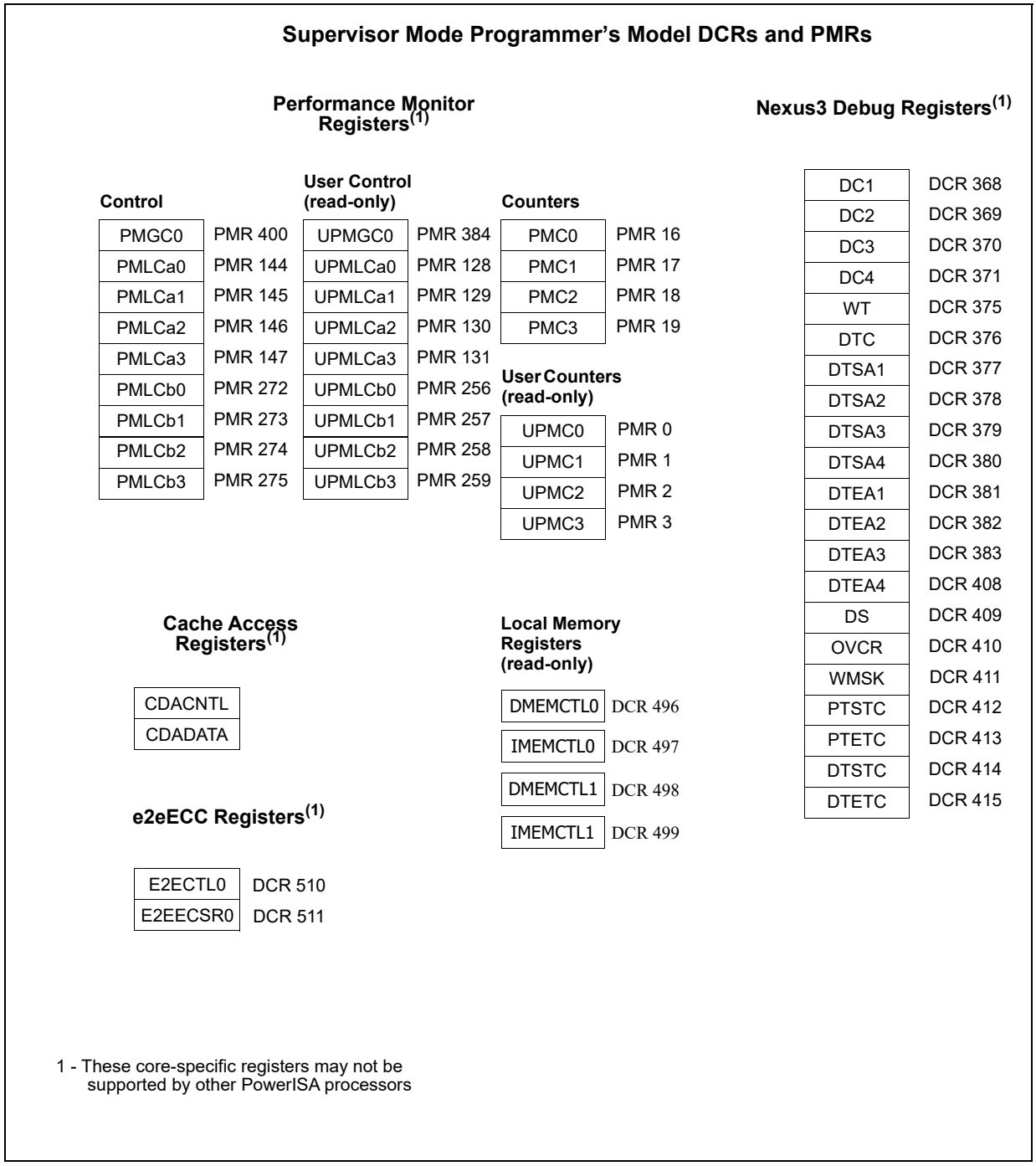


Figure 66. e200z420n3 User Mode Programmer's Model SPRs

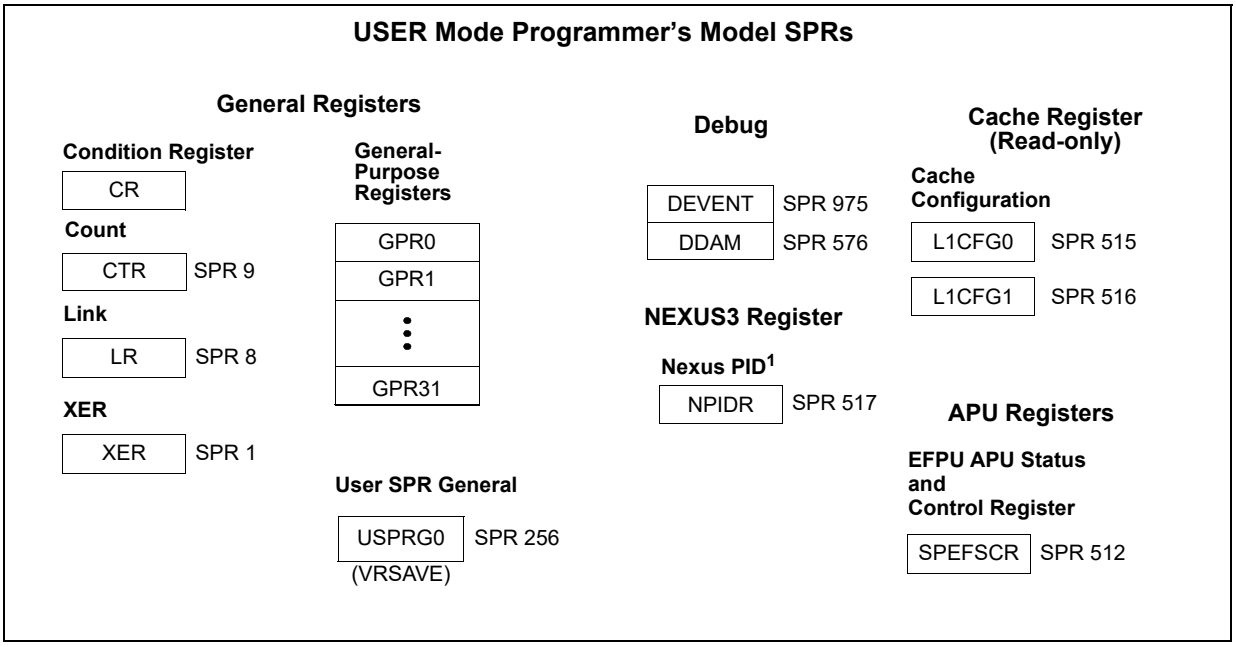
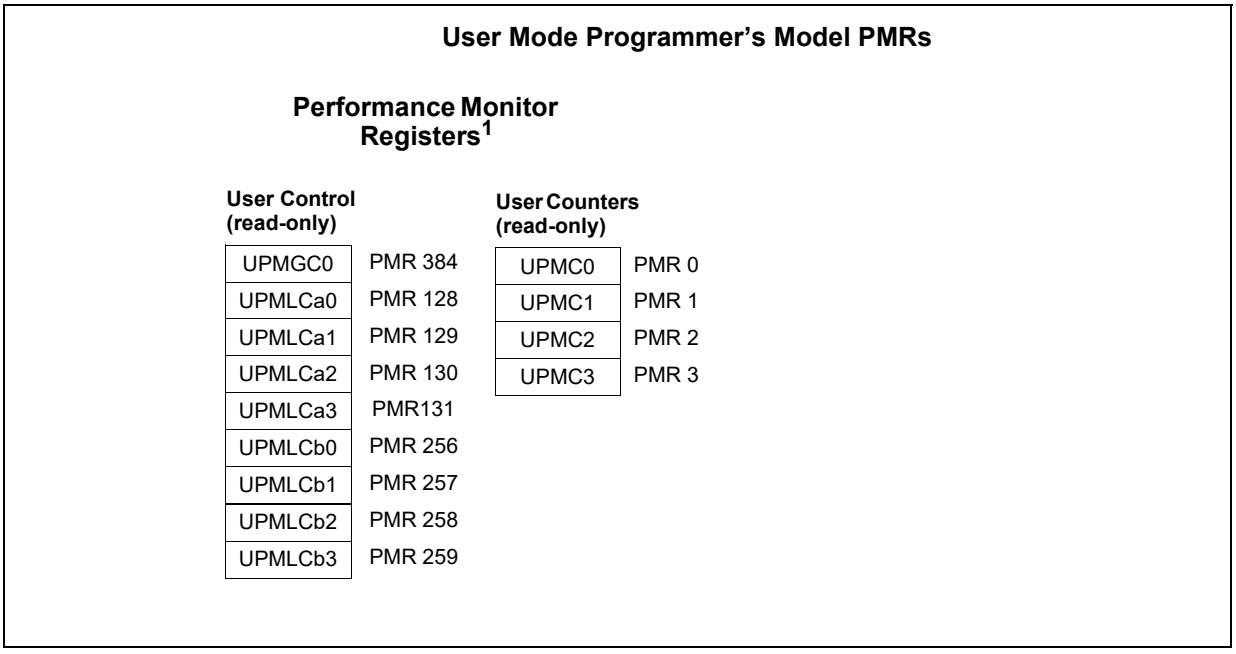


Figure 67. e200z420n3 User Mode Programmer's Model PMRs



12.3 Dual-issue operation

The instruction issue unit attempts to issue a pair of instructions to the execution units each cycle. Source operands for each of the instructions are provided from the GPRs or from the operand feed-forward muxes. Data or resource hazards may create stall conditions that cause instruction issue to be stalled for one or more cycles until the hazard is eliminated.

The execution units write the result of a finished instruction onto the proper result bus and into the destination registers. The writeback logic retires an instruction when the instruction has finished execution. Up to three results can be simultaneously written.

Two execution units are provided to allow dual issue of most instructions. Only a single load/store unit is provided. Only a single integer multiply and divide unit is provided, thus a pair of multiply or divide instructions cannot issue simultaneously. In addition, the divide unit is blocking.

Table 149 shows the e200z420n3 concurrent instruction issue capabilities. Note that data dependencies between instructions will generally preclude dual-issue of those instructions.

Table 149. Concurrent instruction issue capabilities for e200z420n3

Class of instruction	Branch	Load/Store	Scalar integer	Scalar float	Special
Branch	—	yes	yes	yes	—
Load/Store	yes	—	yes	yes	—
Scalar integer	yes	yes	yes(1)	yes	—
Scalar float	yes	yes	yes	—	—
Special	—	—	—	—	—

1. Excludes multiply or divide class instructions occurring in both issue slots.

12.4 Reservation instructions and cache interactions

The CPU treats reservation instruction (**lbarx**, **lharx**, **lwarx**, **stbcx.**, **sthcx.**, and **stwcx.**) accesses as though they were cache-inhibited, and forces a cache miss. A reservation access is always issued to the bus. This is done to allow external reservation logic to be built which properly signals a reservation failure. The bus access will be treated as a single-beat transfer.

12.5 Signal Processing Extension/ Embedded Floating-point Status and Control Register (SPEFSCR)

Status and control for embedded floating-point operations use the SPEFSCR. The SPEFSCR is implemented as special-purpose register (SPR) number 512 and is read and written by the mfspr and mtspr instructions. The SPEFSCR is shown in *Figure 68*

Offset: Refer to *Figure 64* (SPR - 512) Access: Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0										FINXS	FINVS	FDBZS	FUNFS	FOVFS	MODE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	FG	FX	FINV	FDBZ	FUNF	FOVF	0	FINXE	FINVE	FDBZE	FUNFE	FOVFE	FRMC	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 68. EFPU Status and Control Register (SPEFSCR)

The SPEFSCR bits are defined in *Table 150*.

Table 150. EFPU Status and Control Register fields description

Field	Description
10 FINXS	Embedded Floating-point Inexact Sticky Flag The FINXS bit is set to 1 whenever the execution of a floating-point instruction delivers an inexact result and no Floating-point Data exception is taken, or if the result of a Floating-point instruction results in overflow (FOVF = 1), but Floating-point Overflow exceptions are disabled (FOVFE = 0), or if the result of a Floating-point instruction results in underflow (FUNF = 1), but Floating-point Underflow exceptions are disabled (FUNFE = 0), and no Floating-point Data exception occurs. The FINXS bit remains set until it is cleared by a **mtspr** instruction specifying the SPEFSCR.
11 FINVS	Embedded Floating-point Invalid Operation Sticky Flag The FINVS bit is set to a 1 when a floating-point instruction sets the FINV bit to 1. The FINVS bit remains set until it is cleared by a **mtspr** instruction specifying the SPEFSCR.
12 FDBZS	Embedded Floating-point Divide by Zero Sticky Flag The FDBZS bit is set to 1 when a floating-point divide instruction sets the FDBZ bit to 1. The FDBZS bit remains set until it is cleared by a **mtspr** instruction specifying the SPEFSCR.
13 FUNFS	Embedded Floating-point Underflow Sticky Flag The FUNFS bit is set to 1 when a floating-point instruction sets the FUNF bit to 1. The FUNFS bit remains set until it is cleared by a **mtspr** instruction specifying the SPEFSCR.
14 FOVFS	Embedded Floating-point Overflow Sticky Flag The FOVFS bit is set to 1 when a floating-point instruction sets the FOVF bit to 1. The FOVFS bit remains set until it is cleared by a **mtspr** instruction specifying the SPEFSCR.
15 MODE	Embedded Floating-point Operating Mode This bit controls the operating mode of the EFPU. e200z420n3 supports only mode 0. Software should read the value of this bit after writing it to determine if the implementation supports the selected mode. Implementations will return the value written if the selected mode is a supported mode, otherwise the value read will indicate the hardware supported mode. 0 Default hardware results operating mode 1 IEEE754 hardware results operating mode (not supported by Zen)

Table 150. EFPU Status and Control Register fields description (continued)

Field	Description
17 OV	Integer overflow Defined by LSP APU.
18 FG	Embedded Floating-point Guard bit FG is supplied for use by the Floating-point Round exception handler. FG is zeroed if a Floating-point Data Exception occurs.
19 FX	Embedded Floating-point Sticky bit FX is supplied for use by the Floating-point Round exception handler.FX is zeroed if a Floating-point Data Exception occurs.
20 FINV	Embedded Floating-point Invalid Operation / Input error In mode 0, the FINV bit is set to 1 if the A or B operand of a floating-point instruction is Infinity, NaN, or Denorm, or if the operation is a divide and the dividend and divisor are both 0. In mode 1, the FINV bit is set on an IEEE754 invalid operation (IEEE754-1985 sec7.1).
21 FDBZ	Embedded Floating-point Divide by Zero The FDBZ bit is set to 1 when a floating-point divide instruction executed with divisor of 0, and the l dividend is a finite non-zero number.
22 FUNF	Embedded Floating-point Underflow The FUNF bit is set to 1 when the execution of a floating-point instruction results in an underflow.
23 FOVF	Embedded Floating-point Overflow The FOVF bit is set to 1 when the execution of a floating-point instruction results in an overflow.
25 FINXE	Embedded Floating-point Inexact Exception Enable If the exception is enabled, a Floating-point Round exception is taken if the result of a Floating-point instruction does not result in overflow or underflow, and the result is inexact (FG | FX = 1), or if the result of a Floating-point instruction does result in overflow (FOVF = 1) but Floating-point Overflow exceptions are disabled (FOVFE = 0), or if the result of a Floating-point instruction results in underflow (FUNF = 1 or FUNFH = 1) but Floating-point Underflow exceptions are disabled (FUNFE = 0), and no Floating-point Data exception occurs. 0 Exception disabled 1 Exception enabled
26 FINVE	Embedded Floating-point Invalid Operation / Input Error Exception Enable If the exception is enabled, a Floating-point Data exception is taken if the FINV bit is set by a floating-point instruction. 0 Exception disabled 1 Exception enabled
27 FDBZE	Embedded Floating-point Divide by Zero Exception Enable If the exception is enabled, a Floating-point Data exception is taken if the FDBZ bit is set by a floating-point instruction. 0 Exception disabled 1 Exception enabled

Table 150. EFPU Status and Control Register fields description (continued)

Field	Description
28 FUNFE	Embedded Floating-point Underflow Exception Enable If the exception is enabled, a Floating-point Data exception is taken if the FUNF bit is set by a floating-point instruction. 0 Exception disabled 1 Exception enabled
29 FOVFE	Embedded Floating-point Overflow Exception Enable If the exception is enabled, a Floating-point Data exception is taken if the FOVF bit is set by a floating-point instruction. 0 Exception disabled 1 Exception enabled
30:31 FRMC	Embedded Floating-point Rounding Mode Control 00 Round to Nearest 01 Round toward Zero 10 Round toward +Infinity 11 Round toward -Infinity

12.6 Cache

This section describes the cache registers, cache control instructions, and various cache operations.

12.6.1 Cache overview

The e200z420n3 processor supports an 8 KB 2-way set-associative instruction and 4 KB 2-way set-associative data cache with a 32-byte line size. The caches improve system performance by providing low-latency data to the e200z420n3 instruction and data pipelines, which decouples processor performance from system memory performance.

The e200z420n3 processor also contains an 8-entry store buffer to decouple store completion to the processor from store completion on the data interface to further improve system performance.

Instruction and data addresses from the processor are used to index the cache arrays. If the access address matches a valid cache tag entry, the access hits in the cache.

12.6.2 L1 Cache Control and Status Register 0 (L1CSR0)

The L1 Cache Control and Status Register 0 (L1CSR0) is a 32-bit register used for general control of the data cache and for disabling ways in both caches. The L1CSR0 register is accessed using a **mfspr** or **mtspr** instruction. The SPR number for L1CSR0 is 1010 in decimal. The L1CSR0 register is shown in *Figure 69*.

Offset: Refer to *Figure 64* (SPR - 1010) Access: Read/Write

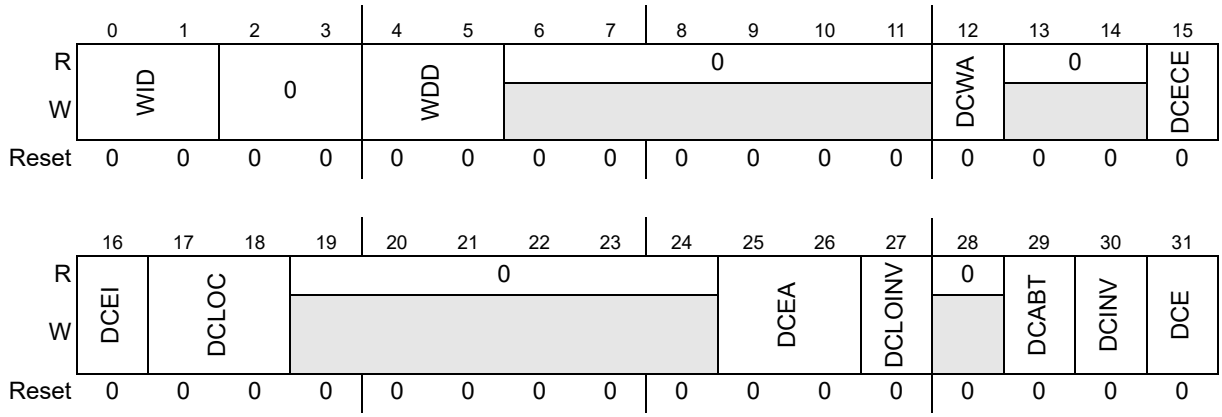


Figure 69. L1 Cache Control and Status Register 0 (L1CSR0)

The L1CSR0 fields are described in *Table 151*.

Table 151. L1CSR0 fields description

Field	Description
0:1 WID	Way Instruction Disable Bit 0 corresponds to way 0. Bit 1 corresponds to way 1. The WID bits may be used for locking ways of the instruction cache, and also affect the replacement policy of the instruction cache. 0 The corresponding way in the instruction cache is available for replacement by instruction miss line fills. 1 The corresponding way instruction cache is not available for replacement by instruction miss line fills.
4:5 WDD	Way Data Disable Bit 4 corresponds to way 0. Bit 5 corresponds to way 1. The WDD bits may be used for locking ways of the data cache, and also affect the replacement policy of the data cache. 0 The corresponding way in the data cache is available for replacement by data miss line fills. 1 The corresponding way in the data cache is not available for replacement by data miss line fills.
12 DCWA	Data Cache Write Allocation Policy This bit also controls merging of store data into the linefill buffer while a cache linefill is in progress. Store data will not be merged when write allocation is disabled. 0 Cache line allocation on a cacheable write miss is disabled 1 Cache line allocation on a cacheable write miss is enabled
15 DCECE	Data Cache Error Checking Enable 0 Error Checking is disabled 1 Error Checking is enabled

Table 151. L1CSR0 fields description (continued)

Field	Description
16 DCEI	Data Cache Error Injection DCEI will cause injection of errors regardless of the setting of DCECE, although reporting of errors will be masked while DCECE = 0. 0 Cache Error Injection is disabled 1 A double-bit error will be injected on each write into the cache data array by inverting the two uppermost parity check bits (p_dchk[0:1]). This includes writes due to store hits as well as writes due to cache line refills.
17:18 DCLOC	Data Cache Lockout Control **Note:** For the **dcbi** and **dcbf** instructions, and for specialized load/store instructions, no lockout operation is performed, regardless of the occurrence of EDC/ECC error conditions, and errors are handled in the same manner as when lockout is disabled. **Note:** When operating in MC mode (DCEA = 00), detected errors on cache-inhibited accesses will not cause a data cache line to be locked out, instead the cache line contents are ignored. 00 Cache line lockout is disabled (and array LO indicators are ignored). 01 Cache line lockout is enabled. Cache lines with any tag errors or any data errors will have the corresponding lockout indicators set. A Machine check will not be generated for the error if the operation would have normally generated one unless a locked line is locked out. 10 Cache line lockout is enabled. Cache lines with any tag or data errors will have the corresponding lockout indicators set. A Machine check will still be generated for an error if the operation would have normally generated one. 11 Cache line lockout is enabled. Cache lines with uncorrectable tag errors or any data errors will have the corresponding lockout indicators set. A Machine check will not be generated for the error if the operation would have normally generated one unless a locked line is locked out. If a correctable tag error occurs, and is re-detected on recycling the access after a correction cycle for the tag is performed, the line is locked out regardless of detecting a single-bit or multi-bit error.
25:26 DCEA	Data Cache Error Action 00 Error Detection causes Machine Check exception. 01 Error Detection causes Correction/Auto-invalidation. No machine check is generated for most cases. Correction is performed for single-bit tag errors, and lines with multi-bit tag errors are invalidated. Correction is performed for single or multi-bit data errors on cache hits by reloading of the line. 10 Reserved 11 Reserved

Table 151. L1CSR0 fields description (continued)

Field	Description
27 DCLOINV	Data Cache Lockout Indicator Invalidate When written to a 1 in conjunction with writing DCINV to a 1, a cache lockout indicator invalidation operation is initiated by hardware, and the LO bits are cleared, removing all line lockout status. Once complete, this bit is reset to 0. Writing a 1 while an invalidation operation is in progress will result in an undefined operation. Writing a 0 to this bit while an invalidation operation is in progress will be ignored. Cache lockout bit invalidation operations require approximately 66 cycles to complete. Invalidation occurs regardless of the data cache enable (DCE) value. When DCINV = 0: Reserved, do not set to 1 When DCINV = 1: 0 No cache lockout bit invalidate 1 Cache lockout indicator invalidation operation
29 DCABT	Data Cache Operation Aborted Indicates a Cache Invalidate operation was aborted prior to completion. This bit is set by hardware on an aborted condition, and will remain set until cleared by software writing 0 to this bit location.
30 DCINV	Data Cache Invalidate When written to a 1, a cache invalidation operation is initiated by hardware. Once complete, this bit is reset to 0. Writing a 1 while an invalidation operation is in progress will result in an undefined operation. Writing a 0 to this bit while an invalidation operation is in progress will be ignored. Cache invalidation operations require approximately 66 cycles to complete. Invalidation occurs regardless of the data cache enable (DCE) value. 0 No cache invalidate 1 Cache invalidation operation
31 DCE	Data Cache Enable When disabled, cache lookups are not performed for normal load or store accesses. Other L1CSR0 cache control operations are still available. Also, operation of the store buffer is not affected by DCE. 0 Cache is disabled 1 Cache is enabled

12.6.3 L1 Cache Control and Status Register 1 (L1CSR1)

The L1 Cache Control and Status Register 1 (L1CSR1) is a 32-bit register used for general control of the instruction cache. The L1CSR1 register is accessed using a **mfspr** or **mtspr** instruction. The SPR number for L1CSR1 is 1011 in decimal. The L1CSR1 register is shown in *Figure 70*.

Offset: Refer to *Figure 64* (SPR - 1011) Access: Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	0(1)															ICECE
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	ICEI	ICLOC			0(1)					ICEA		ICLOINV	0(1)	ICABT	ICINV	ICE
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. These bits are not implemented and should be written with zero for future compatibility.

Figure 70. L1 Cache Control and Status Register 1 (L1CSR1)

The L1CSR1 fields are described in *Table 152*.

Table 152. L1CSR1 fields description

Field	Description
15 ICECE	Instruction Cache Error Checking Enable 0 Error Checking is disabled 1 Error Checking is enabled
16 ICEI	Instruction Cache Error Injection Enable ICEI will cause injection of errors regardless of the setting of ICECE, although reporting of errors will be masked when ICECE = 0. 0 Cache Error Injection is disabled 1 A double-bit error will be injected into each doubleword written into the cache by inverting the two uppermost parity check bits (p_chk[0:1]).
17:18 ICLOC	Instruction Cache Lockout Control **Note:** For the **icbi** instruction, no lockout operation is performed, regardless of the occurrence of EDC/ECC error conditions, and errors are handled in the same manner as when lockout is disabled. When operating in MC mode (ICEA = 00), detected errors on cache-inhibited accesses will not cause an instruction cache line to be locked out, instead the cache line contents are ignored. 00 Cache line lockout is disabled (and array LO indicators are ignored). 01 Cache line lockout is enabled. Cache lines with any tag errors or any data errors will have the corresponding lockout indicators set. A Machine check will not be generated for the error if the operation would have normally generated one unless a locked line is locked out. 10 Cache line lockout is enabled. Cache lines with any tag or data errors will have the corresponding lockout indicators set. A Machine check will still be generated for an error if the operation would have normally generated one. 11 Cache line lockout is enabled. Cache lines with uncorrectable tag errors or any data errors will have the corresponding lockout indicators set. A Machine check will not be generated for the error if the operation would have normally generated one unless a locked line is locked out. If a correctable tag error occurs, and is re-detected on recycling the access after a correction cycle for the tag is performed, the line is locked out regardless of detecting a single-bit or multi-bit error.

Table 152. L1CSR1 fields description (continued)

Field	Description
25:26 ICEA	Instruction Cache Error Action 00 Error Detection causes Machine Check exception. 01 Error Detection causes Correction/Auto-invalidation. No machine check is generated for most cases. Correction is performed for single-bit tag errors, and lines with multi-bit tag errors are invalidated. Correction is performed for single or multi-bit data errors on cache hits by reloading of the line. 10 Reserved 11 Reserved
27 ICLOINV	Instruction Cache Lockout Indicator Invalidate When written to a 1 in conjunction with writing ICINV to a 1, a cache lockout indicator invalidation operation is initiated by hardware, and the LO bits are cleared, removing all line lockout status. Once complete, this bit is reset to 0. Writing a 1 while an invalidation operation is in progress will result in an undefined operation. Writing a 0 to this bit while an invalidation operation is in progress will be ignored. Cache lockout bit invalidation operations require approximately 66 cycles to complete. Invalidation occurs regardless of the instruction cache enable (ICE) value. When ICINV = 0: Reserved, do not set to 1 When ICINV = 1: 0 No cache lockout bit invalidate 1 Cache lockout indicator invalidation operation
29 ICABT	Instruction Cache Operation Aborted Indicates a Cache Invalidate operation was aborted prior to completion. This bit is set by hardware on an aborted condition, and will remain set until cleared by software writing 0 to this bit location.
30 ICINV	Instruction Cache Invalidate When written to a 1, a cache invalidation operation is initiated by hardware. Once complete, this bit is reset to 0. Writing a 1 while an invalidation operation is in progress will result in an undefined operation. Writing a 0 to this bit while an invalidation operation is in progress will be ignored. Cache invalidation operations require approximately 130 cycles to complete. Invalidation occurs regardless of the enable (ICE) value. 0 No cache invalidate 1 Cache invalidation operation
31 ICE	Instruction Cache Enable When disabled, cache lookups are not performed for instruction accesses. Other L1CSR1 cache control operations are still available and are not affected by ICE. 0 Cache is disabled 1 Cache is enabled

12.6.4 L1 Cache Configuration Register 0 (L1CFG0)

The L1 Cache Configuration Register 0 (L1CFG0) is a 32-bit read-only register. L1CFG0 provides information about the configuration of the e200z420n3 L1 data cache design. The contents of the L1CFG0 register can be read using a **mfspr** instruction. The SPR number for L1CFG0 is 515 in decimal. The L1CFG0 register is shown in *Figure 71*.

Offset: Refer to *Figure 64* (SPR - 515) Access: Read only

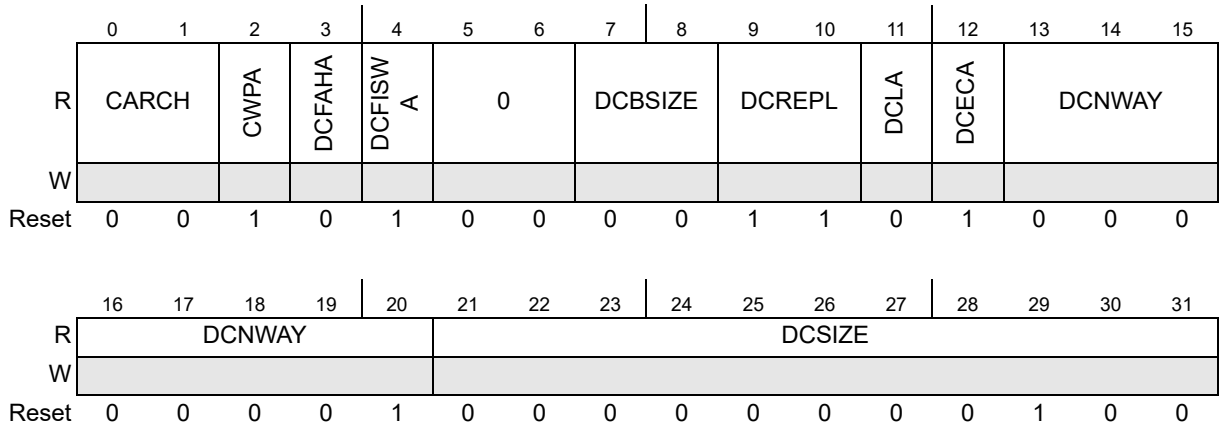


Figure 71. L1 Cache Configuration Register 0 (L1CFG0)

The L1CFG0 bits are described in *Table 153*.

Table 153. L1CFG0 fields description

Field	Description
0:1 CARCH	Cache Architecture 00 The cache architecture is Harvard
2 CWPA	Cache Way Partitioning Available 1 The caches support partitioning of way availability for I/D accesses
3 DCFAHA	Data Cache Flush All by Hardware Available 0 The data cache does not support Flush All in Hardware
4 DCFISWA	Data Cache Flush/Invalidate by Set and Way Available 1 The data cache supports invalidation by Set and Way via L1FINV0
5:6	Reserved - read as zeros
7:8 DCBSIZE	Data Cache Block Size 00 The data cache implements a block size of 32 bytes
9:10 DCREPL	Data Cache Replacement Policy 11 The data cache implements a FIFO replacement policy
11 DCLA	Data Cache Locking APU Available 0 The data cache does not implement the line locking APU
12 DCECA	Data Cache Error Checking Available 1 The data cache implements error checking
13:20 DCNWAY	Data Cache Number of Ways 0x01 The data cache is 2-way set-associative
21:31 DCSIZE	Data Cache Size 0x004 The size of the data cache is 4 KB

12.6.5 L1 Cache Configuration Register 1 (L1CFG1)

The L1 Cache Configuration Register 1 (L1CFG1) is a 32-bit read-only register. L1CFG1 provides information about the configuration of the e200z420n3 L1 instruction cache design. The contents of the L1CFG1 register can be read using a **mfspr** instruction. The SPR number for L1CFG1 is 516 in decimal. The L1CFG1 register is shown in *Figure 72*.

Offset: Refer to *Figure 64* (SPR - 516) Access: Read only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0				ICFISWA	0		ICBSIZE		ICREPL		ICLA	ICECA	ICNWAY[13:15]		
W																
Reset	0	0	0	0	1	0	0	0	0	1	1	0	1	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	ICNWAY[16:19]					ICSIZE										
W																
Reset	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0

Figure 72. L1 Cache Configuration Register 1 (L1CFG1)

The L1CFG1 fields are described in *Table 154*.

Table 154. L1CFG1 fields description

Field	Description
4 ICFISWA	Instruction Cache Flush/Invalidate by Set and Way Available 1 The instruction cache supports invalidation by Set and Way via L1FINV1
7:8 ICBSIZE	Instruction Cache Block Size 00 The instruction cache implements a block size of 32 bytes
9:10 ICREPL	Instruction Cache Replacement Policy 11 The instruction cache implements a FIFO replacement policy
11 ICLA	Instruction Cache Locking APU Available 0 The instruction cache does not implement the line locking APU
12 ICECA	Instruction Cache Error Checking Available 1 The instruction cache implements error checking
13:20 ICNWAY	Instruction Cache Number of Ways 0x01 The instruction cache is 2-way set-associative
21:31 ICSIZE	Instruction Cache Size 0x008 The size of the instruction cache is 8 KB

12.6.6 Data Cache Software Coherency

Data cache coherency is supported through software operations to invalidate lines using either a **dcbi** or **dcbf** instruction, or by using the L1FINV0 control register.

Data cache misses will force the store buffer to empty prior to performing the access.

12.6.7 Data Cache Hardware Coherency

Data cache coherency is not supported in hardware. Software operations must generally be used to maintain coherency. Debug support is provided, however, for a D-Cache line invalidation operation requested by an external hardware debugger. When requested by an external hardware debugger tool operating through the OnCE port, an individual cache line invalidation operation will be scheduled to occur at the next available D-Cache access cycle.

12.6.8 Cache Invalidate by Set and Way

e200z420n3 supports cache set/way invalidation under software control. The caches may be invalidated by index and way through a **mtspr l1finv{0,1}** instruction.

The L1 Flush and Invalidate Control Registers (L1FINV{0,1}) are 32-bit SPRs used to select a cache set and way to be invalidated. This function is available even when a cache is disabled. L1FINV0 is used for data cache operations, while L1FINV1 is used for instruction cache operations.

12.6.8.1 L1FINV0

The SPR number for L1FINV0 is 1016 in decimal. The L1FINV0 register is shown in *Figure 73*.

Offset: Refer to *Figure 64* (SPR - 1016) Access: Read/Write

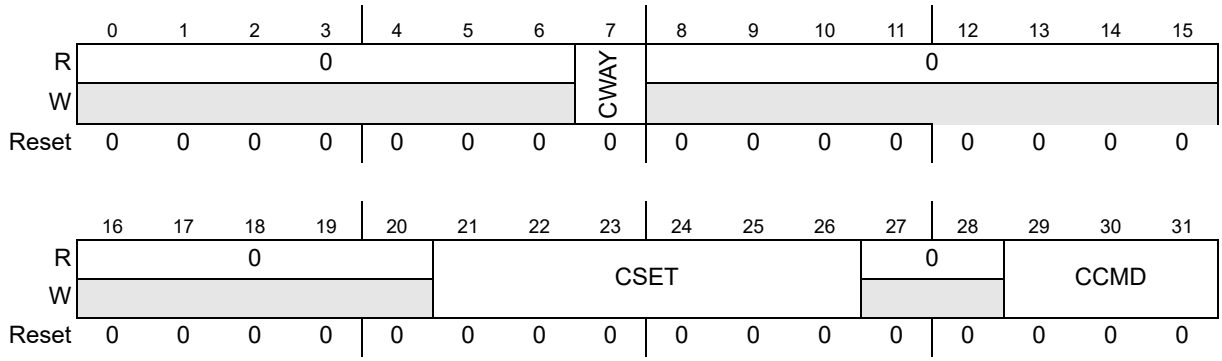


Figure 73. L1 Flush/Invalidate Register 0 (L1FINV0)

The L1FINV0 bits are described in *Table 155*.

Table 155. L1FINV0 fields description

Field	Description
0:6	Reserved(1) for way extension
7 CWAY	Cache Way Specifies the data cache way to be selected
8:20	Reserved(1) for set extension
21:26 CSET	Cache Set Specifies the cache set to be selected

Table 155. L1FINV0 fields description (continued)

Field	Description
27:28	Reserved(1) for set/command extension
29:31 CCMD	Cache Command 000 The data contained in this entry is invalidated. LO bits are unaffected 001 Reserved 01x Reserved 100 Reserved 101 The data contained in this entry is invalidated. LO bits are cleared 11x Reserved

1. These bits are not implemented and should be written with zero for future compatibility.

For invalidation operations via L1FINV0, tag ECC errors and data EDC errors are ignored, and the selected line will be invalidated regardless of any error. Invalidations conditionally affect the state of the lockout bits (LO).

12.6.8.2 L1FINV1

The SPR number for L1FINV1 is 959 in decimal. The L1FINV1 register is shown in *Figure 74*.

Offset: Refer to *Figure 64* (SPR - 959) Access: Read/Write

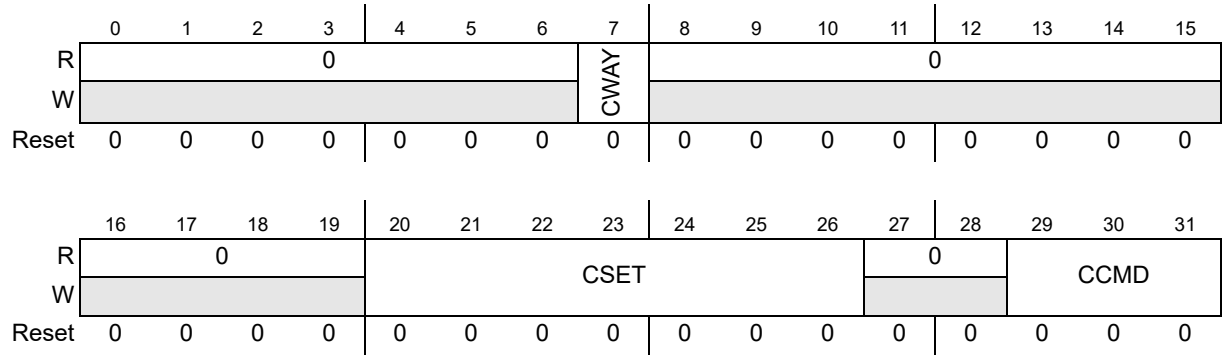


Figure 74. L1 Flush/Invalidate Register 1 (L1FINV1)

The L1FINV1 bits are described in *Table 156*.

Table 156. L1FINV1 fields description

Field	Description
0:6	Reserved(1) for way extension
7 CWAY	Cache Way Specifies the instruction cache way to be selected
8:19	Reserved(1) for set extension
20:26 CSET	Cache Set Specifies the instruction cache set to be selected

Table 156. L1FINV1 fields description (continued)

Field	Description
27:28	Reserved(1) for set/command extension
29:31 CCMD	Cache Command 000 The data contained in this entry is invalidated. LO bits are unaffected 001 Reserved 01x Reserved 100 Reserved 101 The data contained in this entry is invalidated. LO bits are cleared 11x Reserved

1. These bits are not implemented and should be written with zero for future compatibility.

For invalidation operations via L1FINV1, tag ECC errors and data EDC errors are ignored, and the selected line will be invalidated regardless of any error. Invalidations conditionally affect the state of the lockout bits (LO).

12.6.9 Cache EDC/ECC Parity Protection

Cache parity protection is supported for both the tag and data arrays of each cache. Seven parity check bits are provided for each tag entry for the tag arrays of both caches to support error detection and correction (ECC - SECDED). Eight parity check bits are provided for each doubleword in the data arrays of the I-Cache, and each word in the data arrays of the D-Cache, which are used for single- and double-bit error detection (EDC - DED, double error detection). Utilizing ECC/EDC protection, many multi-bit errors are also detected.

The error detection codes also cover the cache index address of the cache line, providing additional protection against addressing errors. If any type of error (including a single-bit error in one of the index bits) is encountered in one of the index address bits, it is treated as an uncorrectable tag ECC error to protect against internal addressing failures. No attempt will be made to correct single-bit errors where the syndrome indicates an index address bit.

Tag ECC and data EDC checking is controlled by the $L1CSR0_{DCECE}$ and $L1CSR1_{ICECE}$ control fields. When error checking is enabled, checking is performed on each cache access. Errors are not signaled by the respective cache when cache error checking is disabled for that cache ($L1CSR\{0,1\}_{[D,I]CECE} = 0$).

If an uncorrectable tag ECC error is detected on any portion of a tag accessed during cache lookups performed because of instruction fetching, normal loads, or normal stores, a tag ECC error is signaled, regardless of whether a cache hit or miss occurs. Otherwise, if a cache hit for an instruction fetch or a normal load occurs and a data EDC error is detected on any portion of the accessed data, a parity error is also signaled.

Store hits to the D-Cache will be placed into one of the RMW buffers to support EDC checkbit operation. If the store is a partial-width store, the cache data word corresponding to the address of the store is read into the next available buffer, the store data is merged, and the new EDC checkbits for the word are calculated. If the store is a full-width store, no read of the cache data is performed, since the store will overwrite the full EDC data granularity, and the EDC checkbits can be computed directly. Buffers are managed on a FIFO basis. If no buffer is available to hold the store data, a stall is incurred while a buffer (or two if possible) is written back to the cache to be freed for holding the new store. Buffers are written back to the D-Cache during otherwise idle cycles when two buffers are occupied, providing a simple form of store gathering.

Signaling of an ECC error or EDC error may cause a Machine Check exception to occur. One or more syndrome bits may be set in the Machine Check Syndrome register, or may instead result in a correction/auto-invalidation operation and not result in an exception being signaled. Both may occur, depending on the error action control setting in the appropriate cache control register.

12.6.9.1 Cache Line Lockout

In addition to the ECC/EDC protection employed for the caches, each cache line in the I-Cache and D-Cache has a lockout indicator composed of a redundant set of lockout bits. These lockout bits can be selectively set when certain errors occur in either the tag or data portion of the cache line. Use of the lockout function and control over error conditions which cause a lockout to occur are controlled by $L1CSR\{0,1\}_{[D,I]CLOC}$. When the lockout function is enabled, lines that encounter selected tag ECC or data EDC errors on normal instruction fetch, load, or store accesses will have their lockout bits set. When the lockout indicator is set, the line will not be replaced and will remain in an invalid state, effectively disabling it. Also, future tag ECC or data EDC errors on the line are ignored. Cache-inhibited accesses will only set lockout indicators on lines not in a locked way. In addition, no lockout indicators are set by cache-inhibited accesses when operating in machine check mode.

When operating in machine check mode, if lockout controls are enabled via $L1CSR\{0,1\}_{[D,I]CLOC}$, lockout bit parity errors on any line detected on a cacheable cache lookup will generate a machine check exception.

If correction/auto-invalidation is instead enabled, on each cache lookup operation for an instruction fetch or normal load or store access, if a single- or double-bit lockout bit parity error is detected in one or more ways and lockout controls are enabled, the lockout error(s) will be corrected by rewriting all lockout bits to the asserted state, and no machine check is generated, unless the line is in a locked way. If a line in a locked way incurs a LO bit parity error, a machine check will be generated to ensure that any possible new lockout of the line is reported. For non-cacheable accesses, lockout bit parity errors will only be corrected for lines not in a locked way. Lockout bit parity errors do not generate a machine check for non-cacheable accesses in either error action mode, thus lockout bit correction is not performed for lockout bit parity errors detected on a line in a locked way.

In addition, to avoid certain exception conditions and for consistency with error reporting, specialized load/store accesses do not set LO bits; instead, cache contents are ignored for lines with those errors. Cache flush and invalidate instructions do not report or correct lockout bit parity errors. For both of these cases, a future cacheable normal access will perform the lockout function if required.

12.6.10 Cache Error Injection

Cache error injection provides a way to test error recovery by intentionally injecting parity errors into the instruction and/or data cache.

Error injection into the instruction cache operates as follows:

- If $L1CSR1_{ICEI}$ is set, any instruction cache line fill to the instruction cache data has the associated two most significant parity check bits inverted in the instruction cache data array for each doubleword loaded.

Error injection for the data cache operates as follows:

- If $L1CSR0_{DCEI}$ is set, any cache line fill to the data cache data array has the associated two most significant parity check bits inverted in the data array for each word loaded.

Additionally, inverted parity bits are generated for any data stored into the data cache data array on a store hit.

Cache parity error injection is not performed for cache debug write accesses, since parity bit values written can be directly controlled.

In order to clear the parity errors, a cache invalidation or an invalidation of the lines that could have had an injected parity error may be performed. Line invalidation may be performed by an **icbi/dcbi** instruction, or an L1FINV{0,1} invalidation operation.

12.7 Exceptions

Interrupts implemented in e200z420n3 and the exception conditions that cause them are listed in *Table 157*.

Table 157. Exceptions and Conditions

Interrupt type	Interrupt vector offset value	Causing conditions
System reset	none, vector to [p_rstbase[0:29]] || 2'b00	Reset
Critical Input	0x00(1)	**p_critint_b** is asserted and MSR_{CE} = 1.
Machine check	0x10	1) **p_mcp_b** transitions from negated to asserted 2)ISI or Bus Error on first instruction fetch for an exception handler 3)Parity Error signaled on Cache access 4)Parity Error signaled on Local Memory access 5)External bus error 6)Stack limit check failure
Machine check (NMI)	0x10	Non-Maskable Interrupt
Data Storage	0x20	Access control
Instruction Storage	0x30	Access control
External Input	0x40(2)	Interrupt Controller interrupt and MSR_{EE} = 1
Alignment	0x50	1) **lmw**, **stmw** not word aligned 2) **lwarx** or **stwcx.** not word aligned, **lharx** or **sthcx.** not halfword aligned 3) 4) dcbz
Program	0x60	Illegal, Privileged, Trap
Performance Monitor	0x70	Performance Monitor Enabled Condition or Event w/$PMGC0_{UDI}$ = 0
System call	0x80	Execution of the System Call (**se_sc**) instruction
Debug	0x90	Trap, Instruction Address Compare, Data Address Compare, Instruction Complete, Branch Taken, Return from Interrupt, Interrupt Taken, Debug Counter, External Debug Event, Unconditional Debug Event, Performance Monitor Enabled Condition or Event w/$PMGC0_{UDI}$ = 1, MPU

Table 157. Exceptions and Conditions (continued)

Interrupt type	Interrupt vector offset value	Causing conditions
EFPU Data Exception	0xA0	See core specification about Embedded Floating-point Data Exception
EFPU Round Exception	0xB0	See core specification about Embedded Floating-point Round Exception
TBD	0xC0–0xF0	Reserved for future processor use

1. Autovectored Critical Input interrupts use this offset value. Vectored interrupts supply an interrupt vector offset directly.
2. Autovectored External Input interrupts use this offset value. Vectored interrupts supply an interrupt vector offset directly.

12.7.1 Exception Syndrome Register (ESR)

The Exception Syndrome Register (ESR) provides a *syndrome* to differentiate between exceptions that can generate the same interrupt type.

Offset: Refer to *Figure 64* (SPR - 62) Access: Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0				PIL	PPR	PTR	FP	ST	0			0	0	BO	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0								SPV	0	VLEMI	0			0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 75. Exception Syndrome Register (ESR)

The ESR bits are defined in *Table 158*.

Table 158. ESR fields description

Field	Description	Associated interrupt type
4 PIL	Illegal Instruction exception For Core (e200z225n3), PIL is used for both illegal and unimplemented instructions.	Program
5 PPR	Privileged Instruction exception	Program
6 PTR	Trap exception	Program
7 FP	Floating-point operation	Program
8 ST	Store operation	Alignment Data Storage

Table 158. ESR fields description (continued)

Field	Description	Associated interrupt type
14 BO	Byte Ordering exception Mismatched Instruction Storage exception (Not used by z4203n3. Implemented but not set by hardware.)	Instruction Storage
24 SPV	EFPU APU Operation	EFPU Floating-point Data Exception EFPU Floating-point Round Exception Alignment Data Storage
26 VLEMI	VLE Mode Instruction	EFPU Floating-point Data Exception EFPU Floating-point Round Exception Data Storage Instruction Storage Alignment Program System Call

12.7.2 Machine State Register (MSR)

The Machine State Register defines the state of the processor. The e200z420n3 MSR is shown in *Figure 76*.

Offset: Refer to *Figure 64* Access: Read/Write

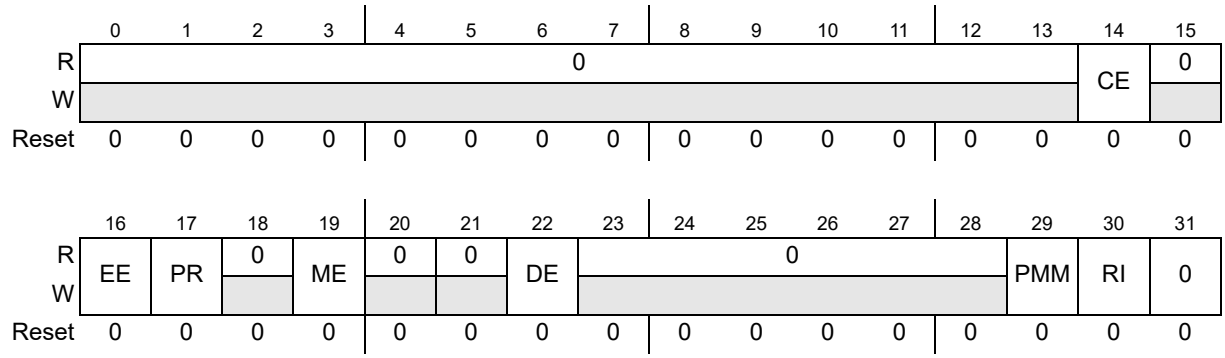


Figure 76. Machine State Register (MSR)

The MSR bits are defined in *Table 159*.

Table 159. MSR fields description

Field	Description
14 CE	Critical Interrupt Enable 0 Critical Input interrupts are disabled. 1 Critical Input interrupts are enabled.
16 EE	External Interrupt Enable 0 External Input interrupts are disabled. 1 External Input interrupts are enabled.
17 PR	Problem State 0 The processor is in supervisor mode, can execute any instruction, and can access any resource (For example: GPRs, SPRs, MSR, etc.). 1 The processor is in user mode, cannot execute any privileged instruction, and cannot access any privileged resource.
19 ME	Machine Check Enable 0 Asynchronous Machine Check interrupts are disabled. 1 Asynchronous Machine Check interrupts are enabled.
22 DE	Debug Interrupt Enable 0 Debug interrupts are disabled. 1 Debug interrupts are enabled.
29 PMM	PMM Performance monitor mark bit System software can set PMM when a marked process is running to enable statistics to be gathered only during the execution of the marked process. MSR_{PR} and MSR_{PMM} together define a state that the processor (supervisor or user) and the process (marked or unmarked) may be in at any time. If this state matches an individual state specified in the Performance Monitor registers PMLCa n, the state for which monitoring is enabled, counting is enabled.
30 RI	Recoverable Interrupt This bit is provided for software use to detect nested machine check exception conditions. This bit is cleared by hardware when a Machine Check interrupt is taken

12.7.3 Machine Check Syndrome Register (MCSR)

When the processor takes a machine check interrupt, it updates the Machine Check Syndrome Register (MCSR) to differentiate between machine check conditions. The MCSR is shown in *Figure 77*.

Offset: Refer to *Figure 64* (SPR - 572) Access: Read/Clear

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	MCP	IC_DPERR	0	DC_DPERR	EXCP_ERR	IC_TPERR	DC_TPERR	IC_LKERR	DC_LKERR	0		NMI	MAV	MEA	U	IF
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	LD	ST	G	0		STACK_ERR	IMEM_PERR	DMEM_RDPERR	DMEM_WRPERR	0		BUS_IRERR	BUS_DRERR	BUS_WRERR	BUS_WRDSI	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 77. Machine Check Syndrome Register (MCSR)

Table 160 describes MCSR fields. The MCSR indicates the source of a machine check condition.

All bits in the MCSR are implemented as “write 1 to clear.” Software in the machine check handler is expected to clear the MCSR bits it has sampled prior to re-enabling MSR_{ME} to avoid a redundant machine check exception and to prepare for updated status bit information on the next machine check interrupt.

Note that any set bit in the MCSR other than status-type bits will cause a subsequent machine check interrupt once MSR_{ME} = 1.

Table 160. Machine Check Syndrome Register (MCSR) fields description

Field	Description	Exception type(1)	Recoverable
0 MCP	Machine check input pin	Async Mchk	Maybe
1 IC_DPERR	Instruction Cache data array parity error	Async Mchk	Precise
3 DC_DPERR	Data Cache data array parity error	Async Mchk	Maybe
4 EXCP_ERR	ISI or Bus Error on first instruction fetch for an exception handler	Async Mchk	Precise
5 IC_TPERR	Instruction Cache Tag parity error	Async Mchk	Precise
6 DC_TPERR	Data Cache Tag parity error	Async Mchk	Maybe

Table 160. Machine Check Syndrome Register (MCSR) fields description (continued)

Field	Description	Exception type(1)	Recoverable
7 IC_LKERR	Instruction Cache Lock error Indicates a cache control operation or invalidation operation invalidated one or more lines in a locked way of the I-Cache for certain situations. May also be set on locked line refill error.	Async Mchk	—
8 DC_LKERR	Data Cache Lock error Indicates a cache control operation or invalidation operation invalidated one or more lines in a locked way of the D-Cache for certain situations. May also be set on locked line refill error.	Async Mchk	—
11 NMI	NMI input pin	NMI	—
12 MAV	MCAR Address Valid Indicates that the address contained in the MCAR was updated by hardware to correspond to the first detected Async Mchk error condition	Status	—
13 MEA	MCAR holds Effective Address If MAV = 1,MEA = 1 indicates that the MCAR contains an effective address and MEA = 0 indicates that the MCAR contains a physical address(2)	Status	—
14 U	User Indicates the value captured in MCAR was generated in user mode.	Status	—
15 IF	Instruction Fetch Error Report An error occurred during the fetch of an instruction and the instruction attempted to execute. This could be due to an internal parity error, or an external bus error. MCSRR0 contains the instruction address.	Error Report	Precise
16 LD	Load type instruction Error Report An error occurred during the attempt to execute the load type instruction located at the address stored in MCSRR0. This could be due to an internal parity error, stack limit check error, or an external bus error.	Error Report	Precise
17 ST	Store type instruction Error Report An error occurred during the attempt to execute the store type instruction located at the address stored in MCSRR0. This could be due to an internal parity error, stack limit check error, or on certain external bus errors.	Error Report	Precise

Table 160. Machine Check Syndrome Register (MCSR) fields description (continued)

Field	Description	Exception type(1)	Recoverable
18 G	Guarded instruction Error Report An error occurred during the attempt to execute the load or store type instruction located at the address stored in MCSRR0 and the access was guarded and encountered an error on the external bus, or an uncorrectable DMEM error.	Error Report	Precise
21 STACK_ERR	Stack Access Limit Check Error Indicates a limit check failure on a CPU data access using R1 in the <EA> calculation.	Async Mchk	Precise
22 IMEM_PERR	Instruction Mem (IMEM) Parity Error Indicates an uncorrectable error in the IMEM on a CPU port access.	Async Mchk	Precise
23 DMEM_RDPERR	Data Mem (DMEM) Parity Error Indicates an uncorrectable error in the DMEM on a CPU port read access.	Async Mchk	Precise
24 DMEM_WRPERR	Data Mem (DMEM) Write Parity Error Indicates an uncorrectable error in the DMEM on a CPU port write access.	Async Mchk	Maybe
27 BUS_IRERR	Read bus error on Instruction recovery linefill	Async Mchk	Precise if data used
28 BUS_DRERR	Read bus error on data load or linefill	Async Mchk	Precise if data used
29 BUS_WRERR	Write bus error on store to bus or DMEM imprecise write error	Async Mchk	Unlikely
30 BUS_WRDSI	Write bus error on buffered store to bus with DSI signaled. Set concurrently with BUS_WRERR for this case	Async Mchk	Unlikely

1. The Exception Type indicates the exception type associated with a given syndrome bit.

   - “Error Report” indicates that this bit is only set for error report exceptions which cause machine check interrupts. These bits are only updated when the machine check interrupt is actually taken. Error report exceptions are not gated by MSR_{ME}. These are synchronous exceptions. These bits will remain set until cleared by software writing a ‘1’ to the bit position(s) to be cleared.

   - “Status” indicates that this bit is provides additional status information regarding the logging of a machine check exception. These bits will remain set until cleared by software writing a ‘1’ to the bit position(s) to be cleared.

   - “NMI” indicates that this bit is only set for the non-maskable interrupt type exception which causes a machine check interrupt. This bit is only updated when the machine check interrupt is actually taken. NMI exceptions are not gated by MSR_{ME}. This is an asynchronous exception. This bit will remain set until cleared by software writing a ‘1’ to the bit position.

   - “Async Mchk” indicates that this bit is set for an asynchronous machine check exception. These bits are set immediately upon detection of the error. Once any “Async Mchk” bit is set in the MCSR, a machine check interrupt will occur if $MSR_{ME} = 1$. If $MSR_{ME} = 0$, the machine check exception will remain pending. These bits will remain set until cleared by software writing a ‘1’ to the bit position(s) to be cleared.

2. Note that since no address translation mechanism is present on Zen Z420n3, the MEA value is always set to `0'.

12.7.4 Interrupt Vector Prefix Registers (IVPR)

The Interrupt Vector Prefix Register is used during interrupt processing for determining the starting address of a software handler used to handle an interrupt. The value of the Vector Offset selected for a particular interrupt type is concatenated with the Vector Base value held in the IVPR to form an instruction address from which execution is to begin. The format of IVPR is shown in *Figure 78*.

Offset: Refer to *Figure 64* (SPR - 63) Access: Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	Vector Base[0:15]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	Vector Base[16:23]								0							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 78. Zen Interrupt Vector Prefix Register (IVPR)

The IVPR fields are defined in *Table 161*.

Table 161. IVPR fields description

Field	Description
0:23 Vector Base	Vector Base This field is used to define the base location of the vector table, aligned to a 256 byte boundary. This field provides the high-order 24 bits of the location of all interrupt handlers (unless hardware vectoring is used). The vector offset value appropriate for the type of exception being processed is concatenated with the IVPR Vector Base to form the address of the handler in memory. For hardware-vectored interrupts, the value of the supplied interrupt vector is logically OR'ed with the low order bits of the Vec Base Field to determine the interrupt handler address.

12.7.5 Interrupt Definitions

12.7.5.1 Critical Input Interrupt (offset 0x00)

A Critical Input exception is signalled to the processor by the assertion of the critical interrupt pin. If the exception is enabled by MSR_{CE}, the Critical Input interrupt is taken.

A Critical Input interrupt may be delayed by other higher priority exceptions or if MSR_{CE} is cleared when the exception occurs.

Table 162 lists register settings when a Critical Input interrupt is taken.

Table 162. Critical Input Interrupt: Register Settings

Register	Setting description					
CSRR0	Set to the effective address of the instruction that the processor would have attempted to execute next if no exception conditions were present.					
CSRR1	Set to the contents of the MSR at the time of the interrupt					
MSR	SPV	0	FP	0	FE1	0
	WE	0	ME	—	IS	0
	CE	0	FE0	0	DS	0
	EE	0	DE	—/0(1)	PMM	0
	PR	0			RI	—
ESR	Unchanged					
MCSR	Unchanged					
DEAR	Unchanged					
Vector	$IVPR_{0:23}$ || 0x00 (autovectored) $IVPR_{0:15}$ || ($IVPR_{16:29}$ | **p_voffset[0:13])** || 2'b00 (non-autovectored)					

1. Clearing of DE is optionally supported by control in HID0.

The MSR_{DE} bit is not automatically cleared by a Critical Input interrupt, but it can be configured to be cleared via the HID0 register ($HID0_{CICLRDE}$).

12.7.5.2 Machine Check Interrupt (offset 0x10)

The EIS Machine Check APU defines a separate set of save/restore registers (MCSRR0/1), a Machine Check Syndrome Register (MCSR) to record the source(s) of machine checks, and a Machine Check Address Register (MCAR) to hold an address associated with a machine check for certain classes of machine checks. Return from Machine Check instructions (**se_rfmci**) are also provided to support returns using MCSRR0/1.

The MSR_{DE} bit is not automatically cleared by a Machine Check exception, but it can be configured to be cleared or left unchanged via the HID0 register ($HID0_{MCCLRDE}$).

When a Machine Check interrupt is taken, registers are updated as shown in *Table 163*.

Table 163. Machine Check Interrupt: Register Settings

Register	Setting description					
MCSRR0	On a best-effort basis, set to the address of some instruction that was executing or about to be executing when the machine check condition occurred.					
MCSRR1	Set to the contents of the MSR at the time of the interrupt					
MSR	SPV	0	FP	0	FE	0
	WE	0	ME	0	IS	0
	CE	0	FE0	0	DS	0
	EE	0	DE	0/—(1)	PMM	0
	PR	0			RI	0

Table 163. Machine Check Interrupt: Register Settings (continued)

Register	Setting description
ESR	Unchanged
MCSR	Updated to reflect the source(s) of a machine check. Hardware only sets appropriate bits, no previously set bits are cleared by hardware.
Vector	$IVPR_{0:23}$ || 0x10

1. Clearing of DE is optionally supported by control in HID0.

12.7.5.3 Data Storage Interrupt (offset 0x20)

A Data Storage interrupt (DSI) may occur if no higher priority exception exists and a Read or Write Access Control exception condition exists.

Table 164 lists register settings when a DSI is taken.

Table 164. Data Storage Interrupt: Register Settings

Register	Setting description					
SRR0	Set to the effective address of the excepting load/store instruction.					
SRR1	Set to the contents of the MSR at the time of the interrupt					
MSR	SPV	0	FP	0	FE1	0
	WE	0	ME	—	IS	0
	CE	—	FE0	0	DS	0
	EE	0	DE	—	PMM	0
	PR	0			RI	—
ESR	Access:		[ST], [SPV], VLEMI. All other bits cleared.			
MCSR	Unchanged					
DEAR	For Access Control exceptions, set to the effective address of the access which caused the violation.					
Vector	$IVPR_{0:23}$ || 0x20					

12.7.5.4 Instruction Storage Interrupt (offset 0x30)

An Instruction Storage interrupt (ISI) occurs when no higher priority exception exists and an Execute Access Control exception occurs.

Table 165 lists register settings when an ISI is taken.

Table 165. Instruction Storage Interrupt: Register Settings

Register	Setting description
SRR0	Set to the effective address of the excepting instruction.
SRR1	Set to the contents of the MSR at the time of the interrupt

Table 165. Instruction Storage Interrupt: Register Settings (continued)

Register	Setting description					
MSR	SPV	0	FP0	0	FE1	0
	WE	0	ME	—	IS	0
	CE	—	FE0	0	DS	0
	EE	0	DE	—	PMM	0
	PR	0			RI	—
ESR	VLEMI. All other bits cleared.					
MCSR	Unchanged					
DEAR	Unchanged					
Vector	$IVPR_{0:23}$ || 0x30					

12.7.5.5 External Input Interrupt (offset 0x40)

An External Input exception is signalled to the processor by the assertion of an interrupt from the interrupt controller. The input is a level-sensitive signal expected to remain asserted until the core acknowledges the external interrupt. If the input is negated early, recognition of the interrupt request is not guaranteed. When the core detects the exception, if the exception is enabled by MSR_{EE}, it takes the External Input interrupt.

An External Input interrupt may be delayed by other higher priority exceptions or if MSR_{EE} is cleared when the exception occurs.

Table 166 lists register settings when an External Input interrupt is taken.

Table 166. External Input Interrupt: Register Settings

Register	Setting description					
SRR0	Set to the effective address of the instruction that the processor would have attempted to execute next if no exception conditions were present.					
SRR1	Set to the contents of the MSR at the time of the interrupt					
MSR	SPV	0	FP	0	FE1	0
	WE	0	ME	—	IS	0
	CE	—	FE0	0	DS	0
	EE	0	DE	—	PMM	0
	PR	0	—	—	RI	—
ESR	Unchanged					
MCSR	Unchanged					
DEAR	Unchanged					
Vector	$IVPR_{0:23}$ || 0x40 (autovectored) IVPR0:15 || (IVPR16:29 | **p_voffset[0:13]**) || 2'b00 (non-autovectored)					

12.7.5.6 Alignment Interrupt (offset 0x50)

An Alignment exception is generated when any of the following occurs:

- The operand of **lmw** or **stmw** is not word aligned.
- The operand of **lwarx** or **stwcx.** is not word aligned.
- The operand of **lharx** or **sthcx.** is not halfword aligned.
- Execution of a **dcbz** instruction is attempted.

Table 167 lists register settings when an alignment interrupt is taken.

Table 167. Alignment Interrupt: Register Settings

Register	Setting description					
SRR0	Set to the effective address of the excepting load/store/dcbz instruction.					
SRR1	Set to the contents of the MSR at the time of the interrupt					
MSR	SPV	0	FP	0	FE1	0
	WE	0	ME	—	IS	0
	CE	—	FE0	0	DS	0
	EE	0	DE	—	PMM	0
	PR	0	—	—	RI	—
ESR	[ST], VLEMI. All other bits cleared.					
MCSR	Unchanged					
DEAR	Set to the effective address of a byte of the load or store access causing the violation.					
Vector	$IVPR_{0:23}$ || 0x50					

12.7.5.7 Program Interrupt (offset 0x60)

A program interrupt occurs when no higher priority exception exists and one or more of the following exception conditions occur:

- Illegal Instruction exception
- Privileged Instruction exception
- Trap exception

The e200z420n3 will invoke an Illegal Instruction program exception on attempted execution of the following instructions:

- Unimplemented instructions
- An instruction from the illegal instruction class
- **mtspr** and **mfspr** instructions with an undefined SPR specified
- **mtdcr** and **mfdcr** instructions with an undefined DCR specified

The e200z420n3 will invoke a Privileged Instruction program exception on attempted execution of the following instructions when MSR_{PR} = 1 (user mode):

- A privileged instruction
- **mtspr** and **mfspr** instructions that specify a SPRN value with $SPRN_5$ = 1 (even if the SPR is undefined)

The e200z420n3 will invoke an Trap exception on execution of the **tw** instruction if the trap conditions are met and the exception is not also enabled as a Debug interrupt.

The core will invoke an Illegal instruction program exception on attempted execution of the instructions **lswi**, **lswx**, **stswi**, **stswx**, **mfapidi**, **mfdcrx**, **mtdcrx**, or on any <asisitalic>PowerISA 2.06 floating-point instruction. All other defined or allocated instructions that are not implemented by core will cause a illegal instruction program exception.

Table 168 lists register settings when a Program interrupt is taken.

Table 168. Program Interrupt: Register Settings

Register	Setting description					
SRR0	Set to the effective address of the excepting instruction.					
SRR1	Set to the contents of the MSR at the time of the interrupt					
MSR	SPV	0	FP	0	FE1	0
	WE	0	ME	—	IS	0
	CE	—	FE0	0	DS	0
	EE	0	DE	—	PMM	0
	PR	0			RI	—
ESR	Illegal, Unimplemented: Privileged: Trap:		PIL, VLEMI. All other bits cleared. PPR, VLEMI. All other bits cleared. PTR, VLEMI. All other bits cleared.			
MCSR	Unchanged					
DEAR	Unchanged					
Vector	$IVPR_{0:23}$ || 0x60					

12.7.5.8 Performance Monitor Interrupt (Offset 0x70)

A performance monitor interrupt that may be generated by an enabled condition or event. An enabled condition or event is as follows:

A PMC*x* register overflow condition occurs with the following settings:

- PMLCax_{CE} = 11; that is, for the given counter the overflow condition is enabled.
- PMCx_{OV} = 11; that is, the given counter indicates an overflow.

For a performance monitor interrupt to be signaled on an enabled condition or event, $PMGC0_{PMIE}$ must be set.

Although an exception condition may occur with MSR_{EE} = 0, the interrupt cannot be taken until MSR_{EE} = 1.

The priority of the performance monitor interrupt is below all other asynchronous interrupts.

Table 169 lists register settings when an performance monitor interrupt is taken.

Table 169. Performance Monitor Interrupt: Register Settings

Register	Setting description					
SRR0/ DSRR0(1)	Set to the effective address of the next instruction to be executed.					
SRR1/ DSRR1(1)	Set to the contents of the MSR at the time of the interrupt.					
MSR	SPV	0	FP	0	FE1	0
	WE	0	ME	—	IS	0
	CE	—/0(2)	FE0	0	DS	0
	EE	0/—(3)	DE	—/0(4)	PMM	0
	PR	0	—		RI	—
ESR	Unchanged					
MCSR	Unchanged					
DEAR	Unchanged					
Vector	$IVPR_{0:23}$ || 0x70					

1. DSRR0/1 are used if $PMGC0_{UDI} = 1$
2. CE is cleared if $PMGC0_{UDI} = 1$ and $HID0_{DCLRCE} = 1$
3. EE is not cleared if $PMGC0_{UDI} = 1$ and $HID0_{DCLREE} = 0$
4. DE is cleared if $PMGC0_{UDI} = 1$

12.7.5.9 System Call Interrupt (offset 0x80)

A System Call interrupt occurs when a System Call (**se_sc)** instruction is executed and no higher priority exception exists.

Table 170 lists register settings when a System Call interrupt is taken.

Table 170. System Call Interrupt: Register Settings

Register	Setting description					
SRR0	Set to the effective address of the instruction *following* the system call instruction.					
SRR1	Set to the contents of the MSR at the time of the interrupt					
MSR	SPV	0	FP	0	FE1	0
	WE	0	ME	—	IS	0
	CE	—	FE0	0	DS	0
	EE	0	DE	—	PMM	0
	PR	0	—	—	RI	—
ESR	VLEMI. All other bits cleared.					
MCSR	Unchanged					
DEAR	Unchanged					
Vector	$IVPR_{0:23}$ || 0x80					

12.7.5.10 Debug Interrupt (offset 0x90)

There are multiple sources that can signal a Debug exception. A Debug interrupt occurs when no higher priority exception exists, a Debug exception exists in the Debug Status Register, and Debug interrupts are enabled (both $DBCR0_{IDM}$ = 1 (internal debug mode) and MSR_{DE} = 1).

Table 171 lists register settings when a Debug interrupt is taken.

Table 171. Debug Interrupt: Register Settings

<table>
<tr><th>Register</th><th colspan="6">Setting description</th></tr>
<tr><td>DSRR0(1)</td><td colspan="6">Set to the effective address of the excepting instruction for IAC, BRT, RET, CRET, and TRAP.
Set to the effective address of the next instruction to be executed following the excepting instruction for DAC (usually) and ICMP.
For a UDE, IRPT, CIRPT, DCNT, or DEVT type exception, set to the effective address of the instruction that the processor would have attempted to execute next if no exception conditions were present.</td></tr>
<tr><td>DSRR1</td><td colspan="6">Set to the contents of the MSR at the time of the interrupt</td></tr>
<tr><td rowspan="5">MSR</td><td>SPV</td><td>0</td><td>FP</td><td>0</td><td>FE1</td><td>0</td></tr>
<tr><td>WE</td><td>0</td><td>ME</td><td>—</td><td>IS</td><td>0</td></tr>
<tr><td>CE</td><td>—/0(2)</td><td>FE0</td><td>0</td><td>DS</td><td>0</td></tr>
<tr><td>EE</td><td>—/0(2)</td><td>DE</td><td>0</td><td>PMM</td><td>0</td></tr>
<tr><td>PR</td><td>0</td><td colspan="2">—</td><td>RI</td><td>—</td></tr>
<tr><td>DBSR(3)</td><td colspan="3">Unconditional Debug Event:
Instr. Complete Debug Event:
Branch Taken Debug Event:
Interrupt Taken Debug Event:
Critical Interrupt Taken Debug Event:
Trap Instruction Debug Event:
Instruction Address Compare:
Data Address Compare:
Debug Notify Interrupt:
Return Debug Event:
Critical Return Debug Event:
External Debug Event:
Performance Monitor Debug Event:
MPU Debug Event:
and optionally, an
Imprecise Debug Event flag</td><td colspan="3">UDE
ICMP
BRT
IRPT
CIRPT
TRAP
{IAC}
{DACR, DACW}
DNI
RET
CRET
{DEVT1, DEVT2}
PMI
MPU

{IDE}</td></tr>
<tr><td>ESR</td><td colspan="6">Unchanged</td></tr>
<tr><td>MCSR</td><td colspan="6">Unchanged</td></tr>
<tr><td>DEAR</td><td colspan="6">Unchanged</td></tr>
<tr><td>Vector</td><td colspan="6">$IVPR_{0:23}$ || 0x90</td></tr>
</table>

1. Assumes that the Debug interrupt is precise
2. Conditional based on control bits in HID0

3. Note that multiple DBSR bits may be set

12.7.5.11 Embedded Floating-point Data Interrupt (Offset 0xA0)

The Embedded Floating-point Data interrupt is taken if no higher priority exception exists and an EFPU Floating-point Data exception is generated. When a Floating-point Data exception occurs, the processor suppresses execution of the instruction causing the exception.

Table 172 lists register settings when a EFPU Floating-point Data interrupt is taken.

Table 172. Embedded Floating-point Data Interrupt: Register Settings

Register	Setting description					
SRR0	Set to the effective address of the excepting EFPU instruction.					
SRR1	Set to the contents of the MSR at the time of the interrupt					
MSR	SPV	0	FP	0	FE1	0
	WE	0	ME	—	IS	0
	CE	—	FE0	0	DS	0
	EE	0	DE	—	PMM	0
	PR	0	—		RI	—
ESR	SPV, VLEMI. All other bits cleared.					
MCSR	Unchanged					
DEAR	Unchanged					
Vector	$IVPR_{0:23}$ || 0xA0					

12.7.5.12 Embedded Floating-point Round Interrupt (Offset 0xB0)

The Embedded Floating-point Round interrupt is taken when an EFPU floating-point instruction generates an inexact result and inexact exceptions are enabled.

Table 173 lists register settings when a EFPU Floating-point Round interrupt is taken.

Table 173. Embedded Floating-point Round Interrupt: Register Settings

Register	Setting description					
SRR0	Set to the effective address of the instruction following the excepting EFPU instruction.					
SRR1	Set to the contents of the MSR at the time of the interrupt					
MSR	SPV	0	FP	0	FE1	0
	WE	0	ME	—	IS	0
	CE	—	FE0	0	DS	0
	EE	0	DE	—	PMM	0
	PR	0	—		RI	—
ESR	SPV, VLEMI. All other bits cleared.					
MCSR	Unchanged					

Table 173. Embedded Floating-point Round Interrupt: Register Settings (continued)

Register	Setting description
DEAR	Unchanged
Vector	$IVPR_{0:23}$ || 0xB0

12.7.5.13 System Reset Interrupt

The System Reset exception is a non-maskable, asynchronous exception signalled to the processor through the assertion of system-defined signals.

A System Reset may be initiated by either a software reset or during power-on reset.

When a reset request occurs, the processor branches to the system reset exception vector without attempting to reach a recoverable state. If reset occurs during normal operation, all operations cease and the machine state is lost.

Table 174 lists register settings when a System Reset interrupt is taken.

Table 174. System Reset Interrupt: Register Settings

Register	Setting description					
CSRR0	Undefined					
CSRR1	Undefined					
MSR	SPV	0	FP	0	FE1	0
	WE	0	ME	0	IS	0
	CE	0	FE0	0	DS	0
	EE	0	DE	0	PMM	0
	PR	0	—		RI	0
ESR	Cleared					
DEAR	Undefined					
Vector	**[p_rstbase[0:29]]** || 2'b00					

12.8 Memory Protection Unit (MPU)

12.8.1 MPU Overview

The e200z420n3 Memory Protection Unit (MPU) protects regions of memory, with the following feature set:

- 24-entry region descriptor table with support for 6 arbitrary-sized instruction memory regions, 12 arbitrary-sized data memory regions, and 6 additional arbitrary-sized regions programmable as instruction or data memory regions
- Ability to set access permissions and memory attributes on a per-region basis
- Process ID aware, with per-bit masking of TID values
- Capability for masking upper address bits in the range comparison
- Capability of bypassing permissions checking for selected access types
- Per-entry write-once logic for entry protection
- Hardware flash invalidation support and per-entry invalidation protection controls
- Ability to optionally utilize region descriptors for generating debug events and watchpoints
- Software managed by **mpure** and **mpuwe** instructions

12.8.2 Access Protection Operation

The Zen z420n3 Memory Protection Unit (MPU) provides access protection to memory regions using a set of region descriptors contained in a table structure. Each region descriptor defines an address range and a set of access protections and memory attributes. Individual region descriptors monitor either instruction accesses or data accesses. Independent access protection may be configured for supervisor-mode and user-mode reads, writes, and instruction accesses. In addition, any subset of these access types can bypass the protection system and instead rely on an external system MPU for enforcing access protections. This allows for an increased number of regions to be supported in a SoC, without requiring excessive processor resources.

Access addresses which fall within the address range(s) of one or more enabled region descriptors (a match) are allowed to proceed if the access permissions for any matching descriptor are set to allow the access. If no matching region descriptor is found, the access will not be allowed, and an Instruction Storage Interrupt (ISI) or Data Storage Interrupt (DSI) will be generated. If more than one matching descriptor is found, the access permissions enforced are the least restrictive access permissions defined by the matching entries.

Allowed accesses are based only on an address compare of the first byte of data being accessed; no checking is performed that the data associated with a single access is entirely contained in an allowed region.

For accesses which are not configured to bypass the protection system, a matching region descriptor with access permissions set to allow the type of access is required for the access to proceed without an access permission violation. An access permission violation results in the MPU signaling a protection violation via a DSI or an ISI, and the access is aborted.

Memory attributes of Guarded (for data accesses only) and Cache-Inhibited are provided in each descriptor, and are applied to matching accesses which pass permissions checking. In addition to controlling cache behavior, these attributes are used to control certain aspects of access ordering and buffering, and also to drive external system interface access attributes.

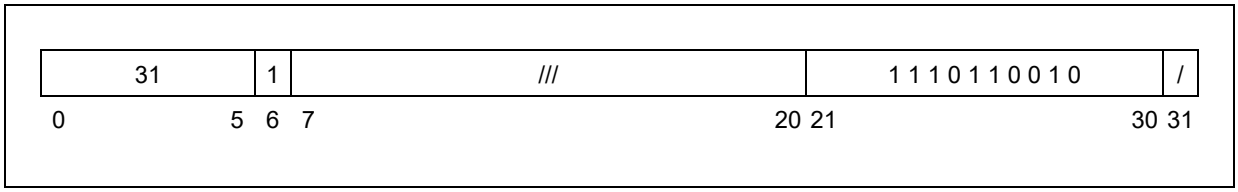
12.8.3 Software Interface and MPU Instructions

The MPU entries are accessed indirectly through several MPU Assist (MAS) registers. Software can write and read the MAS registers with **mtspr** and **mfspr** instructions. These registers contain information related to reading and writing a given entry within the MPU. Data is read from the MPU into the MAS registers with an **mpure** (MPU read entry) instruction. Data is written to the MPU from the MAS registers with an **mpuwe** (MPU write entry) instruction.

The **mpure**, **mpuwe**, and **mpusync** instructions are privileged.

12.8.4 MPU Read Entry Instruction (mpure)

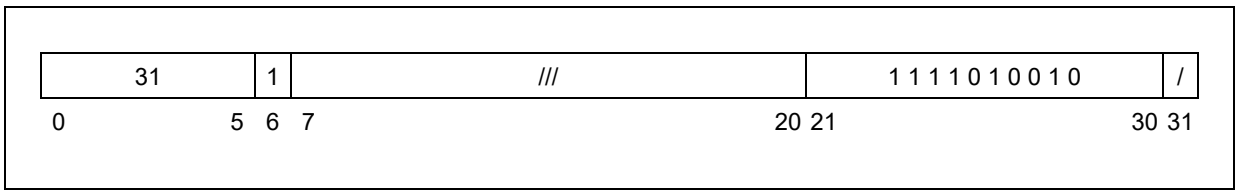
The MPU read entry instruction causes the content of a single MPU entry to be placed in the MPU assist registers. The entry is specified by the INST, SHD, and ESEL fields of the MAS0 register. The entry contents are placed in the MAS0, MAS1, MAS2, and MAS3 registers.




```
mpu_entry_id = MAS0(INST, SHD, ESEL)
result = MPU(mpu_entry_id)
MAS0, MAS1, MAS2, MAS3 = result
```

12.8.5 MPU Write Entry Instruction (mpuwe)

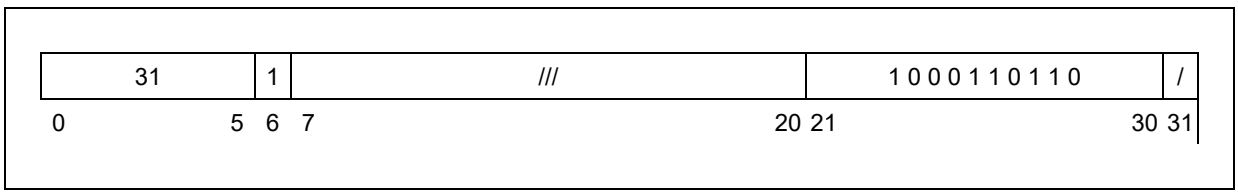
The MPU write entry instruction causes the contents of certain fields within the MPU assist registers MAS0, MAS1, MAS2, and MAS3 to be written into a single MPU entry in the MPU. The entry written is specified by the INST, SHD, and ESEL fields of the MAS0 register.




```
mpu_entry_id = MAS0(INST, SHD, ESEL)
MPU(mpu_entry_id) = MAS0, MAS1, MAS2, MAS3
```

12.8.6 MPU Synchronize Instruction (mpusync)

The MPU Synchronize instruction is treated as a privileged no-op by the e200z420n3.



12.8.7 MMU/MPU Configuration Register (MMUCFG)

The MMU/MPU Configuration Register (MMUCFG) is a 32-bit read-only register. The SPR number for MMUCFG is 1015 in decimal. MMUCFG provides information about the configuration of the e200z420n3 MPU design. The MMUCFG register is shown in *Figure 79*.

Offset: Refer to *Figure 64* (SPR - 1015) Access: Read Only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0								RASIZE							0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0					PIDSIZE					NMPUS		NTLBS		MAVN	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 79. MMU/MPU Configuration Register (MMUCFG)

The MMUCFG bits are described in *Table 175*.

Table 175. MMUCFG fields description

Field	Function
8:14 RASIZE	Number of Bits of Real Address supported 0100000 This version of the MPU implements 32 real address bits
21:25 PIDSIZE	PID Register Size 00111 PID registers contain 8 bits in this version of the MPU
26:27 NMPUS	Number of MPUs 01 This version of the MMU implements one MPU structure: a fully associative MPU for MPU0
28:29 NTLBS	Number of TLBs 00 This version of the MMU implements no TLB structures
30:31 MAVN	MMU Architecture Version Number 11 This version of the MMU implements Version 3.1 of the EIS MMU Architecture

12.8.8 MPU0 Configuration Register (MPU0CFG)

The MPU0 Configuration Register (MPU0CFG) is a 32-bit read-only register. The SPR number for MPU0CFG is 692 in decimal. MPU0CFG provides information about the configuration of MPU0 in the e200z420n3 MPU. The MPU0CFG register is shown in *Figure 80*.

Offset: Refer to *Figure 64* (SPR - 692) Access: Read Only

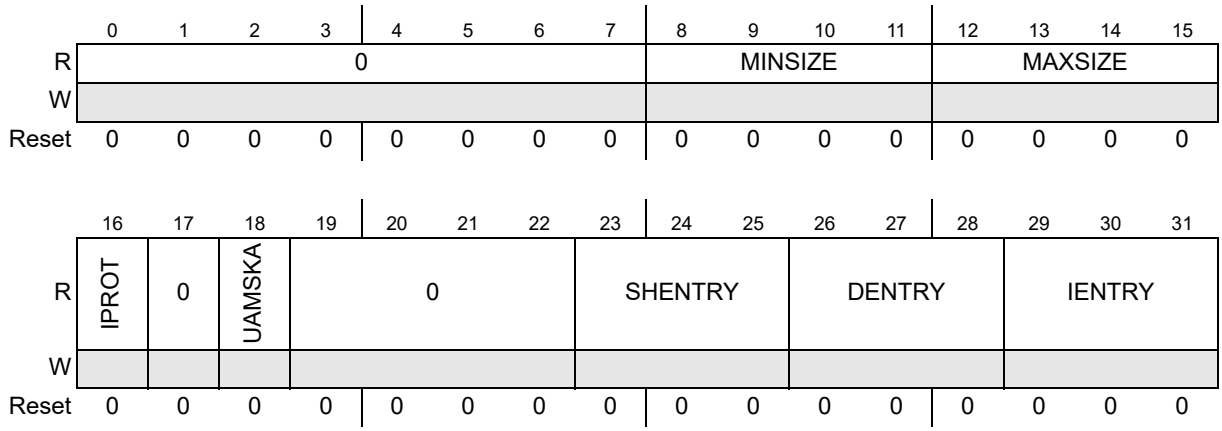


Figure 80. MPU0 Configuration Register (MPU0CFG)

The MPU0CFG bits are described in *Table 176*.

Table 176. MPU0CFG fields description

Field	Function
0 FASSOC	Fully Associative 1 Indicates that MPU0 is fully associative
1:7	Reserved for non-fully associative implementation use
8:11 MINSIZE	Minimum Region Size Due to the use of access address matching, the effective smallest region size is 8 bytes 0x0 Smallest region size is 1 byte
12:15 MAXSIZE	Maximum Region Size 0xB Largest region size is 4 GB
16 IPROT	Invalidate Protect Capability 1 Invalidate Protect Capability is supported in MPU0
18 UAMSKA	Upper Address Masking Availability 1 Upper Address Masking is Available
23:25 SHENTRY	Number of Shared (configurable for I or D) Entries 0x2 Indicates that MPU0 contains six shared entries
26:28 DENTRY	Number of Data Entries 0x4 Indicates that MPU0 contains 12 dedicated data entries
29:31 IENTRY	Number of Instruction Entries 0x2 Indicates that MPU0 contains six dedicated Instruction entries

12.8.9 MPU0 Control and Status Register 0 (MPU0CSR0)

The MPU0 Control and Status Register 0 (MPU0CSR0) is a 32-bit register. The SPR number for MPU0CSR0 is 1014 in decimal. MPU0CSR0 controls the operation of the MPU. The MPU0CSR0 register is shown in *Figure 81*.

Offset: Refer to *Figure 64* (SPR - 1014) Access: Read/Write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	BYPSR	BYPSW	BYPSX	BYPUR	BYPUW	BYPUX	0		DRDEN	DWDEN	IDEN	0	TIDCTL	0	MPUFI	MPUEN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 81. 32-bit register (REG NAME)

The MPU0CSR0 bits are described in *Table 177*.

Table 177. MPU0CSR0 fields description

Field	Description
16 BYPSR	Bypass MPU protections for Supervisor Read accesses This bit controls whether supervisor-mode read accesses bypass the MPU protection mechanism. When bypass is enabled, supervisor-mode read accesses do not generate a DSI when no MPU match occurs. Supervisor-mode read accesses are still compared to MPU entries, and a hit will provide access attributes (CI, G). 0 No Bypass of MPU protections for Supervisor Read accesses 1 Bypass MPU protections for Supervisor Read accesses
17 BYPSW	Bypass MPU protections for Supervisor Write accesses This bit controls whether supervisor-mode write accesses bypass the MPU protection mechanism. When bypass is enabled, supervisor-mode write accesses do not generate a DSI when no MPU match occurs. Supervisor-mode write accesses are still compared to MPU entries, and a hit will provide access attributes (CI, G). 0 No Bypass of MPU protections for Supervisor Write accesses 1 Bypass MPU protections for Supervisor Write accesses
18 BYPSX	Bypass MPU protections for Supervisor Instruction accesses This bit controls whether supervisor-mode instruction accesses bypass the MPU protection mechanism. When bypass is enabled, supervisor-mode instruction accesses do not generate an ISI when no MPU match occurs. Supervisor-mode instruction accesses are still compared to MPU entries, and a hit will provide access attributes (CI). 0 No Bypass of MPU protections for Supervisor Instruction accesses 1 Bypass MPU protections for Supervisor Instruction accesses
19 BYPUR	Bypass MPU protections for User Read accesses This bit controls whether user-mode read accesses bypass the MPU protection mechanism. When bypass is enabled, user-mode read accesses do not generate a DSI when no MPU match occurs. User-mode read accesses are still compared to MPU entries, and a hit will provide access attributes (CI, G). 0 No Bypass of MPU protections for User Read accesses 1 Bypass MPU protections for User Read accesses

Table 177. MPU0CSR0 fields description (continued)

Field	Description
20 BYPUW	Bypass MPU protections for User Write accesses This bit controls whether user-mode write accesses bypass the MPU protection mechanism. When bypass is enabled, user-mode write accesses do not generate a DSI when no MPU match occurs. User-mode write accesses are still compared to MPU entries, and a hit will provide access attributes (CI, G). 0 No Bypass of MPU protections for User Write accesses 1 Bypass MPU protections for User Write accesses
21 BYPUX	Bypass MPU protections for User Instruction accesses This bit controls whether user-mode instruction accesses bypass the MPU protection mechanism. When bypass is enabled, user-mode instruction accesses do not generate an ISI when no MPU match occurs. User-mode instruction accesses are still compared to MPU entries, and a hit will provide access attributes (CI). 0 No Bypass of MPU protections for User Instruction accesses 1 Bypass MPU protections for User Instruction accesses
24 DRDEN	Data Read Debug Enable This bit controls whether data read accesses are enabled to generate MPU debug events. When disabled, no data read access will generate an MPU debug event, regardless of whether an access match occurs to a valid MPU region descriptor with DEBUG = 1. If such a match occurs, no MPU debug event is generated, and no access protection is performed. When enabled, data read accesses are not blocked from generating MPU debug events. 0 MPU debug events are disabled for data read accesses 1 MPU debug events are enabled for data read accesses
25 DWDEN	Data Write Debug Enable This bit controls whether data write accesses are enabled to generate MPU debug events. When disabled, no data write access will generate an MPU debug event, regardless of whether an access match occurs to a valid MPU region descriptor with DEBUG = 1. If such a match occurs, no MPU debug event is generated, and no access protection is performed. When enabled, data write accesses are not blocked from generating MPU debug events. 0 MPU debug events are disabled for data write accesses 1 MPU debug events are enabled for data write accesses
26 IDEN	Instruction Debug Enable This bit controls whether instruction accesses are enabled to generate MPU debug events. When disabled, no instruction access will generate an MPU debug event, regardless of whether an access match occurs to a valid MPU region descriptor with DEBUG = 1. When enabled, instruction accesses are not blocked from generating MPU debug events. 0 MPU debug events are disabled for instruction accesses 1 MPU debug events are enabled for instruction accesses
28 TIDCTL	TID usage Control When TIDCTL is set to 1, the TID match is disabled (forced match) in Supervisor mode. 0 TID comparisons performed normally 1 No TID comparisons are performed for matching while in Supervisor mode

Table 177. MPU0CSR0 fields description (continued)

Field	Description
30 MPUFI	MPU flash invalidate When written to a 1, a MPU invalidation operation is initiated by hardware. Once complete, this bit is reset to 0. Writing a 1 while an invalidation operation is in progress will result in an undefined operation. Writing a 0 to this bit while an invalidation operation is in progress will be ignored. MPU invalidation operations require 3 cycles to complete. Note that entries which have the IPROT bit set will not be invalidated. 0 No flash invalidate 1 MPU1 invalidation operation
31 MPUEN	MPU Enable This bit enables operation of the MPU. When enabled, access addresses are compared to each entry in the MPU for a match condition. If no match condition occurs, and the access type is not enabled to bypass the MPU protections, then an ISI or DSI condition is signaled for the access. Note that this bit is ignored for matching entries with DEBUG = 1 if in EDM and hardware owns MPU Debug entries. 0 MPU is disabled 1 MPU is enabled

12.8.10 MPU Assist Registers (MAS)

The e200z420n3 uses four special-purpose registers (MAS0, MAS1, MAS2, and MAS3) to facilitate reading and writing MPU entries. The MAS registers can be read or written using the **mfspr** and **mtspr** instructions.

The MAS0 register is shown in *Figure 82*. Fields are defined in *Table 178*.

Offset: Refer to *Figure 64* (SPR - 624) Access: Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	VALID	IPROT	SEL		0	RO	DEBUG	INST	SHD	0			ESEL			
W																
Reset	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	UAMSK			UW	SW	UX/UR	SX/SR	IOVR	GOVR(1)	1	1	I	0	G(1)	0
W																
Reset	—	—	—	—	—	—	—	—	—	—	1	1	—	—	—	—

1. This bit is not implemented in dedicated instruction entries, and is ignored in shared entries with INST = 1

Figure 82. MPU Assist Register 0 (MAS0)

Table 178. MAS0 (MPU Read/Write and Replacement Control) fields description

Name	Comments, or Function when Set
0 VALID	MPU Entry Valid 0 This MPU entry is invalid 1 This MPU entry is valid
1 IPROT	Invalidation Protect 0 Entry is not protected from invalidation 1 Entry is protected from invalidation
2:3 SEL	Selects MPU for access: 00 Reserved, no access 01 Reserved, no access 10 Select MPU 11 Reserved, no access
5 RO	Read-Only When set to 1, the entry is no longer writable by software. Once set, this bit will remain set until a processor reset occurs. 0 This MPU entry is writable 1 This MPU entry is Read-only
6 DEBUG	Debug Control for Entry This bit is used to assign an entry for debug event use instead of normal access protection use. When used for debug purposes, an MPU debug event is generated when a hit to the entry occurs and the access permissions allow the access. 0 This MPU entry is used for access protections and allows accesses when matched based on protection attributes 1 This MPU entry is used for generating a debug event when a match occurs and the access protections would allow access
7 INST	Instruction Entry When SHD = 0, this bit is used to select the INST entry portion of the region descriptor table for **mpure** and **mpuwe** instruction operations. When SHD = 1, this bit is used to indicate whether the entry in the Shared portion of the region descriptor table is assigned as an entry for either instruction access or data access matching. Only one of these access types is monitored for matching by a given shared region descriptor entry. 0 This MPU entry is used for matching data accesses only 1 This MPU entry is used for matching instruction accesses only
8 SHD	Shared Entry Select This bit is used to control selection of the Shared portion of the region descriptor table. 0 The shared portion of the region descriptor table is not accessed on a **mpure** or **mpuwe** operation. Either the instruction portion or the data portion is accessed based on the setting of INST. The entry within the selected portion is based on ESEL. 1 The shared portion of the region descriptor table is accessed on a **mpure** or **mpuwe** operation. The instruction and data portions are not accessed. The INST bit is used to indicate whether the entry in the Shared portion of the region descriptor table is assigned as an entry for instruction access or data access matching. Only one of these access types is monitored for matching by a given shared region descriptor entry. ESEL defines which shared entry is selected.

Table 178. MAS0 (MPU Read/Write and Replacement Control) fields description

Name	Comments, or Function when Set
12:15 ESEL	Entry select for MPU. This field is used to select an entry for reading or writing in conjunction with the settings of SHD and INST. Only valid entry numbers should be used in this field, otherwise the result of an operation is boundedly undefined.
17:19 UAMSK	Upper Address Mask Control This field is used to support masking of upper address bits before performing range comparisons. Masked bits will be forced to 0 for the purposes of the range compare. Range upper and lower bounds should be set accordingly. 000 No upper address bits are masked, address matching uses all 32 address bits for accesses 001 The most significant access address bit is forced to zero before matching 010 The two most significant access address bits are forced to zero before matching 101 The upper five access address bits are forced to zero before matching 11x Reserved, do not use
20 UW	User Mode Write Permission Determines User mode Write (W) permission when INST = 0. Ignored when INST = 1. 0 No User mode Write permission 1 User mode has Write permission
21 SW	Supervisor Mode Write / Read Permission Determines Supervisor mode Write (W) permission when INST = 0. Ignored when INST = 1. 0 No Supervisor mode Write permission 1 Supervisor mode has Write permission
22 UX/UR	User Mode Execute / Read Permission Determines User mode Execute (X) permission when INST = 1, or User mode Read (R) permission when INST = 0. 0 No User mode Execute/ Read permission 1 User mode has Execute/ Read permission
23 SX/SR	Supervisor Mode Execute / Read Permission Determines Supervisor mode Execute (X) permission when INST = 1, or Supervisor mode Read (R) permission when INST = 0. 0 No Supervisor mode Execute/ Read permission 1 Supervisor mode has Execute/ Read permission
24 IOVR	Cache-Inhibit attribute Override Determines whether this matching entry overrides the I bit settings of other matching entries and the cache-inhibited region configuration signals 0 No override of I attribute by entry 1 Entry I bit overrides I attribute
25 GOVR/—	G attribute Override Determines whether this entry overrides the G bit settings of other matching entries and the Guarded region configuration signals This bit is not implemented in dedicated instruction entries, and is ignored in shared entries with INST = 1. 0 No override of G attribute by entry 1 Entry G bit overrides G attribute

Table 178. MAS0 (MPU Read/Write and Replacement Control) fields description

Name	Comments, or Function when Set
28 I	Cache Inhibited This attribute may be overridden by the cache-inhibited region configuration input settings. 0 This region is considered cacheable 1 This region is considered cache-inhibited
30 G/—	Guarded This attribute may be overridden by the guarded region configuration input settings. This bit is not implemented in dedicated instruction entries, and is ignored in shared entries with INST = 1. 0 Accesses to this region are not guarded, and can be performed before it is known if they are required by the sequential execution model 1 All loads and stores to this region are performed without speculation (this means they are known to be required)

The MAS1 register is shown in *Figure 83*. Fields are defined in *Table 179*.

Offset: Refer to *Figure 64* (SPR - 625) Access: Read/Write

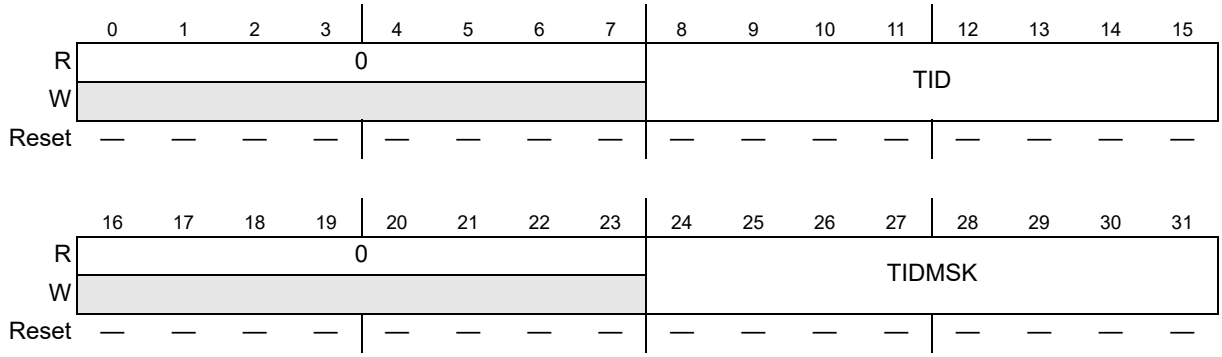


Figure 83. MPU Assist Register 1 (MAS1)

Table 179. MAS1 (Descriptor Context and Configuration Control) fields description

Name	Comments, or Function when Set
8:15 TID	Region ID bits This field is compared with the current process ID of the access address. A TID value of 0 defines an entry as global and matches with all process IDs.
24:31 TIDMSK	Region ID mask 0xxxxxxx TID[0] comparison to PID0[0] not masked 1xxxxxxx TID[0] comparison to PID0[0] is masked x0xxxxxx TID[1] comparison to PID0[1] not masked x1xxxxxx TID[1] comparison to PID0[1] is masked xx0xxxxx TID[2] comparison to PID0[2] not masked xx1xxxxx TID[2] comparison to PID0[2] is masked xxx0xxxx TID[3] comparison to PID0[3] not masked xxx1xxxx TID[3] comparison to PID0[3] is masked xxxx0xxx TID[4] comparison to PID0[4] not masked xxxx1xxx TID[4] comparison to PID0[4] is masked xxxxx0xx TID[5] comparison to PID0[5] not masked xxxxx1xx TID[5] comparison to PID0[5] is masked xxxxxx0x TID[6] comparison to PID0[6] not masked xxxxxx1x TID[6] comparison to PID0[6] is masked xxxxxxx0 TID[7] comparison to PID0[7] not masked xxxxxxx1 TID[7] comparison to PID0[7] is masked This field controls whether bits within the TID to PID0 comparison are masked for determining a range match. When a bit is masked, the value of that TID and PID0 bit is ignored for matching purposes.

The MAS2 register is shown in *Figure 84*. Fields are defined in *Table 180*.

Offset: Refer to *Figure 64* (SPR - 626) Access: Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	UPPER BOUND															
Reset	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	UPPER BOUND															
Reset	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—

Figure 84. MPU Assist Register 2 (MAS2)

Table 180. MAS2 (Upper Bound Control) fields description

Name	Comments, or Function when Set
0:31 UPPER BOUND	Upper bound of address range covered by entry. Entry match requires LOWER_BOUND <= EA <= UPPER_BOUND

The MAS3 register is shown in *Figure 85*. Fields are defined in *Table 181*.

Offset: Refer to *Figure 64* (SPR - 627) Access: Read/Write

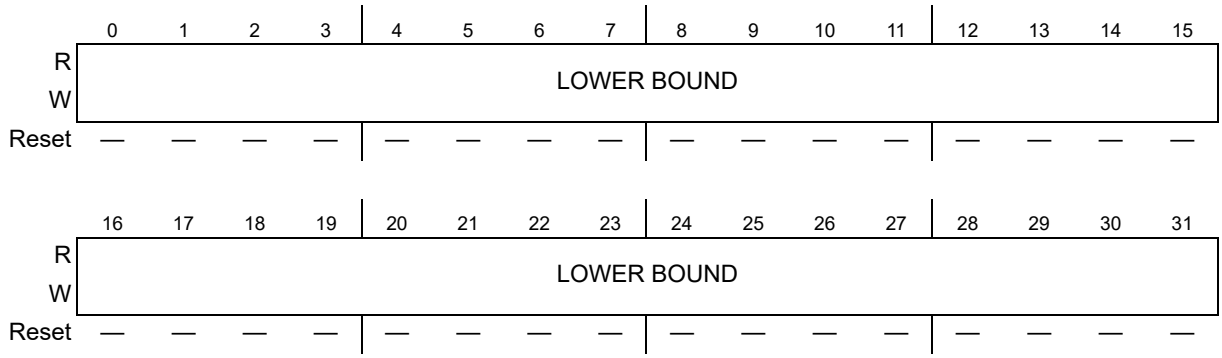


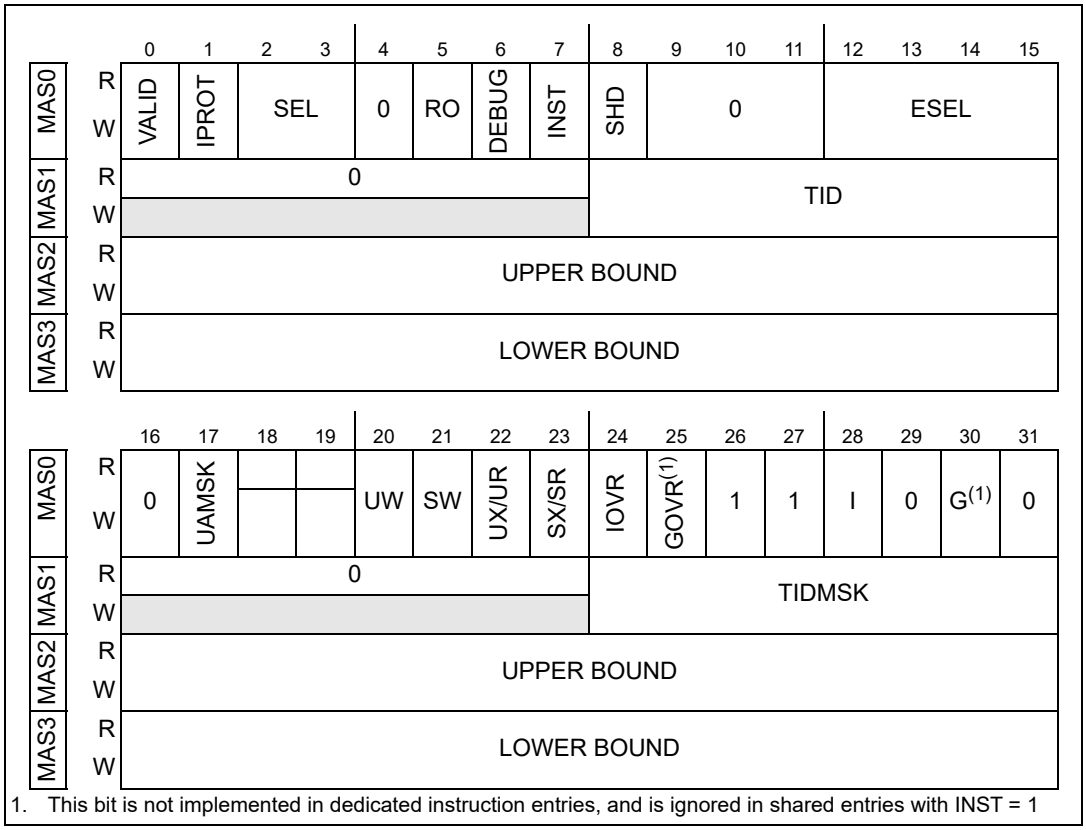
Figure 85. MPU Assist Register 3 (MAS3)

Table 181. MAS3 (Lower Bound Control) fields description

Name	Comments, or Function when Set
0:31 LOWER BOUND	Lower bound of address range covered by entry. Entry match requires LOWER_BOUND <= EA <= UPPER_BOUND

12.8.11 MAS Registers Summary

The MAS registers are summarized in *Figure 86*.



1. This bit is not implemented in dedicated instruction entries, and is ignored in shared entries with INST = 1

Figure 86. MPU Assist Registers Summary

12.9 Local memories

12.9.1 Local Instruction and Data Memory Overview

In order to provide for low latency memory access for critical instruction routines and data operands, local instruction (IMEM) and data (DMEM) memory capabilities have been added. This provides low latency access (zero wait-states) similar to an instruction or data cache, but does not suffer from the overhead of cache miss or cache writethrough operations. Instead, it provides a large capacity “tightly coupled” storage capability.

12.9.2 Local Memory Control and Configuration

DMEM and IMEM local memory configuration information is provided by a set of privileged read-only SPRs which are accessed using **mfspr** instructions. DMEM and IMEM local memory operation is controlled by a set of privileged device control registers (DCRs) which are accessed using the **mfdcr** and **mtdcr** instructions. These registers and a description of the operation of various features are described in the following subsections.

12.9.2.1 DMEM Configuration Register 0 (DMEMCFG0)

The DMEM Configuration Register 0 (DMEMCFG0) is a 32-bit read-only register. DMEMCFG0 provides information about the configuration of the e200z420n3 local data memory design. The contents of the DMEMCFG0 register can be read using a **mfspr** instruction. The SPR number for DMEMCFG0 is 694 in decimal. The DMEMCFG0 register is shown in *Figure 87*.

Offset: Refer to *Figure 64* (SPR - 694) Access: Read Only Privileged

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DMEM BASE ADDR[0:15]															
W																
Reset	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	DMEM BASE ADDR[16:19]				0	DECUA	DECA	0	DMSIZE						0	
W																
Reset	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—

Figure 87. DMEM Configuration Register 0 (DMEMCFG0)

The DMEMCFG0 fields are described in *Table 182*.

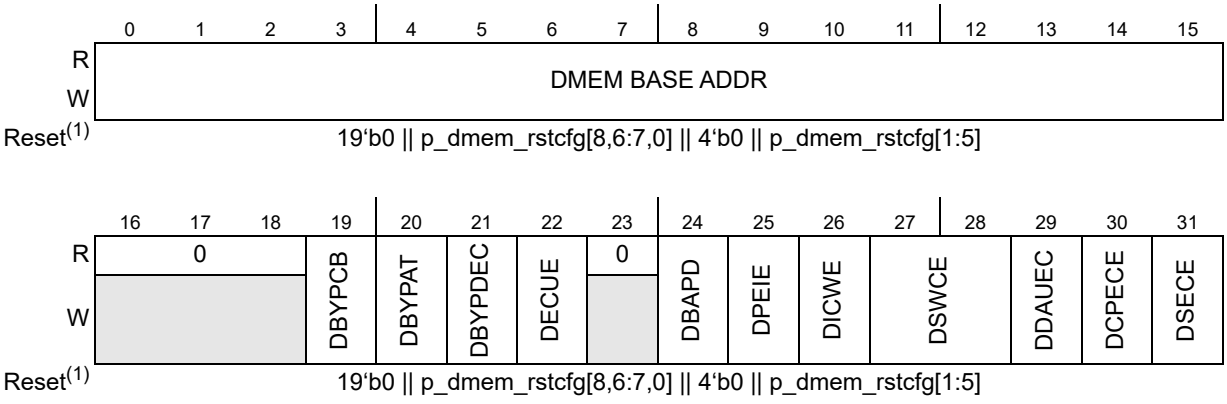
Table 182. DMEMCFG0 fields description

Field	Description
0:19 DMEM BASE ADDR	DMEM BASE ADDRESS (CPU Port) This field defines the current Base Address value being used for the CPU port to the DMEM. This field reflects the value of the external DMEM base address port inputs when the external base address port inputs are enabled via the DBAPD control bit, or the value of the BASE ADDRESS field of DMEMCTL0 when the external base address port inputs are disabled via the DBAPD control bit. **Note:** Low order bits of this field are driven to 0's when the DMEM size is > 4 KB
21 DECUA	DMEM Error Correction Update Available DECUA indicates an error scrubbing function for the DMEM is available. 0 Error Correction Update is not available 1 Error Correction Update is available for correction of a correctable single-bit error detected on a read access
22 DECA	DMEM Error Correction Available DECA indicates an error correction function for the DMEM is available. 0 Error Correction is not available 1 Error Correction is available
24:29 DMSIZE	DMEM Size 010000 The size of the DMEM is 64 KB

12.9.2.2 DMEM Control Register 0 (DMEMCTL0)

The DMEM Control Register 0 (DMEMCTL0) controls operation of certain functions of the DMEM logic.

Offset: Refer to *Figure 65* (DCR - 496) Access: Read/Write Privileged



1. Reset by **m_por** and **slv_reset_b**. Unaffected by **p_reset_b**.

Figure 88. DMEM Control Register 0 (DMEMCTL0)

Table 183. DMEMCTL0 field descriptions

Field	Description
0:15 DMEM BASE ADDR	DMEM BASE ADDRESS Field (CPU Port) This field defines the Base Address used for the CPU port to the DMEM when the external base address port inputs are disabled via the DBAPD control bit. **Note:** Changes to this value and to the BAPD control bit must be performed carefully by software to ensure coherency is maintained. Specifically, software must ensure that no cached entries are present in the D-Cache for the memory space to be occupied by the DMEM.
19 DBYPCB	DMEM Bypass Cache Bypass CPU accesses DBYPCB can be used to force Non-decorated Cache Bypass loads and Store with Writethrough (**lbcbx**, **lhcbx**, **lwcbx**, and **stwwtx**, **sthwtx**, **stbwtx**) accesses which target the DMEM address space to be presented to the external bus. Note that the protection settings in DMEMCTL1 are still applied to these accesses. **Note:** This field is updated on **m_por** and **slv_reset_b** based on the value of the **p_dmem_rstcfg[8]** input. 0 Non-decorated Cache Bypass loads and Store with Writethrough CPU accesses do not bypass the DMEM CPU port 1 Non-decorated Cache Bypass loads and Store with Writethrough CPU accesses bypass the DMEM CPU port and are routed out through the BIU to the DAHB external interface.
20 DBYPAT	DMEM Bypass Atomic CPU accesses DBYPAT can be used to force atomic (load and reserve, store conditional) accesses to the DMEM address space to be presented to external reservation hardware in systems which support reservation and/or decoration functionality, since no internal reservation hardware is present in the CPU for the DMEM. In general, software must not rely on normal CPU write accesses to clear a reservation, since they are not monitored by reservation hardware; instead only store conditional accesses should be used for this purpose. Note that the protection settings in DMEMCTL1 are still applied to these accesses. **Note:** This field is updated on **m_por** and **slv_reset_b** based on the value of the **p_dmem_rstcfg[6]** input. 0 Atomic CPU accesses do not bypass the DMEM CPU port 1 Atomic CPU accesses bypass the DMEM CPU port and are routed out through the BIU to the DAHB external interface.
21 DBYPDEC	DMEM Bypass Decorated CPU accesses DBYPDEC can be used to force decorated accesses (**lbdx, lhdx, lwdx, lbdcbx, lhdcbx, lwdcbx, stbdx, sthdx, stwdx, stbdcbx, sthdcbx, stwdcbx, dsn, dsncb**) to the DMEM address space to be presented to external decoration hardware for systems which support this functionality, since no internal decoration hardware is present in the CPU. Note that the protection settings in DMEMCTL1 are still applied to these accesses. **Note:** This field is updated on **m_por** and **slv_reset_b** based on the value of the **p_dmem_rstcfg[7]** input. 0 Decorated Load and Store CPU accesses do not bypass the DMEM CPU port 1 Decorated Load and Store CPU accesses bypass the DMEM CPU port and are routed out through the BIU to the DAHB external interface.

Table 183. DMEMCTL0 field descriptions (continued)

Field	Description
22 DECUE	DMEM Error Correction Update Enable DECUE can be used to provide an error scrubbing function for the DMEM. **Note:** This field is updated on **m_por** and **slv_reset_b** based on the value of the **p_dmem_rstcfg[0]** input. 0 Error Correction Update is disabled 1 Error Correction Update is enabled. A correctable single-bit error detected on a read access from either the CPU or the slave port will cause the DMEM to be re-written with the corrected data if the corresponding error checking enable bit is set (DCPECE for CPU accesses, DSECE for slave port accesses). (Write accesses perform this correction automatically during partial-width writes when error checking is enabled.) DECUE can be used in conjunction with DCPECE and DSECE to provide an error scrubbing function for the DMEM.
24 DBAPD	DMEM Base Address Port Disable This bit controls usage of the external Base Address port (**p_dmem_baseaddr[0:n]**) to define the base address of the DMEM for CPU port accesses. It has no effect on DMEM slave port accesses. 0 The external DMEM base address port (**p_dmem_baseaddr[0:n]**) is used to define the base address of the DMEM for CPU accesses. 1 The external DMEM base address port (**p_dmem_baseaddr[0:n]**) is disabled for use, and instead the BASE ADDR field value is used to define the base address of the DMEM for CPU accesses.
25 DPEIE	DMEM Processor Error Injection Enable DPEIE will cause injection of errors regardless of the setting of DCPECE, although reporting of errors to the CPU will be masked while DCPECE = 0. 0 Error Injection is disabled 1 A double-bit error will be injected on each CPU port write into the DMEM data array by inverting the two uppermost parity check bits (p_dchk[0:1]).
26 DICWE	DMEM Imprecise CPU Write Enable DICWE may allow for increased partial-width write performance when set. 0 Imprecise writes by the CPU are disabled. All partial-width write ECC errors are reported precisely (error report machine check). 1 Imprecise writes by the CPU are enabled. In certain cases, partial-width write errors may be reported imprecisely (async machine check only).
27:28 DSWCE	DMEM Slave port Write Check/Correct Enable **Note:** This field is updated on **m_por** and **slv_reset_b** based on the values of the **p_dmem_rstcfg[1:2]** inputs. 00 Slave write data is not checked or corrected for errors. 01 Slave write data is checked but not corrected for errors. Detected errors generate a slave port ERROR response and no Write is performed to the DMEM. 10 Slave write data is checked and corrected for errors on all writes. Single-bit correctable errors do not automatically generate an ERROR response, but are instead corrected. 11 Slave write data is checked and corrected for errors on partial-width (1, 2, 3, 5, 6, or 7 byte, or misaligned 4-byte writes) writes. Single-bit correctable errors on partial-width writes do not automatically generate an ERROR response, but are instead corrected. Aligned word or doubleword writes are not checked or corrected for errors.

Table 183. DMEMCTL0 field descriptions (continued)

Field	Description
29 DDAUEC	DMEM Disable Address Use in Error Check This bit controls usage of the address portion of the ECC H matrix. When use is disabled, the address portions are not used for checkbit/syndrome generation for DMEM CPU or slave port accesses. **Note:** This field is updated on **m_por** and **slv_reset_b** based on the value of the **p_dmem_rstcfg[3]** input. 0 Use of access address is enabled in checkbit and syndrome generation. 1 Use of access address is disabled in checkbit and syndrome generation.
30 DCPECE	DMEM CPU Port ECC Enable (CPU port) **Note:** This field is updated on **m_por** and **slv_reset_b** based on the value of the **p_dmem_rstcfg[4]** input. 0 End-to-End ECC is disabled for the CPU interface. No checking of read data is performed by the CPU. Writes will still generate the proper check bit values to be written. 1 End-to-End ECC is enabled for performing ECC on the CPU interface. Error checking of read data is performed with single-bit error correction. Detection of uncorrectable single- or multi-bit errors on a CPU port access cause a machine check to be generated.
31 DSECE	DMEM Slave port Error Checking Enable (Slave port) **Note:** This field is updated on **m_por** and **slv_reset_b** based on the value of the **p_dmem_rstcfg[5]** input. 0 End-to-End ECC checking logic is disabled for the Slave interface. No checking of read data is performed during read-modify-write operations for Slave port partial-width writes. Writes will still generate the proper check bit values to be written. 1 End-to-End ECC is enabled for performing ECC on the Slave Port interface. Error checking of read data is performed with single-bit error correction during read-modify-write operations for partial-width writes. Detection of uncorrectable single- or multi-bit errors on a Slave port partial-width write access causes a bus error ERROR response to be generated.

12.9.2.3 DMEM Control Register 1 (DMEMCTL1)

The DMEM Control Register 1 (DMEMCTL1) provides protection functions for the DMEM. Separate Supervisor-mode and User-mode read and write access controls allow accesses to be granted, denied, or conditionally granted independently for four equally sized blocks of DMEM. Conditional granting of access permissions causes the MPU memory protection functions to be employed, all other settings override MPU settings. The DMEMCTL1 settings are applied to all accesses which target DMEM address range, regardless of whether access may bypass the DMEM based on settings in DMEMCTL0$_{DBYPCB, DBYPATF, DBYPDEC}$.

Offset: Refer to *Figure 65* (DCR - 498) Access: Read/Write Privileged

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	DSWAC3		DSWAC2		DSWAC1		DSWAC0		DSRAC3		DSRAC2		DSRAC1		DSRAC0	
Reset(1)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	DUWAC3		DUWAC2		DUWAC1		DUWAC0		DURAC3		DURAC2		DURAC1		DURAC0	
Reset(1)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. Reset by **m_por** and **p_reset_b**.

Figure 89. DMEM Control Register 1 (DMEMCTL1)

Table 184. DMEMCTL1 fields description

Field	Description
0:1 DSWAC3	DMEM Supervisor Write Access Control for Quadrant 3 **Note:** Individual control fields are provided for four equal sized sections of DMEM regardless of DMEM size. A single control field is used for each access based on the upper two address bits used to index into the DMEM. DSWAC3 is used when these two bits = 2'b11 00 Supervisor-mode CPU write accesses are allowed to quadrant 3 of DMEM 01 Supervisor-mode CPU write accesses are not allowed to quadrant 3 of DMEM 10 Supervisor-mode CPU write accesses are conditionally allowed to quadrant 3 of DMEM based on memory protection logic 11 Reserved
2:3 DSWAC2	DMEM Supervisor Write Access Control for Quadrant 2 **Note:** Individual control fields are provided for four equal sized sections of DMEM regardless of DMEM size. A single control field is used for each access based on the upper two address bits used to index into the DMEM. DSWAC2 is used when these two bits = 2'b10 00 Supervisor-mode CPU write accesses are allowed to quadrant 2 of DMEM 01 Supervisor-mode CPU write accesses are not allowed to quadrant 2 of DMEM 10 Supervisor-mode CPU write accesses are conditionally allowed to quadrant 2 of DMEM based on memory protection logic 11 Reserved
4:5 DSWAC1	DMEM Supervisor Write Access Control for Quadrant 1 **Note:** Individual control fields are provided for four equal sized sections of DMEM regardless of DMEM size. A single control field is used for each access based on the upper two address bits used to index into the DMEM. DSWAC1 is used when these two bits = 2'b01 00 Supervisor-mode CPU write accesses are allowed to quadrant 1 of DMEM 01 Supervisor-mode CPU write accesses are not allowed to quadrant 1 of DMEM 10 Supervisor-mode CPU write accesses are conditionally allowed to quadrant 1 of DMEM based on memory protection logic 11 Reserved

Table 184. DMEMCTL1 fields description (continued)

Field	Description
6:7 DSWAC0	DMEM Supervisor Write Access Control for Quadrant 0 **Note:** Individual control fields are provided for four equal sized sections of DMEM regardless of DMEM size. A single control field is used for each access based on the upper two address bits used to index into the DMEM. DSWAC0 is used when these two bits = 2‘b00 00 Supervisor-mode CPU write accesses are allowed to quadrant 0 of DMEM 01 Supervisor-mode CPU write accesses are not allowed to quadrant 0 of DMEM 10 Supervisor-mode CPU write accesses are conditionally allowed to quadrant 0 of DMEM based on memory protection logic 11 Reserved
8:9 DSRAC3	DMEM Supervisor Read Access Control for Quadrant 3 **Note:** Individual control fields are provided for four equal sized sections of DMEM regardless of DMEM size. A single control field is used for each access based on the upper two address bits used to index into the DMEM. DSRAC3 is used when these two bits = 2‘b11 00 Supervisor-mode CPU read accesses are allowed to quadrant 3 of DMEM 01 Supervisor-mode CPU read accesses are not allowed to quadrant 3 of DMEM 10 Supervisor-mode CPU read accesses are conditionally allowed to quadrant 3 of DMEM based on memory protection logic 11 Reserved
10:11 DSRAC2	DMEM Supervisor Read Access Control for Quadrant 2 **Note:** Individual control fields are provided for four equal sized sections of DMEM regardless of DMEM size. A single control field is used for each access based on the upper two address bits used to index into the DMEM. DSRAC2 is used when these two bits = 2‘b10 00 Supervisor-mode CPU read accesses are allowed to quadrant 2 of DMEM 01 Supervisor-mode CPU read accesses are not allowed to quadrant 2 of DMEM 10 Supervisor-mode CPU read accesses are conditionally allowed to quadrant 2 of DMEM based on memory protection logic 11 Reserved
12:13 DSRAC1	DMEM Supervisor Read Access Control for Quadrant 1 **Note:** Individual control fields are provided for four equal sized sections of DMEM regardless of DMEM size. A single control field is used for each access based on the upper two address bits used to index into the DMEM. DSRAC1 is used when these two bits = 2‘b01 00 Supervisor-mode CPU read accesses are allowed to quadrant 1 of DMEM 01 Supervisor-mode CPU read accesses are not allowed to quadrant 1 of DMEM 10 Supervisor-mode CPU read accesses are conditionally allowed to quadrant 1 of DMEM based on memory protection logic 11 Reserved

Table 184. DMEMCTL1 fields description (continued)

Field	Description
14:15 DSRAC0	DMEM Supervisor Read Access Control for Quadrant 0 **Note:** Individual control fields are provided for four equal sized sections of DMEM regardless of DMEM size. A single control field is used for each access based on the upper two address bits used to index into the DMEM. DSRAC0 is used when these two bits = 2‘b00 00 Supervisor-mode CPU read accesses are allowed to quadrant 0 of DMEM 01 Supervisor-mode CPU read accesses are not allowed to quadrant 0 of DMEM 10 Supervisor-mode CPU read accesses are conditionally allowed to quadrant 0 of DMEM based on memory protection logic 11 Reserved
16:17 DUWAC3	DMEM User Write Access Control for Quadrant 3 **Note:** Individual control fields are provided for four equal sized sections of DMEM regardless of DMEM size. A single control field is used for each access based on the upper two address bits used to index into the DMEM. DUWAC3 is used when these two bits = 2‘b11. 00 User-mode CPU write accesses are allowed to quadrant 3 of DMEM 01 User-mode CPU write accesses are not allowed to quadrant 3 of DMEM 10 User-mode CPU write accesses are conditionally allowed to quadrant 3 of DMEM based on memory protection logic 11 Reserved
18:19 DUWAC2	DMEM User Write Access Control for Quadrant 2 **Note:** Individual control fields are provided for four equal sized sections of DMEM regardless of DMEM size. A single control field is used for each access based on the upper two address bits used to index into the DMEM. DUWAC2 is used when these two bits = 2‘b10 00 User-mode CPU write accesses are allowed to quadrant 2 of DMEM 01 User-mode CPU write accesses are not allowed to quadrant 2 of DMEM 10 User-mode CPU write accesses are conditionally allowed to quadrant 2 of DMEM based on memory protection logic 11 Reserved
20:21 DUWAC1	DMEM User Write Access Control for Quadrant 1 **Note:** Individual control fields are provided for four equal sized sections of DMEM regardless of DMEM size. A single control field is used for each access based on the upper two address bits used to index into the DMEM. DUWAC1 is used when these two bits = 2‘b01 00 User-mode CPU write accesses are allowed to quadrant 1 of DMEM 01 User-mode CPU write accesses are not allowed to quadrant 1 of DMEM 10 User-mode CPU write accesses are conditionally allowed to quadrant 1 of DMEM based on memory protection logic 11 Reserved

Table 184. DMEMCTL1 fields description (continued)

Field	Description
22:23 DUWAC0	DMEM User Write Access Control for Quadrant 0 **Note:** Individual control fields are provided for four equal sized sections of DMEM regardless of DMEM size. A single control field is used for each access based on the upper two address bits used to index into the DMEM. DUWAC0 is used when these two bits = 2'b00 00 User-mode CPU write accesses are allowed to quadrant 0 of DMEM 01 User-mode CPU write accesses are not allowed to quadrant 0 of DMEM 10 User-mode CPU write accesses are conditionally allowed to quadrant 0 of DMEM based on memory protection logic 11 Reserved
24:25 DURAC3	DMEM User Read Access Control for Quadrant 3 **Note:** Note: Individual control fields are provided for four equal sized sections of DMEM regardless of DMEM size. A single control field is used for each access based on the upper two address bits used to index into the DMEM. DURAC3 is used when these two bits = 2'b11. 00 User-mode CPU read accesses are allowed to quadrant 3 of DMEM 01 User-mode CPU read accesses are not allowed to quadrant 3 of DMEM 10 User-mode CPU read accesses are conditionally allowed to quadrant 3 of DMEM based on memory protection logic 11 Reserved
26:27 DURAC2	DMEM User Read Access Control for Quadrant 2 **Note:** Individual control fields are provided for four equal sized sections of DMEM regardless of DMEM size. A single control field is used for each access based on the upper two address bits used to index into the DMEM. DURAC2 is used when these two bits = 2'b10 00 User-mode CPU read accesses are allowed to quadrant 2 of DMEM 01 User-mode CPU read accesses are not allowed to quadrant 2 of DMEM 10 User-mode CPU read accesses are conditionally allowed to quadrant 2 of DMEM based on memory protection logic 11 Reserved

Table 184. DMEMCTL1 fields description (continued)

Field	Description
28:29 DURAC1	DMEM User Read Access Control for Quadrant 1 **Note:** Individual control fields are provided for four equal sized sections of DMEM regardless of DMEM size. A single control field is used for each access based on the upper two address bits used to index into the DMEM. DURAC1 is used when these two bits = 2'b01 00 User-mode CPU read accesses are allowed to quadrant 1 of DMEM 01 User-mode CPU read accesses are not allowed to quadrant 1 of DMEM 10 User-mode CPU read accesses are conditionally allowed to quadrant 1 of DMEM based on memory protection logic 11 Reserved
30:31 DURAC0	DMEM User Read Access Control for Quadrant 0 **Note:** Individual control fields are provided for four equal sized sections of DMEM regardless of DMEM size. A single control field is used for each access based on the upper two address bits used to index into the DMEM. DURAC0 is used when these two bits = 2'b00 00 User-mode CPU read accesses are allowed to quadrant 0 of DMEM 01 User-mode CPU read accesses are not allowed to quadrant 0 of DMEM 10 User-mode CPU read accesses are conditionally allowed to quadrant 0 of DMEM based on memory protection logic 11 Reserved

12.9.2.4 IMEM Configuration Register 0 (IMEMCFG0)

Note: These locations are accessible by S/W and will not result in error if accessed.

The IMEM Configuration Register 0 (IMEMCFG0) is a 32-bit read-only register. IMEMCFG0 provides information about the configuration of the e200z420n3 local instruction memory design. The contents of the IMEMCFG0 register can be read using a **mfspr** instruction. The SPR number for IMEMCFG0 is 695 in decimal. The IMEMCFG0 register is shown in *Figure 90*.

Offset: Refer to *Figure 64* (SPR - 695) Access: Read Only Privileged

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	IMEM BASE ADDR[0:15]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	IMEM BASE ADDR[16:19]				0	IECUA	IECA	0	IMSIZE						0	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 90. IMEM Configuration Register 0 (IMEMCFG0)

The IMEMCFG0 fields are described in *Table 185*.

Table 185. IMEMCFG0 fields description

Field	Description
0:19 IMEM BASE ADDR	IMEM BASE ADDRESS (CPU Port) This field defines the current Base Address value being used for the CPU port to the IMEM. This field reflects the value of the external IMEM base address port inputs when the external base address port inputs are enabled via the DBAPD control bit, or the value of the BASE ADDRESS field of IMEMCTL0 when the external base address port inputs are disabled via the DBAPD control bit. **Note:** Low order bits of this field are driven to 0's when the IMEM size is > 4 KB
21 IECUA	IMEM Error Correction Update Available IECUA indicates an error scrubbing function for the IMEM is available. 0 Error Correction Update is not available 1 Error Correction Update is available for correction of a correctable single-bit error detected on a read access
22 IECA	IMEM Error Correction Available IECA indicates an error correction function for the IMEM is available. 0 Error Correction is not available 1 Error Correction is available
24:29 IMSIZE	IMEM Size 000100 The size of the IMEM is 16 KB

12.9.2.5 IMEM Control Register 0 (IMEMCTL0)

Note: These locations are accessible by S/W and will not result in error if accessed.

The IMEM Control Register 0 (IMEMCTL0) controls operation of certain functions of the IMEM logic.

Offset: Refer to *Figure 65* (DCR - 497) Access: Read/Write Privileged

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	IMEM BASE ADDR[0:15]															
Reset(1)	22'b0 ||| p_imem_rstcfg[0] || 4'b0 || p_imem_rstcfg[1:5]															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	IMEN BASE ADDR[16:17]		0		0		IECUE	0	IBAPD	0		ISWCE		IDAUEC	ICPECE	ISECE
W																
Reset(1)	22'b0 ||| p_imem_rstcfg[0] || 4'b0 || p_imem_rstcfg[1:5]															

1. Reset by **m_por** and **slv_reset_b**. Unaffected by **p_reset_b**.

Figure 91. IMEM Control Register 0 (IMEMCTL0)

Table 186. IMEMCTL0 fields description

Field	Description
0:17 IMEM BASE ADDR	IMEM BASE ADDRESS Field (CPU Port) This field defines the Base Address used for the CPU port to the IMEM when the external base address port inputs are disabled via the BAPD control bit. **Note:** Changes to this value and to the BAPD control bit must be performed carefully by software to ensure coherency is maintained. Specifically, software must ensure that the BTB is invalidated, and that no cached entries are present in the I-Cache for the memory space to be occupied by the IMEM.
18:19	Reserved for Base Address extension
22 IECUE	IMEM Error Correction Update Enable IECUE can be used to provide an error scrubbing function. This field is updated on **m_por** and **slv_reset_b** based on the value of the **p_imem_rstcfg[0]** input. 0 Error Correction Update is disabled 1 Error Correction Update is enabled. A correctable single-bit error detected on CPU instruction fetches or slave port read cycles will cause the IMEM to be re-written with the corrected data if the corresponding error checking enable bit is set (ICPECE for CPU accesses, ISECE for slave port accesses). IECUE can be used in conjunction with ICPECE and ISECE to provide an error scrubbing function.
24 IBAPD	IMEM Base Address Port Disable This bit controls usage of the external Base Address port (**p_imem_baseaddr[0:n]**) to define the base address of the IMEM for CPU port accesses. It has no effect on IMEM slave port accesses. 0 The external IMEM base address port (**p_imem_baseaddr[0:n]**) is used to define the base address of the IMEM for CPU accesses. 1 The external IMEM base address port (**p_imem_baseaddr[0:n]**) is disabled for use, and instead the BASE ADDR field value is used to define the base address of the IMEM for CPU accesses.
27:28 ISWCE	IMEM Slave port Write Check/Correct Enable **Note:** This field is updated on **m_por** and **slv_reset_b** based on the values of the **p_imem_rstcfg[1:2]** inputs. 00 Slave write data is not checked or corrected for errors. 01 Slave write data is checked but not corrected for errors. Detected errors generate a slave port ERROR response and no Write is performed to the IMEM. 10 Slave write data is checked and corrected for errors on all writes. Single-bit correctable errors do not automatically generate an ERROR response, but are instead corrected. 11Slave write data is checked and corrected for errors on partial-width (1, 2, 3, 4, 5, 6, or 7 byte) writes. Single-bit correctable errors on partial-width writes do not automatically generate an ERROR response, but are instead corrected.

Table 186. IMEMCTL0 fields description (continued)

Field	Description
29 IDAUEC	IMEM Disable Address Use in Error Check This bit controls usage of the address portion of the ECC H matrix. When use is disabled, the address portions are not used. for checkbit/syndrome generation for IMEM CPU or slave port accesses. **Note:** This field is updated on **m_por** and **slv_reset_b** based on the value of the **p_imem_rstcfg[3]** input. 0 Use of access address is enabled in checkbit and syndrome generation. 1 Use of access address is disabled in checkbit and syndrome generation.
30 ICPECE	IMEM CPU Port ECC Enable (CPU port) **Note:** This field is updated on **m_por** and **slv_reset_b** based on the value of the **p_imem_rstcfg[4]** input. 0 End-to-End ECC is disabled for the CPU interface. No checking of read data is performed by the CPU. Writes will still generate the proper check bit values to be written. 1 End-to-End ECC is enabled for performing ECC on the CPU interface. Error checking of read data is performed with single-bit error correction. Detection of uncorrectable single- or multi-bit errors on a CPU port access cause a machine check to be generated.
31 ISECE	IMEM Slave port Error Checking Enable (Slave port) **Note:** This field is updated on **m_por** and **slv_reset_b** based on the value of the **p_imem_rstcfg[5]** input. 0 End-to-End ECC checking logic is disabled for the Slave interface. No checking of read data is performed during read-modify-write operations for partial-width writes. Writes will still generate the proper check bit values to be written. 1 End-to-End ECC is enabled for performing ECC on the Slave Port interface. Error checking of read data is performed during read-modify-write operations for partial-width writes with single-bit error correction. Detection of uncorrectable single- or multi-bit errors on a Slave port access causes a bus error ERROR response to be generated.

12.9.2.6 IMEM Control Register 1 (IMEMCTL1)

Note: These locations are accessible by S/W and will not result in error if accessed.

The IMEM Control Register 1 (IMEMCTL1) provides protection functions for the IMEM. Separate Supervisor-mode and User-mode instruction fetch access controls allow accesses to be granted, denied, or conditionally granted independently for four equally sized blocks of IMEM. Conditional granting of access permissions causes the MPU memory protection functions to be employed, all other settings override MPU settings.

Offset: Refer to *Figure 65* (DCR - 499) Access: Read/Write Privileged

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0								ISXAC3		ISXAC2		ISXAC1		ISXAC0	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0								IUXAC3		IUXAC2		IUXAC1		IUXAC0	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 92. IMEM Control Register 1 (IMEMCTL1)

Table 187. IMEMCTL1 fields description

Field	Description
8:9 ISXAC3	IMEM Supervisor Instruction Fetch Access Control for Quadrant 3 **Note:** Individual control fields are provided for four equal sized sections of IMEM regardless of IMEM size. A single control field is used for each access based on the upper two address bits used to index into the IMEM. ISXAC3 is used when these two bits = 2'b11 00 Supervisor-mode CPU instruction fetch accesses are allowed to quadrant 3 of IMEM 01 Supervisor-mode CPU instruction fetch accesses are not allowed to quadrant 3 of IMEM 10 Supervisor-mode CPU instruction fetch accesses are conditionally allowed to quadrant 3 of IMEM based on memory protection logic 11 Reserved
10:11 ISXAC2	IMEM Supervisor Instruction Fetch Access Control for Quadrant 2 **Note:** Individual control fields are provided for four equal sized sections of IMEM regardless of IMEM size. A single control field is used for each access based on the upper two address bits used to index into the IMEM. ISXAC2 is used when these two bits = 2'b10 00 Supervisor-mode CPU instruction fetch accesses are allowed to quadrant 2 of IMEM 01 Supervisor-mode CPU instruction fetch accesses are not allowed to quadrant 2 of IMEM 10 Supervisor-mode CPU instruction fetch accesses are conditionally allowed to quadrant 2 of IMEM based on memory protection logic 11 Reserved
12:13 ISXAC1	IMEM Supervisor Instruction Fetch Access Control for Quadrant 1 **Note:** Individual control fields are provided for four equal sized sections of IMEM regardless of IMEM size. A single control field is used for each access based on the upper two address bits used to index into the IMEM. ISXAC1 is used when these two bits = 2'b01 00 Supervisor-mode CPU instruction fetch accesses are allowed to quadrant 1 of IMEM 01 Supervisor-mode CPU instruction fetch accesses are not allowed to quadrant 1 of IMEM 10 Supervisor-mode CPU instruction fetch accesses are conditionally allowed to quadrant 1 of IMEM based on memory protection logic 11 Reserved

Table 187. IMEMCTL1 fields description (continued)

Field	Description
14:15 ISXAC0	IMEM Supervisor Instruction Fetch Access Control for Quadrant 0 **Note:** Individual control fields are provided for four equal sized sections of IMEM regardless of IMEM size. A single control field is used for each access based on the upper two address bits used to index into the IMEM. ISXAC0 is used when these two bits = 2‘b00 00 Supervisor-mode CPU instruction fetch accesses are allowed to quadrant 0 of IMEM 01 Supervisor-mode CPU instruction fetch accesses are not allowed to quadrant 0 of IMEM 10 Supervisor-mode CPU instruction fetch accesses are conditionally allowed to quadrant 0 of IMEM based on memory protection logic 11 Reserved
24:25 IUXAC3	IMEM User Instruction Fetch Access Control for Quadrant 3 **Note:** Individual control fields are provided for four equal sized sections of IMEM regardless of IMEM size. A single control field is used for each access based on the upper two address bits used to index into the IMEM. IUXAC3 is used when these two bits = 2‘b11 00 User-mode CPU instruction fetch accesses are allowed to quadrant 3 of IMEM 01 User-mode CPU instruction fetch accesses are not allowed to quadrant 3 of IMEM 10 User-mode CPU instruction fetch accesses are conditionally allowed to quadrant 3 of IMEM based on memory protection logic 11 Reserved
26:27 IUXAC2	IMEM User Instruction Fetch Access Control for Quadrant 2 **Note:** Individual control fields are provided for four equal sized sections of IMEM regardless of IMEM size. A single control field is used for each access based on the upper two address bits used to index into the IMEM. IUXAC2 is used when these two bits = 2‘b10 00 User-mode CPU instruction fetch accesses are allowed to quadrant 2 of IMEM 01 User-mode CPU instruction fetch accesses are not allowed to quadrant 2 of IMEM 10 User-mode CPU instruction fetch accesses are conditionally allowed to quadrant 2 of IMEM based on memory protection logic 11 Reserved

Table 187. IMEMCTL1 fields description (continued)

Field	Description
28:29 IUXAC1	MEM User Instruction Fetch Access Control for Quadrant 1 **Note:** Individual control fields are provided for four equal sized sections of IMEM regardless of IMEM size. A single control field is used for each access based on the upper two address bits used to index into the IMEM. IUXAC1 is used when these two bits = 2‘b01 00 User-mode CPU instruction fetch accesses are allowed to quadrant 1 of IMEM 01 User-mode CPU instruction fetch accesses are not allowed to quadrant 1 of IMEM 10 User-mode CPU instruction fetch accesses are conditionally allowed to quadrant 1 of IMEM based on memory protection logic 11 Reserved
30:31 IUXAC0	IMEM User Instruction Fetch Access Control for Quadrant 0 **Note:** Individual control fields are provided for four equal sized sections of IMEM regardless of IMEM size. A single control field is used for each access based on the upper two address bits used to index into the IMEM. IUXAC0 is used when these two bits = 2‘b00 00 User-mode CPU instruction fetch accesses are allowed to quadrant 0 of IMEM 01 User-mode CPU instruction fetch accesses are not allowed to quadrant 0 of IMEM 10 User-mode CPU instruction fetch accesses are conditionally allowed to quadrant 0 of IMEM based on memory protection logic 11 Reserved

12.10 End-to-End ECC Support

This section describes end-to-end error detection and correction capability (e2eECC) enhancements to address additional diagnostic capabilities for the next generation of embedded designs.

12.10.1 End-to-End ECC Control and Configuration

e2eECC is controlled by a set of privileged device control registers (DCRs) accessed using the **mfdcr** and **mtdcr** instructions. These registers and the operation of various features are described in the following subsections.

12.10.1.1 End-to-End ECC Control Register 0 (E2ECTL0)

The End-to-End ECC Control Register 0 (E2ECTL0) controls operation of the e2eECC logic.

Offset: Refer to *Figure 65* (DCR - 510) Access: Read/Write Privileged

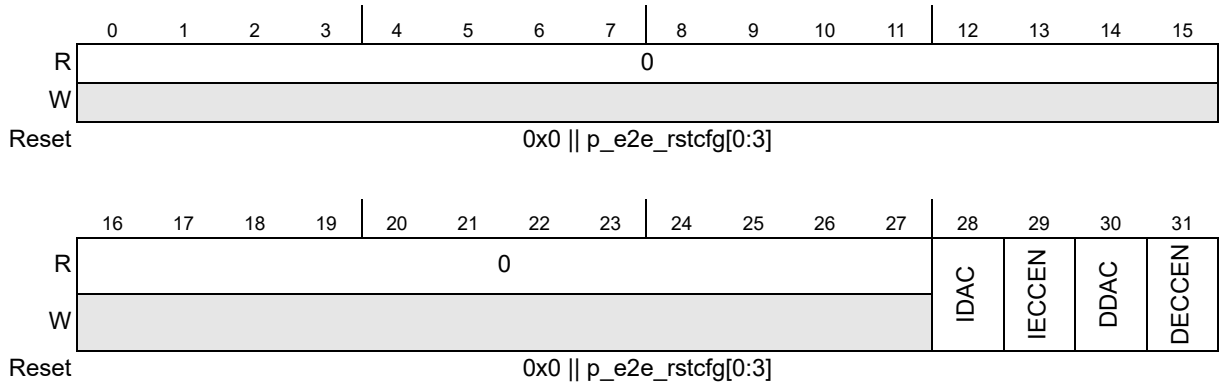


Figure 93. e2eECC Control Register 0 (E2ECTL0)

Table 188. E2ECTL0 fields description

Field	Description
28 IDAC	Instruction Disable Address Checking This bit controls usage of the address portion of the ECC H matrix. When use is disabled, the address portion is not used for checkbit/syndrome generation for Instruction AHB accesses. **Note:** This field is updated on **p_reset_b** based on the value of the **p_e2e_rstcfg[0]** input. 0 Use of access address is enabled in checkbit and syndrome generation. 1 Use of access address is disabled in checkbit and syndrome generation.
29 IECCEN	Instruction ECC Enable Field **Note:** This field is updated on **p_reset_b** based on the value of the **p_e2e_rstcfg[1]** input. 0 End-to-End ECC is disabled for the Instruction interface. No checking of instruction fetch data is performed. 1 End-to-End ECC is enabled for performing error detection and correction on the Instruction interface. Checking of instruction fetch data is performed with correction of single-bit data errors. Detection of any address errors or uncorrectable multi-bit data errors cause a machine check to be generated if the instruction fetch information is attempted to be used for instruction decoding and execution.

Table 188. E2ECTL0 fields description (continued)

Field	Description
30 DDAC	Data Disable Address Checking This bit controls usage of the address portion of the ECC H matrix. When use is disabled, the address portion is not used for checkbit/syndrome generation for Data AHB accesses. **Note:** This field is updated on **p_reset_b** based on the value of the **p_e2e_rstcfg[2]** input. 0 Use of access address is enabled in checkbit and syndrome generation. 1 Use of access address is disabled in checkbit and syndrome generation.
31 DECCEN	Data ECC Enable Field **Note:** This field is updated on **p_reset_b** based on the value of the **p_e2e_rstcfg[3]** input. 0 End-to-End ECC is disabled for the Data interface. No checking of read data is performed. Writes will still generate the proper check bit values to be supplied to external storage devices. 1 End-to-End ECC is enabled for performing error detection and correction on the Data interface. Checking of read data is performed with correction of single-bit data errors. Detection of any address errors, or uncorrectable multi-bit data errors cause a machine check to be generated. Writes generate the proper check bit values to be supplied to external storage devices.

e2eECC is enabled by setting the [I,D]ECCEN bit of the E2ECTL0 DCR to a 1. On reset, e2eECC operation may be disabled from detecting or correcting errors based on reset configuration inputs, and no exceptions will be signaled for non-zero calculated syndrome values. This allows for the proper initialization of stored checkbits in any volatile storage by the CPU without causing error conditions due to uninitialized storage. Following the initialization, software may enable e2eECC operation by writing the appropriate values to the $E2ECTL0_{[I,D]ECCEN}$ control bits.

The $E2ECTL0_{[I,D]DAC}$ control bits can be used to remove the address checking component of the ECC logic for the Instruction AHB and Data AHB when address inclusion is not desired.

12.10.1.2 End-to-End ECC Error Generation by Software

Error generation provides a way to test error recovery and portions of the error detection/correction logic by intentionally injecting parity errors into the driven checkbit information on the external bus during write operations. No provision is made to generate internal errors on read data due to timing issues; instead, software may intentionally generate errors in the checkbit values to emulate data, address or checkbit error on external bus write cycles, and read back the written data to cause an error to be detected.

The End-to-End ECC Error Control/Status Register (E2EECSR0) controls operation of the e2eECC error injection logic. It allows for generation of a single error injection operation on the next CPU data write to the external bus only, or for a series of error injection write operations to the external bus, using the WRC and INVC control fields. The WRCHKBIT/CHKINVT field controls which of the **p_hwchkbit[7:0]** outputs are affected.

Control is provided to allow for either inverting the value(s) of one or more of the checkbit outputs **p_hwchkbit[7:0]** in hardware on one or more external write accesses, or for software to directly supply checkbit values to be used for the external write access(es).

The End-to-End ECC Error Control/Status Register (E2EECSR0) also supports access to raw checkbit values via the RCHKBIT status field. This field is updated with checkbit values received on each CPU data read operation to either the DMEM or to the external bus (but not on cache hits).

Note that Nexus 3 accesses do not cause error injection or the capture of read checkbits, regardless of the settings of the E2EECSR0 register.

Figure 94 shows the layout of E2EECSR0.

Offset: Refer to *Figure 65* (DCR - 511) Access: Read/Write Privileged

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0				RCHKBIT								0			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	WRC		INVC		0				WCHKBIT/CHKINVT							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 94. e2eECC Error Control/Status Register 0 (E2EECSR0)

Table 189. E2EECSR0 fields description

Field	Description
4:11 RCHKBIT	Read Checkbits This field provides the raw checkbits received on the last CPU data access to the DMEM or to external memory via the Data BIU. This field corresponds bit-wise to **p_hrchkbit[7:0]**. for external reads and to **p_dchk[0:7]** for DMEM read accesses. Software may use this information to determine the stored checkbits of external memories or the DMEM by performing CPU load operations and then accessing this field prior to the next load operation (assuming interrupts are masked).
16:17 WRC	Write Control **Note:** Checkbit substitution has priority over Error injection 00 No write checkbit substitution is performed by this control bit 01 Checkbit substitution is performed for the next CPU external write access (only) by driving **p_hwchkbit[7:0]** with the values in the WCHKBIT control field. Software must clear this field before rewriting to 01 in order to generate a subsequent checkbit substitution operation. 10 Checkbit substitution is performed for each CPU external write access while this field remains set to 10 by driving **p_hwchkbit[7:0]** with the values in the WCHKBIT control field. This field must be cleared by software to discontinue further checkbit substitution. 11 Reserved

Table 189. E2EECSR0 fields description (continued)

Field	Description
18:19 INVC	Invert Control **Note:** Checkbit substitution has priority over Error injection 00 No error injection is performed by this control bit 01 Error injection is performed for the next CPU external write access (only) by modifying **p_hwchkbit[7:0]** based on CHKINVT. Software must clear this field before rewriting to 01 in order to generate a subsequent error injection operation. 10 Error injection is performed for each CPU external write access while this field remains set to 10 by modifying **p_hwchkbit[7:0]** based on CHKINVT. This field must be cleared by software to discontinue further error generation. 11 Reserved
24:31 WCHKBIT/CHKINVT	Write Checkbits / Checkbit Invert Mask This field provides the checkbit substitution values when performing checkbit substitution via the WRC control field, and controls which checkbits are inverted when performing error injection via the INVC control field. This field corresponds bit-wise to **p_hwchkbit[7:0]**. For Checkbit inversion operations via error injection: 0 Checkbit is driven normally 1 Checkbit is inverted before being driven out on **p_hwchkbit[7:0]** when error injection occurs.

13 Crossbar switch (XBAR)

13.1 Introduction

This chapter provides information on the layout, configuration, and programming of the crossbar switch. The crossbar switch connects bus masters and bus slaves using a hardware interconnect matrix. This structure allows all bus masters to access different bus slaves simultaneously with no interference while providing arbitration among the bus masters when they access the same slave. A variety of bus arbitration methods and attributes may be programmed on a slave-by-slave basis.

13.1.1 Features

The crossbar switch includes these distinctive features:

- Symmetric crossbar bus switch implementation
  - Concurrent accesses from different masters to different slaves
  - Configurable slave arbitration attributes on a slave-by-slave basis
- 32-bit datapath width
- Support for 8-, 16-, and 32-bit single transfers
- Support for a variety of 4-, 8-, and 16-beat burst transfers
- Operation at one-to-one clock frequency with the bus masters
- Support for low-power park mode

13.1.2 Memory map and register definition

Each slave port of the crossbar switch contains configuration registers. Read and write transfers of the configuration registers require two bus clock cycles. The registers can only be read from and written to in supervisor mode. Additionally, these registers can only be read from or written to by 32-bit accesses.

A bus error response is returned if an unimplemented location is accessed within the crossbar switch.

The slave registers also feature a bit that, when set, prevents the registers from being written. The registers remain readable, but future write attempts have no effect on the registers and are terminated with a bus error response to the master initiating the write. The core, for example, takes a data storage interrupt.

Note: This section shows the registers for all eight master and slave ports. If a master or slave is not used on this particular device, then unexpected results occur when writing to its registers. See the Chapter 7: Device configuration: Section 7.3.2.1: Crossbar switch (XBAR) configuration for the exact master/slave assignments for your device. Additionally, all references to the crossbar switch registers are based on the physical port connections, not the logical port numbers.

Table 190. XBAR memory map

Offset address (hex)	Register name	Width (in bits)	Access	Location
SLAVE channel n (n=0 to 4)				
0x0000 + *n**0x100	XBAR Priority Register Slave *n* (XBAR_PRS*n*)	32	R	*Section 13.1.2.1*
0x0010 + *n**0x100	XBAR0 Control Register *n* (XBAR_CRS*n*)	32	R/W	*Section 13.1.2.2*

13.1.2.1 XBAR Priority Registers Slave (XBAR_PRS*n*)

The priority registers (XBAR*x*_PRS*n*) set the priority of each master port on a per-slave-port basis and reside in each slave port. The priority register can be accessed only with 32-bit accesses. The XBAR*x*_PRS*n* register is read-only; attempts to write to it have no effect on it and result in a bus-error response to the master initiating the write.

See *Chapter 7: Device configuration*: *Section 7.3.2: Crossbar switch* for XBAR*x*_PRS*n* priority values.

Offset: 0x0000 + *n**0x100 (*n* = 0 to 4) Access: Supervisor read-only

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	M3			0	M2			0	M1			0	M0		
W																
Reset	0	0	0	0	0	0	0	1	0	0	1	0	0	0	1	1

Figure 95. Priority Registers Slave (XBAR_PRS*n*)

Table 191. XBAR_PRS*n* field descriptions

Field	Description
17:19 M3	Master 3 priority Sets the arbitration priority for this port on the associated slave port: 000 This master has level 1 (highest) priority when accessing the slave port. 001 This master has level 2 priority when accessing the slave port. 010 This master has level 3 priority when accessing the slave port. 011 This master has level 4(lowest) priority when accessing the slave port.
21:23 M2	Master 2 priority Sets the arbitration priority for this port on the associated slave port: 000 This master has level 1 (highest) priority when accessing the slave port. 001 This master has level 2 priority when accessing the slave port. 010 This master has level 3 priority when accessing the slave port. 011 This master has level 4(lowest) priority when accessing the slave port.
25:27 M1	Master 1 priority Sets the arbitration priority for this port on the associated slave port: 000 This master has level 1 (highest) priority when accessing the slave port. 001 This master has level 2 priority when accessing the slave port. 010 This master has level 3 priority when accessing the slave port. 011 This master has level 4(lowest) priority when accessing the slave port.
29:31 M0	Master 0 priority Sets the arbitration priority for this port on the associated slave port: 000 This master has level 1 (highest) priority when accessing the slave port. 001 This master has level 2 priority when accessing the slave port. 010 This master has level 3 priority when accessing the slave port. 011 This master has level 4(lowest) priority when accessing the slave port.

13.1.2.2 XBAR Control Register (XBAR_CRS*n*)

Offset: 0x0010 + *n**0x100 (*n* = 0 to 4) Access: Supervisor read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	RO	HRP	0	0	0	0	0	0	0	0	0	0	HPE3	HPE2	HPE1	HPE0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	ARB		0		PCTL		0	Reserved(1)	PARK	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. This bit is RW and must not be used.

Figure 96. XBAR Control Register (XBAR_CRS*n*)

Table 192. XBAR_CRS*n* field descriptions

Field	Description
0 RO	Read only Forces both of the slave port's registers to be read-only. After set, only a hardware reset clears it. 0 The slave port's registers are writable. 1 The slave port's registers are read-only and cannot be written (attempted writes have no effect on the registers and result in a bus error response).
1 HRP	Halt Request Priority Determines if a request to halt the crossbar is treated as the highest priority request, or the lowest priority request. 0 The halt request has the highest priority for arbitration on this slave port. 1 The halt request has the lowest initial priority for arbitration on this slave port.
12:15 HPE[3:0]	High Priority Enable Determines if Master x is able to temporarily elevate its request to the slave to high priority status. This can help reduce the amount of time the high priority requesting mster must wait to gain control of the slave. 0 High Priority requests from Master x are ignored and are treated the same as regular requests. 1 High Priority requests from Master x can temporarily elevate the priority level of the master's request to the slave.
22:23 ARB	Arbitration mode Selects the arbitration policy for the slave port. 00 Fixed priority 01 Round-robin (rotating) priority 10 Reserved 11 Reserved
26:27 PCTL	Parking control Determines the slave port's parking control. The low-power park feature results in overall power savings if the slave port is not saturated; however, this forces an extra latency clock when any master tries to access the slave port while not in use because it is not parked on any master. 00 When no master makes a request, the arbiter parks the slave port on the master port defined by the PARK bit field. 01 When no master makes a request, the arbiter parks the slave port on the last master to be in control of the slave port. 10 When no master makes a request, the slave port is not parked on a master and the arbiter drives all outputs to a constant safe state. 11 Reserved
30:31 PARK	Park Determines which master port the current slave port parks on when no masters are actively making requests and the PCTL bits are cleared. **Note:** Only select master ports that are actually present on the device. If not, undefined behavior may occur. 00 Park on master port M0 01 Park on master port M1 10 Park on master port M2 11 Park on master port M3

13.2 Functional description

This section provides the functional details of the crossbar switch.

13.2.1 General operation

When a master sends an access to the crossbar switch, the access is immediately taken. If the targeted slave port of the access is available, then the access is immediately presented on the slave port. It is possible to make single-clock (zero wait state) accesses through the crossbar. If the targeted slave port of the access is busy or parked on a different master port, the requesting master simply sees wait states inserted until the targeted slave port can service the master's request. The latency in servicing the request depends on each master's priority level and the responding slave's access time.

Since the crossbar switch appears to be just another slave to the master device, the master device has no knowledge of whether it actually owns the slave port it is targeting. While the master does not have control of the slave port it is targeting, it simply enters a wait state.

A master is given control of the targeted slave port only after a previous access to a different slave port completes, regardless of its priority on the newly targeted slave port. This prevents deadlock from occurring when:

- A higher priority master has:
  - An outstanding request to one slave port that has a long response time and
  - A pending access to a different slave port, and
- A lower priority master is also making a request to the same slave port as the pending access of the higher priority master.

After the master has control of the slave port it is targeting, the master remains in control of that slave port until it gives up the slave port by running an IDLE cycle or by leaving that slave port for its next access. The master could also lose control of the slave port if another higher priority master makes a request to the slave port; however, if the master is running a fixed-length burst transfer it retains control of the slave port until that transfer completes. When a master has control of a given slave port, the crossbar returns all response information from the slave back to the requesting master.

The crossbar terminates all master IDLE transfers (as opposed to allowing the termination to come from one of the slave buses). Additionally, when no master is requesting access to a slave port, the crossbar drives IDLE transfers onto the slave bus even though a default master may be granted access to the slave port.

When a slave bus is being IDLEd by the crossbar, it can park the slave port on the master port indicated by the XBAR*x*_CRS*n*[PARK]. This is done in an attempt to save the initial clock of arbitration delay that otherwise would be seen if the master had to arbitrate to gain control of the slave port. The slave port can also be put into low-power park mode in attempt to save power by using XBAR*x*_CRS*n*[PCTL].

13.2.2 Register coherency

Since the content of the registers has a real-time effect on the operation of the crossbar, it is important to understand that any register modifications take effect as soon as the register is written. The values of the registers do not track with slave-port-related master accesses; instead, they track only with slave accesses.

13.2.3 Arbitration

The crossbar switch supports two arbitration schemes: a simple fixed-priority comparison algorithm and a simple round-robin fairness algorithm. The arbitration scheme is independently programmable for each slave port.

13.2.3.1 Fixed-priority operation

When operating in fixed-priority mode, each master is assigned a unique priority level in the XBAR*x*_PRS*n* (priority registers). If two masters request access to a slave port, the master with the higher priority in the selected priority register gains control over the slave port.

When a master makes a request to a slave port, the slave port checks whether the new requesting master's priority level is higher than that of the master that currently has control over the slave port (unless the slave port is in a parked state). The slave port performs an arbitration check at every clock edge to ensure that the proper master (if any) has control of the slave port.

If the new requesting master's priority level is higher than that of the master that currently has control of the slave port, the new requesting master is granted control over the slave port at the next clock edge. The exception to this rule is if the master that currently has control over the slave port is running a fixed-length burst transfer. In this case, the new requesting master must wait until the end of the burst transfer or locked transfer before it is granted control of the slave port.

If the new requesting master's priority level is lower than the master that currently has control of the slave port, the new requesting master is forced to wait until the current master runs one of the following cycles:

- An IDLE cycle
- A non-IDLE cycle to a location other than the current slave port

13.2.3.2 Round-robin priority operation

When operating in round-robin mode, each master is assigned a relative priority based on the physical master port number. This relative priority is compared to the ID (master port number) of the last master to perform a transfer on the slave bus. The highest priority requesting master becomes owner of the slave bus at the next transfer boundary (accounting for fixed-length burst transfers). Priority is based on how far the ID of the requesting master is ahead of the ID of the last master.

After access is granted to a slave port, a master may perform as many transfers as desired to that port until another master makes a request to the same slave port. The next master in line is granted access to the slave port at the next transfer boundary, or possibly on the next clock cycle if the current master has no pending access request.

As an example of arbitration in round-robin mode, assume the crossbar is implemented with master ports 0, 1, 4, and 5. If the last master of the slave port was master 1, and master 0, 4, and 5 make simultaneous requests, they are serviced in the order 4, 5, and then 0.

Parking may continue to be used in a round-robin mode, but it does not affect the round-robin pointer unless the parked master actually performs a transfer. Handoff occurs to the next master in line after one cycle of arbitration. If the slave port is put into low-power park mode, the round-robin pointer is reset to point at master port 0, giving it the highest priority.

13.3 Initialization/application information

No initialization is required by or for the crossbar switch. Hardware reset ensures all the register bits used by the crossbar switch are properly initialized to a valid state. Settings and priorities should be programmed to achieve maximum system performance.

14 System Integration Unit Lite2 (SIUL2)

14.1 Introduction

14.1.1 Overview

The System Integration Unit Lite2 (SIUL2) provides control over all the I/O ports on this device and supports five ports with 16 bit of bidirectional, general-purpose input and output signals. It supports six external interrupts with independent trigger event configuration, that are combined to generate four internal interrupts versus the interrupt controller. For more details see *Section 14.3.4: External interrupts/DMA requests (EIRQ pins)*. *Figure 97* is the block diagram of SIUL2 and its interfaces to other system components for this device.

The module provides dedicated general-purpose pads that can be configured as either inputs or outputs.

- When configured as output, you can write to an internal register to control the state driven on the associated output pad.
- When configured as input, you can detect the state of the associated pad by reading the value from an internal register.
- When configured as input and output, the pad value can be read back, which can be used as a method of checking if the written value appeared on the pad.

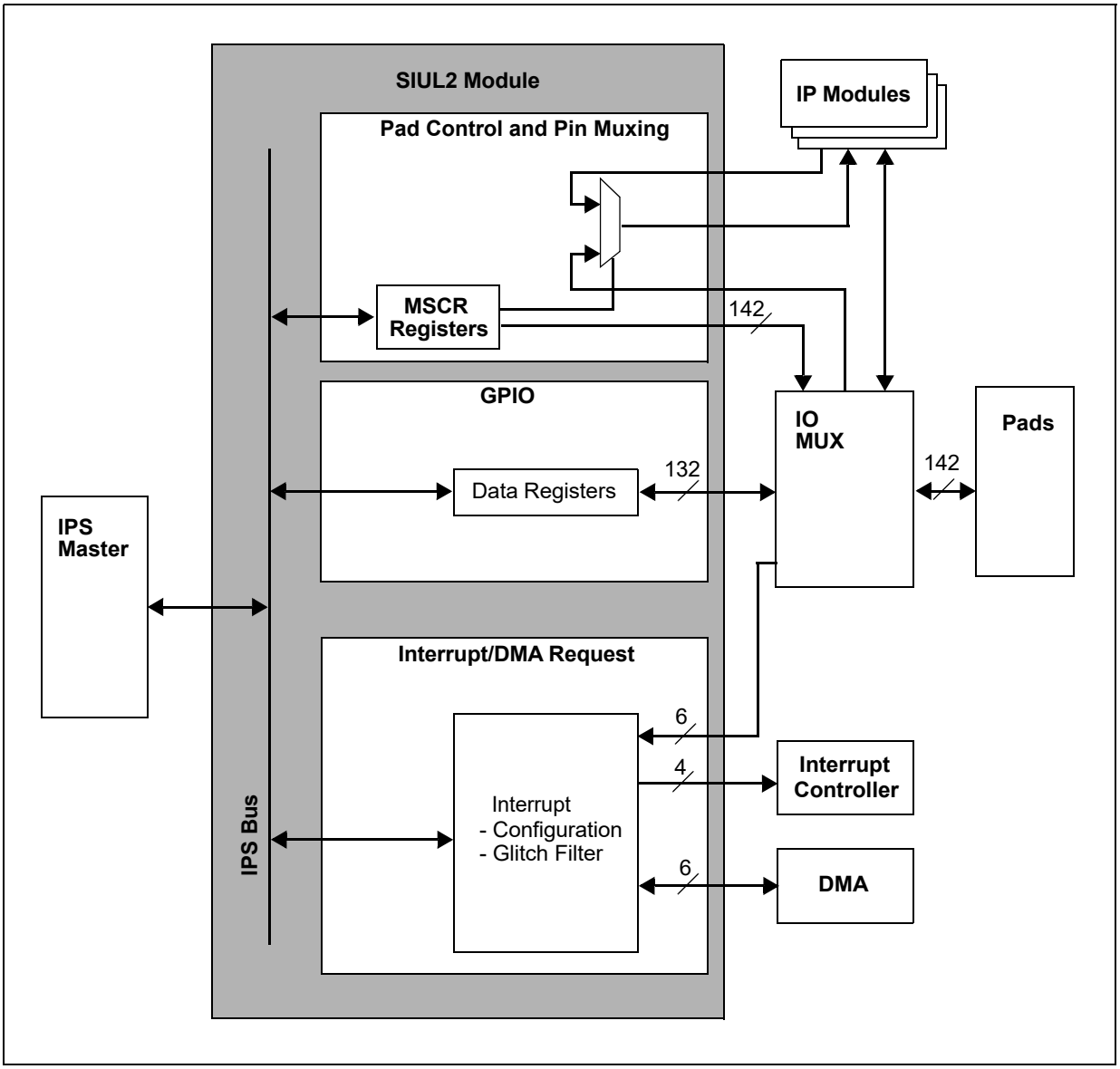
In order to assist in software development, there are three different access mechanisms to the GPIO data registers:

- Access to two 16-bits ports in one access
- Read/Write access to a single bit
- A 16-bits port write with a bit mask, using single 32-bits access

These differing mechanisms allow support for port access or for bit manipulation without the need to use read-modify-write operations or access with Power Architecture reservation.

The external interrupt sources can be configured on MCU level to be used with any chip pad. The six interrupt sources can be configured to have a digital filter to reject short glitches on the inputs. The external interrupt/DMA requests map to the six EIRQ pins in the device packages.

Figure 97. SIUL2 block diagram



14.1.2 Features

The SIUL2 supports these distinctive features:

- Five GPIO ports with data control
  - Drive data to up to 16 independent I/O channels
  - Sample data from up to 16 independent I/O channels
- Two 16-bits registers can be read/written with one access to a 32-bits port if needed.
- External interrupt/DMA request support with:
  - Four system interrupt vectors for 6 interrupt sources (six connected to EIRQ pins) with independent interrupt mask
  - Six programmable digital glitch filters (one for each EIRQ)
  - Independent DMA channels for each EIRQ pin
  - Edge detection

Additionally the SIUL2 contains the multiplexed signal configuration registers that configure the electrical parameters and settings of all functional pads (132 GPIO + 10 dedicated pads)[g]. These are used to configure the following pad features:

- Drive strength
- Output impedance control
- Open drain/source output enable
- Slew rate control of JTAG pins
- Internal weak pull control
- Pin function assignment
- Control of analog path switches
- Safe mode behavior configuration
- TTL or CMOS or Automotive input levels

14.1.3 Register protection

The System Integration Unit Lite2 uses the Register Protection Scheme to protect the individual registers from accidental writes.Details about the protected registers at specific addresses can be found in *Chapter 67: Register Protection (REG_PROT)* and *Chapter 7: Device configuration*.

14.2 Memory map and register description

14.2.1 Memory map

Table 193 gives an overview of the SIUL2 registers implemented.

g. Refer to the *Chapter 7: Device configuration* for details on package-specific pin and register implementation.

Table 193. SIUL2 memory map

Address offset	Register name	Access	Size (bits)	Supported access (bit)	Reset value	Register classification (1)	Location
0x0000-0x0003	Reserved						
0x0004	SIUL2 MCU ID Register #1 (SIUL2_MIDR1)	R	32	32	0x5701_X400	Saf-Relv	*Section 14.2.2.1*
0x0008	SIUL2 MCU ID Register #2 (SIUL2_MIDR2)	R	32	32	0xA800_530F	Saf-Relv	*Section 14.2.2.2*
0x000C–0x000F	Reserved						
0x0010	SIUL2 DMA/Interrupt Status Flag Register 0 (SIUL2_DISR0)	R/W	32	32/ 16/8	0x0000_0000	Saf-Relv	*Section 14.2.2.3*
0x0014–0x0017	Reserved						
0x0018	SIUL2 DMA/Interrupt Request Enable Register 0 (SIUL2_DIRER0)	R/W	32	32/ 16/8	0x0000_0000	Saf-Relv	*Section 14.2.2.4*
0x001C–0x001F	Reserved						
0x0020	SIUL2 DMA/Interrupt Request Select Register 0 (SIUL2_DIRSR0)	R/W	32	32/ 16/8	0x0000_0000	Saf-Relv	*Section 14.2.2.5*
0x0024-0x0027	Reserved						
0x0028	SIUL2 Interrupt Rising-Edge Event Enable Register 0 (SIUL2_IREER0)	R/W	32	32/ 16/8	0x0000_0000	Saf-Relv	*Section 14.2.2.6*
0x002C-0x002F	Reserved						
0x0030	SIUL2 Interrupt Falling-Edge Event Enable Register 0 (SIUL2_IFEER0)	R/W	32	32/ 16/8	0x0000_0000	Saf-Relv	*Section 14.2.2.7*
0x0034-0x0037	Reserved						
0x0038	SIUL2 Interrupt Filter Enable Register 0 (SIUL2_IFER0)	R/W	32	32 /16/8	0x0000_0000	Saf-Lat	*Section 14.2.2.8*
0x003C-0x003F	Reserved						

Table 193. SIUL2 memory map (continued)

Address offset	Register name	Access	Size (bits)	Supported access (bit)	Reset value	Register classification (1)	Location
0x0040–0x00BC	SIUL2 Interrupt Filter Maximum Counter Registers (SIUL2_IFMCR0–SIUL2_IFMCR31)	R/W	32	32	0x0000_0000	Saf-Lat	*Section 14.2.2.9*
0x00C0	SIUL2 Interrupt Filter Clock Prescaler (SIUL2_IFCPR)	R/W	32	32	0x0000_0000	Saf-Lat	*Section 14.2.2.10*
0x00C4–0x023F	Reserved(7)						
0x0240–0x0A3C	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR0–SIUL2_MSCR511)(2)	R/W	32	32	0x0080_0000	Saf-Relv	*Section 14.2.2.11*
0x0A40–0x123C	Multiplexed Signal Configuration Registers for Multiplexed Input Selection (SIUL2_MSCR512–SIUL2_MSCR1023)(2)	R/W	32	32	0x0000_0000	Saf-Relv	*Section 14.2.2.12*
0x1240–0x12FF	Reserved						
0x1300–0x1383	SIUL2 GPIO Pad Data Out Register 0 – GPIO Pad Data Out Register 131 (SIUL2_GPDO0 – SIUL2_GPDO131)	R/W	8	8	0x00	Saf-Relv	*Section 14.2.2.16*
0x1384–0x14FF	Reserved						
0x1500–0x1583	SIUL2 GPIO Pad Data In Register 0 – GPIO Pad Data In Register 131 (SIUL2_GPDI0 – SIUL2_GPDI131)	R	8	8	0x00	Saf-Relv	*Section 14.2.2.17*
0x1584–0x16FF	Reserved						
0x1700–0x1710	SIUL2 Parallel GPIO Pad Data Out Register (SIUL2_PGPDO0 – SIUL2_PGPDO8)	R/W	16	16/8	0x0000	Saf-Relv	*Section 14.2.2.18*
0x1712–0x173F	Reserved						

Table 193. SIUL2 memory map (continued)

Address offset	Register name	Access	Size (bits)	Supported access (bit)	Reset value	Register classification (1)	Location
0x1740–0x1750	SIUL2 Parallel GPIO Pad Data In Register (SIUL2_PGPDI0 – SIUL2_PGPDI8)	R	16	16/8	0x0000	Saf-Relv	*Section 14.2.2.19*
0x1752–0x177F	Reserved						
0x1780–0x17A0	SIUL2 Masked Parallel GPIO Pad Data Out Register (SIUL2_MPGPDO0 – SIUL2_MPGPDO8)	R/W	32	32(3)	0x0000_0000	Saf-Relv	*Section 14.2.2.20*
0x17A4–0x17FF	Reserved(4)						

1. Saf-Relv: Safety Relevant; Saf-Lat: Safety Latent.
2. For implemented registers MSCR, please refer to Pin Multiplexing table description table and Peripheral multiplexing table description in *Chapter 4: Signal description*.
3. 32-bit writes, 8/16/32-bit reads are allowed. (Only 8-bit and 16-bit writes are not supported and generate a transfer error.)
4. Writes are possible without having an impact on SoC functionality (No transfer errors are generated).

Note: Reserved registers will be read as 0. If supported and enabled in this MCU, a transfer error will be issued when trying to access completely reserved register space.

14.2.2 Register descriptions

This section describes in address order all the SIUL2 registers.

14.2.2.1 SIUL2 MCU ID Register #1 (SIUL2_MIDR1)

This register contains identification information about the device.

Offset: 0x0004 Access: Read-only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	PARTNUM															
W																
Reset	0	1	0	1	0	1	1	1	0	1	0	0	0	0	1	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	ED	PKG					0	0	MAJOR_MASK				MINOR_MASK			
W																
Reset	0	0(1)	X(1)	X	0(1)	1(1)	0	0	X(2)	X(2)	X(2)	X(2)	X(2)	X(2)	X(2)	X(2)

1. Only possible to change value by modifying the side band signals (in the Test Flash). Values are set at factory and cannot be modified.
2. Value is set at factory and will vary each revision of device

Figure 98. SIUL2 MCU ID Register #1 (SIUL2_MIDR1)

Table 194. SIUL2_MIDR1 field descriptions

Field	Description
0:15 PARTNUM	MCU Part Number These digits identify the part number of the chip. PARTNUM[15:8] represents Technology: 0x56 90 nm 0x57 55 nm 0x58 40 nm PARTNUM[7:4] represents Core: 0x0 e200z0h 0x2 e200z2h 0x4 e200z4h PARTNUM[3:0] represents Subfamily: 0x2 SPC574Sx
16 ED	Flag if an emulation device is present for calibration. 0 No emulation device present 1 Emulation device present
17:21 PKG	Can by read by software to determine the package type that is used for the particular device: 01001 100-pin QFP 01101 144-pin QFP 01110 244 BGA
24:27 MAJOR_MASK	Major Mask revision
28:31 MINOR_MASK	Minor Mask revision

14.2.2.2 SIUL2 MCU ID Register #2 (SIUL2_MIDR2)

This register contains identification information about the device.

Offset: 0x0008 Access: Read-only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	SUPPLIER	FLASH_SIZE_1			FLASH_SIZE_2				0	0	0	0	0	0	0	0
W																
Reset	1	0(1)	1(1)	1(1)	0(1)	0(1)	0(1)	0(1)	0(1)	0(1)	0(1)	0(1)	0(1)	0(1)	0(1)	0(1)

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	FAMILYNUM								0	0	0	0	0	0	0	0
W																
Reset	0(1)	1(1)	0(1)	1(1)	0(1)	0(1)	1(1)	1(1)	0(1)	0(1)	0(1)	0(1)	0(1)	0(1)	0(1)	0(1)

1. Only possible to change value by modifying the side band signals (in the Test Flash). Software can not modify

Figure 99. SIUL2 MCU ID Register #2 (SIUL2_MIDR2)

Table 195. SIUL2_MIDR2 field description

Field	Description
0 SUPPLIER	Manufacturer: 0 Reserved 1 STMicroelectronics
1:4 FLASH_SIZE_1	Coarse granularity for Flash memory size Needs to be combined with FLASH_SIZE_2 to calculate the actual memory size: 0000 16 KB 0001 32 KB 0010 64 KB 0011 128 KB 0100 256 KB 0101 512 KB 0110 1 MB 0111 2 MB 1000 4 MB 1001 8 MB 1010 16 MB 1011 32 MB 1100 64 MB 1101 128 MB 1110 256 MB 1111 512 MB

Table 195. SIUL2_MIDR2 field description (continued)

Field	Description
5:8 FLASH_SIZE_2	Fine granularity for Flash memory size Needs to be combined with FLASH_SIZE_1 to calculate the actual memory size: 0000 0 KB 0001 (base size, as specified by FLASH_SIZE_1) / 8 0010 2 * (base size, as specified by FLASH_SIZE_1) / 8 0011 3 * (base size, as specified by FLASH_SIZE_1) / 8 0100 4 * (base size, as specified by FLASH_SIZE_1) / 8 0101 5 * (base size, as specified by FLASH_SIZE_1) / 8 0110 6 * (base size, as specified by FLASH_SIZE_1) / 8 0111 7 * (base size, as specified by FLASH_SIZE_1) / 8 1000 8 * (base size, as specified by FLASH_SIZE_1) / 8 1001 9 * (base size, as specified by FLASH_SIZE_1) / 8 1010 10 * (base size, as specified by FLASH_SIZE_1) / 8 1011 11 * (base size, as specified by FLASH_SIZE_1) / 8 1100 12 * (base size, as specified by FLASH_SIZE_1) / 8 1101 13 * (base size, as specified by FLASH_SIZE_1) / 8 1110 14 * (base size, as specified by FLASH_SIZE_1) / 8 1111 15 * (base size, as specified by FLASH_SIZE_1) / 8
16:23 FAMILYNUM	ASCII character in MCU part number: 01010011 character “S” (SPC574Sxx) family

14.2.2.3 SIUL2 DMA/Interrupt Status Flag Register 0 (SIUL2_DISR0)

The DMA/Interrupt Status Register contains flag bits that record an event on the external IRQ pins. When an event as defined in IRQ Rising-Edge Event Enable Register (SIUL2_IREER0) and IRQ Falling-Edge Event Enable Register (SIUL2_IFEER0) occurs, the corresponding flag bit is set. The IRQ Flag bit is set irrespective of the corresponding DMA/Interrupt Request Enable bit in DMA/Interrupt Request Enable Register (SIUL2_DIRER0) is enabled. The EIF bit remains set until cleared by software or through the servicing of a DMA request. The EIF bits are cleared by writing a ‘1’ to the bits. A write of ‘0’ has no effect.

This register contains the interrupt flags.

Offset: 0x0010 Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	EIF5	EIF4	EIF3	EIF2	EIF1	EIF0
W											w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 100. SIUL2 DMA/Interrupt Status Flag Register (SIUL2_DISR0)

Table 196. SIUL2_DISR0 field descriptions

Field	Description
26:31 EIF[5:0]	External Interrupt Status Flag x This flag can be cleared only by writing 1. Writing 0 has no effect. If enabled (SIUL2_DIRER[x]), EIF[x] causes an interrupt or DMA request. 0 No interrupt or DMA event has occurred on the pad 1 An interrupt or DMA event as defined by SIUL2_IREER[x] and SIUL2_IFEER[x] has occurred

14.2.2.4 SIUL2 DMA/Interrupt Request Enable Register 0 (SIUL2_DIRER0)

The DMA/Interrupt Request Enable Register enables the assertion of DMA or interrupt request if the corresponding External IRQ Flag bit is set in SIUL2 DMA/Interrupt Status Flag Register 0 (SIUL2_DISR0). The type of request enabled is determined by the corresponding DMA/Interrupt Request Select bit in SIUL2 DMA/Interrupt Request Select Register 0 (SIUL2_DIRSR0). This register enables the interrupt messaging to the interrupt controller.

Offset: 0x0018 Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	EIRE5	EIRE4	EIRE3	EIRE2	EIRE1	EIRE0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 101. SIUL2 DMA/Interrupt Request Enable Register 0 (SIUL2_DIRER0)

Table 197. SIUL2_DIRER0 field descriptions

Field	Description
26:31 EIRE[5:0]	External Interrupt or DMA Request Enable x 0 Interrupt or DMA requests from the corresponding EIF[x] bit are disabled 1 Set EIF[x] bit causes either a DMA or an interrupt request depending on SIUL2_DIRSR

14.2.2.5 SIUL2 DMA/Interrupt Request Select Register 0 (SIUL2_DIRSR0)

The DIRSR selects between the DMA or interrupt request. If the corresponding bits are set in SIUL2 DMA/Interrupt Status Flag Register 0 (SIUL2_DISR0) and SIUL2 DMA/Interrupt Request Enable Register 0 (SIUL2_DIRER0), then the DMA/Interrupt Request Select bit determines whether DMA or an interrupt request is asserted. EIRQ are the external interrupt package pins on the device.

Offset: 0x0020 Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	DIRS5	DIRS4	DIRS3	DIRS2	DIRS1	DIRS0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 102. SIUL2 DMA/Interrupt Request Select Register 0 (SIUL2_DIRSR0)

Table 198. SIUL2_DIRSR0 field descriptions

Field	Description
26:31 DIRS[5:0]	DMA/Interrupt Request Select Register Selects between DMA request or external interrupt request when an edge-triggered event occurs on the corresponding pin. 0 Interrupt request is selected 1 DMA request is selected

14.2.2.6 SIUL2 Interrupt Rising-Edge Event Enable Register 0 (SIUL2_IREER0)

This register is used to enable the rising-edge triggered events on the corresponding interrupt pads.

Offset: 0x0028 Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	IREE5	IREE4	IREE3	IREE2	IREE1	IREE0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 103. SIUL2 Interrupt Rising-Edge Event Enable Register 0 (SIUL2_IREER0)

Table 199. SIUL2_IREER0 field descriptions

Field	Description
26:31 IREE[5:0]	Enable rising-edge events to cause the EIF[x] bit to be set. 0 Rising-edge event is disabled 1 Rising-edge event is enabled

14.2.2.7 SIUL2 Interrupt Falling-Edge Event Enable Register 0 (SIUL2_IFEER0)

This register is used to enable falling-edge triggered events on the corresponding interrupt pads.

Offset: 0x0030 Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	IFEE5	IFEE4	IFEE3	IFEE2	IFEE1	IFEE0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 104. SIUL2 Interrupt Falling-Edge Event Enable Register 0 (SIUL2_IFEER0)

Table 200. SIUL2_IFEER0 field descriptions

Field	Description
26:31 IFEE[5:0]	Enable falling-edge events to cause the EIF[x] bit to be set. 0 Falling-edge event is disabled 1 Falling-edge event is enabled

Note: If both IREE bit and IFEE bit are cleared for the same interrupt source, the interrupt status flag for the corresponding external interrupt will never be set.

14.2.2.8 SIUL2 Interrupt Filter Enable Register 0 (SIUL2_IFER0)

This register is used to enable a digital filter counter on the corresponding interrupt pads to filter out glitches on the inputs.

Offset: 0x0038 Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	IFE5	IFE4	IFE3	IFE2	IFE1	IFE0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 105. SIUL2 Interrupt Filter Enable Register 0 (SIUL2_IFER0)

Table 201. SIUL2_IFER0 field descriptions

Field	Description
26:31 IFE[5:0]	Enable digital glitch filter on the interrupt pad input. 0 Filter is disabled 1 Filter is enabled

14.2.2.9 SIUL2 Interrupt Filter Maximum Counter Register (SIUL2_IFMCR0–SIUL2_IFMCR31)

These registers are used to configure the filter counter associated with each digital glitch filter.

Offset: 0x0040–0x00BC Access: User read/write

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

Figure 106. SIUL2 Interrupt Filter Maximum Counter Register (SIUL2_IFMCR0–SIUL2_IFMCR31)

Table 202. SIUL2_IFMCR0–SIUL2_IFMCR31 field descriptions

Field	Description
28:31 MAXCNT	Maximum Interrupt Filter Counter setting (MAXCNTx) MAXCNT can be 0 to 15 For MAXCNT = 0, 1, 2, the filter behaves as ALL PASS filter For MAXCNT = 3, 4, 5,…15, filter period = TCK * MAXCNTx + n * TCK (n= 0 to 4) TCK: Prescaled filter clock period, which is IRC clock prescaled to IFCP value, specified in SIUL2_IFCPR TIRC: Basic filter clock period 62.5 ns (f = 16 MHz) Note that filter delay is 2 TCK clock cycles more than filter period. In general, TFILTER and TDELAY are not integer multiples of TCK because DIN is asynchronous whereas DOUT is synchronous with respect to the CK clock, and the factor n accounts for this uncertainty in filter period calculation.

14.2.2.10 SIUL2 Interrupt Filter Clock Prescaler Register (SIUL2_IFCPR)

This register is used to configure a clock prescaler which is used to select the clock for all digital filters. Prescaler is applied to the input clock to the SIUL2, which is the system PBRIDGE clock counters in the SIUL2.

Offset: 0x00C0 Access: User read/write

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

Figure 107. SIUL2 Interrupt Filter Clock Prescaler Register (SIUL2_IFCPR)

Table 203. SIUL2_IFCPR field descriptions

Field	Description
28:31 IFCP	Interrupt Filter Clock Prescaler setting Prescaled Filter Clock Period = T(IRC) x (IFCP + 1) T(IRC) is the internal oscillator period. IFCP can be 0 to 15.

14.2.2.11 SIUL2 Multiplexed Signal Configuration Registers (SIUL2_MSCR0–SIUL2_MSCR1023)

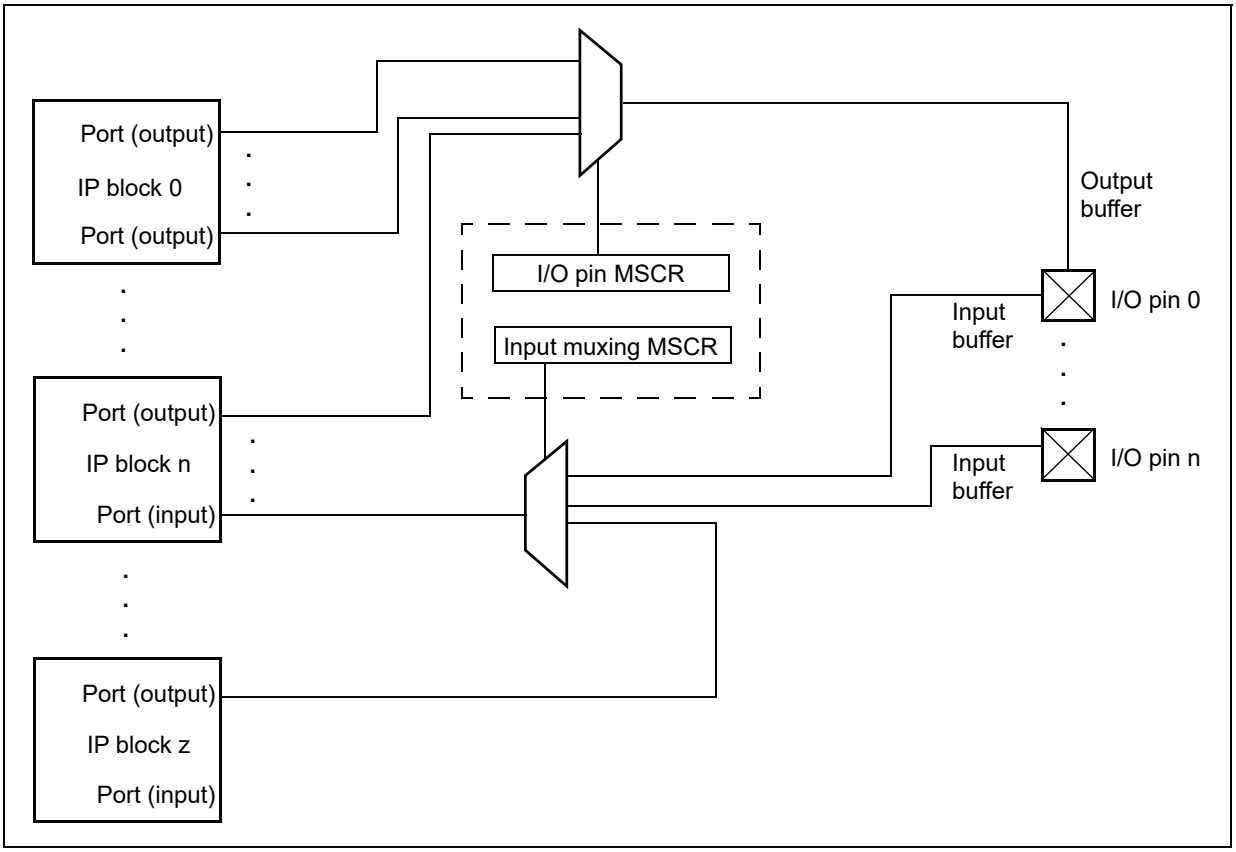
The MSCRs control the I/O pin function and electrical properties for each pin on the device, and are used to select the input signal for IP blocks on the device that have multiple input sources. There is a dedicated MSCR for each I/O pin on the device. Each MSCR is independent of the others, except in the case of an LVDS configuration, where two MSCRs must be configured together in this mode.

The MSCRs are divided into two sets:

- I/O pin control registers (SIUL2_MSCR0–SIUL2_MSCR511)
- Multiplexed input selection registers (SIUL2_MSCR512–SIUL2_MSCR1023)

The signal data flow from IP blocks on the device to and from the I/O pins, and from IP block to IP block is given in *Figure 108*.

Figure 108. MSCR I/O pin and IP block port connectivity



14.2.2.12 I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR0–SIUL2_MSCR511)

The I/O pin MSCR definition is given in *Figure 109*. These registers select the output function on the associated I/O pin. The I/O MSCRs control the drive strength, output drive circuit, pull up/down, input buffer enable, input hysteresis, input level selection, safe-mode operation, and analog input enable of the associated I/O pin.

Note that when an I/O pin is configured as a single-ended input, the input buffer must be enabled for the pin using the MSCR[IBE] bit. This enables the input to all input destinations connected to the pin, including the GPIO input. In this way, the GPIO can be read at any time, regardless of the pin function. For IP blocks that have multiple input sources, the input source is selected in the multiplexed input selection MSCRs, and the associated I/O pin MSCR for the selected input must have the MSCR[IBE] bit enabled.

Offset: 0x0240–0x0A3C Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	OERC			0	0	ODC		SMC	0	ILS		IBE	HYS	WPDE	WPUE
W																
Reset	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	INV	0	0	0	0	0	0	0	SSS							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 109. I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR0–SIUL2_MSCR511)

Table 204. SIUL2_MSCR0–SIUL2_MSCR511 field description

Field	Description
1:3 OERC	Output Edge Rate Control Used only when the associated destination is a chip pin. Specifies the output impedance, drive strength, and slew rate of the associated pin. 00 Weak 01 Medium 10 Strong 11 Very Strong Refer to Output Impedance column of Signal Description table earlier in document for associated pins.
6:7 ODC	Output Drive Control Used only when the associated destination is a chip pin. Specifies the type of output drive control for the associated pin. 00 Disabled (high impedance for turning off the pin or using it as an input) 01 Open-drain (Open drain low-side) 10 Push-pull 11 Open-source (Open drain high-side)

Table 204. SIUL2_MSCR0–SIUL2_MSCR511 field description (continued)

Field	Description
8 SMC	Safe Mode Control Used only when the associated destination is a chip pin. Specifies whether the chip disables the pin's output buffer when the chip enters Safe mode. 0 Disable (the output buffer returns to its previous state when the chip leaves Safe mode) 1 Don't disable
10:11 ILS	Input Level Selection Used only when the associated destination is a chip pin. Specifies the logic family for the associated pin, which determines its logic switching levels. 00 Automotive 01 TTL 10 LVDS 11 CMOS
12 IBE	Input Buffer Enable Used only when the associated destination is a chip pin. Enables the associated pin's input buffer. 0 Disabled 1 Enabled
13 HYS	Input Hysteresis Used only when the associated destination is a chip pin. Enables input hysteresis for the associated pin. 0 Disabled 1 Enabled
14 WPDE	Weak Pulldown Enable Used only when the associated destination is a chip pin. Enables the associated pin's weak pulldown resistor. It is OK for both WPDE and WPUE to be enabled. 0 Disabled 1 Enabled
15 WPUE	Weak Pullup Enable Used only when the associated destination is a chip pin. Enables the associated pin's weak pullup resistor. It is OK for both WPDE and WPUE to be enabled. 0 Disabled 1 Enabled
16 INV	Invert Inverts the signal selected by SSS before transmitting it to the associated destination (chip pin or module port). 0 Don't invert 1 Invert
24:31 SSS	Source Signal Select Selects which source signal is connected to the associated destination (chip pin or module port). For a chip pin, the source signals are outputs from module ports. For a module port, the source signals are either outputs from module ports or inputs from chip pins. The meaning of each value depends on the destination. See *Section 14.2.2.14* and *Section 14.2.2.15*.

All bits and fields in the I/O MSCRs are applicable for all GPIO ports on the device. This includes all ports on the device with digital output function. For ADC input ports with digital input only (no output), the OERC, ODC, SMC, INV, and SSS bits/fields do not apply. The SSS is always 8b0 for input only ports, enabling the general purpose input. The analog input path is enabled in the ADC logic when the channel is selected for conversion.

The I/O state exceptions are given in *Table 205*.

Table 205. I/O MSCR Reset State Exceptions

Port	SMC	HYS	WPDE	WPUE	IBE	SRE1	SRE0
PA[18:33]	0	1	0	0	0	0	0
PA[94:113]	0	1	0	0	0	0	0
PA[129:131]	0	1	0	0	0	0	0
JCOMP	1	1	0	1	1	0	0
TCK	1	1	1	0	1	0	0
TMS	1	1	0	1	1	0	0
TDI	1	1	0	1	1	0	0
TDO	1	1	0	0	0	1	1

14.2.2.13 Multiplexed Signal Configuration Registers for Multiplexed Input Selection (SIUL2_MSCR512–SIUL2_MSCR1023)

The multiplexed input selection MSCR definition is given in *Figure 110*. These registers select the input source for IP blocks on the device that have inputs from multiple sources, both other IP blocks and I/O pins. It is possible to configure the pad to be in input without having to connect it externally. This is possible, either by enabling internal pull-up/pull-down or disabling the input buffer (but in this case data is seen as 0 internally).

Offset: 0xA40–0x123C Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	INV	0	0	0	0	0	0	0	SSS							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 110. Multiplexed Signal Configuration Registers for Multiplexed Input Selection (SIUL2_MSCR512–SIUL2_MSCR1023)

Table 206. SIUL2_MSCR512–SIUL2_MSCR1023 field descriptions

Field	Description
16 INV	Invert Inverts the signal selected by SSS before transmitting it to the associated destination (chip pin or module port). 0 Don't invert 1 Invert
24:31 SSS	Source Signal Select Selects which source signal is connected to the associated destination (chip pin or module port). For a chip pin, the source signals are outputs from module ports. For a module port, the source signals are either outputs from module ports or inputs from chip pins. The meaning of each value depends on the destination. See *Section 14.2.2.14* and *Section 14.2.2.15*.

14.2.2.14 Chip-pin MSCR assignments and related information

For chip-pin MSCR assignments, pin types, and SSS values, please refer to *Chapter 4: Signal description*.

14.2.2.15 Module-port MSCR assignments and SSS values

For each internal module port that is or can be configured as an input and has more than one possible source, *Chapter 4: Signal description*, provides the assigned MSCR, and the SSS values and their corresponding source signals.

14.2.2.16 SIUL2 GPIO Pad Data Out Register (SIUL2_GPDO0–SIUL2_GPDO131)

These registers can be used to set or clear a single GPIO pad with a byte access. See *Chapter 4: Signal description* for mapping of ports to GPDO registers.

Offset: 0x1300-0x1383 Access: User read/write

	0	1	2	3	4	5	6	7
R	0	0	0	0	0	0	0	PDO
W								
Reset	0	0	0	0	0	0	0	0

Figure 111. SIUL2 GPIO Pad Data Out Register (SIUL2_GPDO0–SIUL2_GPDO131)

Table 207. SIUL2_GPDO0–SIUL2_GPDO131 field descriptions

Field	Description
7 PDO	Pad Data Out This bit stores the data to be driven out on the external GPIO pad controlled by this register. 0 Logic low value is driven on the corresponding GPIO pad when the pad is configured as an output 1 Logic high value is driven on the corresponding GPIO pad when the pad is configured as an output

14.2.2.17 SIUL2 GPIO Pad Data Input Registers (SIUL2_GPDI0–SIUL2_GPDI131)

These registers can be used to read the GPIO pad data with a byte access. See *Chapter 4: Signal description* for mapping of ports to GPDI registers.

Offset: 0x1500-0x1583 Access: User read/write

	0	1	2	3	4	5	6	7
R	0	0	0	0	0	0	0	PDI
W								
Reset	0	0	0	0	0	0	0	0

Figure 112. SIUL2 GPIO Pad Data Input Registers (SIUL2_GPDI0–SIUL2_GPDI131)

Table 208. SIUL2_GPDI0–SIUL2_GPDI131 field descriptions

Field	Description
7 PDI	Pad Data In This bit stores the value of the external GPIO pad associated with this register. 0 The value of the data in signal for the corresponding GPIO pad is logic low 1 The value of the data in signal for the corresponding GPIO pad is logic high

14.2.2.18 SIUL2 Parallel GPIO Pad Data Out Register (SIUL2_PGPDO*n*)

These registers are used to set or clear the respective pads of the device. Parallel port registers for input (PGPDI) and output (PGPDO) are provided to allow a complete port to be written or read in one operation, dependent on the individual pad configuration. The difference between PGPDO and GPDO is that PGPDO registers can be used to set the values of all output pins assigned to a device port with a single 16-bit register write vs. the GPDO registers, which are used to set the value on a specific pin with a byte write.

Offset: 0x1700 + n*0x2 (n=0 to 8) Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	PPDO															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 113. SIUL2 Parallel GPIO Pad Data Out Register (SIUL2_PGPDO*n*)

Table 209. SIUL2_PGPDO*n* field descriptions

Field	Description
0:15 PPDO	Parallel Pad Data Out Write or read the data register that stores the value to be driven on the pad in output mode. Access to this register location is coherent with access to the bit-wise GPIO Pad Data Output Registers (SIUL2_GPDO). The x and y bit indexes define which PPDO register bit is equivalent to which PDO register bit according to the following equation: SIUL2_PGPDO[x].PPDO[y] = PDO[(x * 16) + y]

Note: The SIUL2_PGPDO registers access the same physical resource as the SIUL2_PDO and SIUL2_MPGPDO address locations. Some examples of the mapping:

SIUL2_PGPDO0.PPDO[0] = PDO[0]

SIUL2_PGPDO2.PPDO[0] = PDO[32]

SIUL2_PGPDO8.PPDO[3] = PDO[131]

14.2.2.19 SIUL2 Parallel GPIO Pad Data In Register (SIUL2_PGPDI*n*)

These registers hold the synchronized input value from the pads. Parallel port registers for input (PGPDI) and output (PGPDO) are provided to allow a complete port to be written or read in one operation, dependent on the individual pad configuration. The difference between PGPDI and GPDI registers is that PGPDI registers can be used to read the values of all input pins assigned to a device port with a single 16-bit register read vs. the GPDI registers, which are used to read the value on a specific pin with a byte read.

Address: 0x1740 + n*0x2 (n=0 to 8) Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	PPDI															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 114. SIUL2 Parallel GPIO Pad Data In Register (SIUL2_PGPDI*n*)

Note: PGPDI registers reset values depend on the values on the PADs. After reset, registers will reflect the values on PADs.

Table 210. SIUL2_PGPDI*n* field descriptions

Field	Description
0:15 PPDI	Parallel Pad Data In Reads the current pad value. Access to this register location is coherent with access to the bit-wise GPIO Pad Data Input Registers (SIUL2_GPDI). The x and y bit indexes define which PPDI register bit is equivalent to which PDI register bit according to the following equation: SIUL2_PGPDO[x].PPDI[y] = PDI[(x * 16) + y]

14.2.2.20 SIUL2 Masked Parallel GPIO Pad Data Out Register (SIUL2_MPGPDO*n*)

This register can be used to selectively modify the pad values associated to PPDO[x][0:15]. The MPGPDO[x] register may only be accessed with 32-bit writes. 8-bit or 16-bit writes will not modify any bits in the register and cause a transfer error response by the module. Read access will return 0.

Offset: 0x1780 +n*0x4 (n=0 to 8) Access: User read/write

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

Figure 115. SIUL2 Masked Parallel GPIO Pad Data Out Register (SIUL2_MPGPDO*n*)

Table 211. SIUL2_MPGPDO*n* field descriptions

Field	Description
0:15 MASK[0:15]	Mask Field Each bit corresponds to one data bit in the MPPDO at the same bit location. MASK[x] = 0 The associated bit value in the MPPDO field is ignored MASK[x] = 1 The associated bit value in the MPPDO field is written
16:31 MPPDO[0:15]	Masked Parallel Pad Data Out Write the data register that stores the value to be driven on the pad in output mode. Access to this register location is coherent with access to the bit-wise GPIO Pad Data Output Registers (PDO). The x and y bit indexes define which MPPDO register bit is equivalent to which PDO register bit according to the following equation: SIUL2_MPGPDOx.MPPDO[y] = PDO[(x * 16) + y]

14.3 Functional description

14.3.1 General

This section provides a complete functional description of the SIUL2.

14.3.2 Pad control

The SIUL2 is capable of controlling the electrical characteristic of 80 pads on this device. It provides a consistent interface for all pads, both on a by-port and a by-bit basis. The following describes the pad characteristics that can be supported by the SIUL2. Not all of these features will be supported on all devices or on all pads as this is dependent on the specific application needs and the pads implemented on the device.

The required controls are:

- Drive strength control
  - This is needed to offer improved EMC performance.
- Internal weak pull capability
  - This is needed to offer flexibility in the device configuration and the elimination of external hardware in some cases.
  - The configuration of the pull on any pad should be independently controlled to be either pull-up, pull-down or no-pull enabled.
- Control of analog path switches
- Safe Mode behavior configuration
- Open drain/source enable
  - Needed to support different pads muxed (for example mux IIC, with non open drain pads)
- Pin Function Assignment
  - This setting defines which chip function has control over the output of the pad.

The setting of each pad out of reset is fixed per MCU, but can be configured individually. In this way it is possible to select special pull settings or peripheral pad ownership per design.

It is possible for the user software to configure each pad independently of all other pads on the device or other pads grouped within a single port. This allows different pad types to be grouped together in ports and allows the necessary flexibility needed for the pads individual operation. This is achieved by grouping all of the above functions into a single register for each pad on the device, and allows each pad to be configured with a single write to one register and allowing simplified duplication of software for each pad with indexed changes for each pad.

14.3.3 General purpose input or output pads (GPIO)

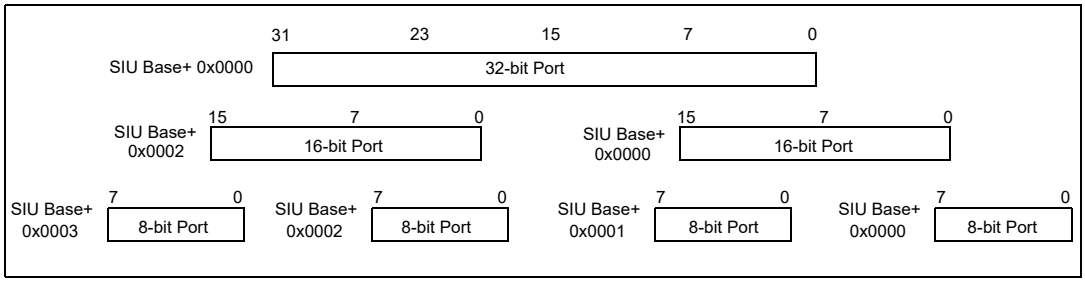
The SIUL2 allows each pad to be configured as either a General Purpose Input Output pad (GPIO), and as one or more alternate functions (input or output), the function of which is determined by the peripheral that will use the pad.

GPIO pads can also optionally be implemented without any alternate function.

The SIUL2 is designed to manage 203 GPIO pads organized as ports that can be accessed for data reads and writes as 32-bit, 16-bit or 8-bit.

As shown in *Figure 116*, all port accesses are identical with each read or write being performed only at a different location to access a different port width.

Figure 116. Data Port example arrangement showing configuration for different port width accesses



This implementation requires that the registers are arranged in such a way as to support this range of port widths without having to split reads or writes into multiple accesses.

The SIUL2 has separate data input and data output registers for all pads, allowing the possibility of reading back an input or output value of a pad directly. This supports the ability to validate what is present on the pad rather than simply confirming the value that was written to the data register by accessing the data input registers.

The data output registers support both read and write operations to be performed.

The data input registers support read access only.

When the pad is configured to use one of its alternate functions, the data input value reflect the respective value of the pad. If a write operation is performed to the data output register for a pad configured as an alternate function (non-GPIO), this write will not be reflected by the pad value until reconfigured to GPIO.

All general purpose pads are implemented as bidirectional.

Note: In case the bidirectional operation impacts some performances (for example ADC accuracy), or is not required for a specific pad function it is acceptable to limit the functionality of some pads to "Input Only".

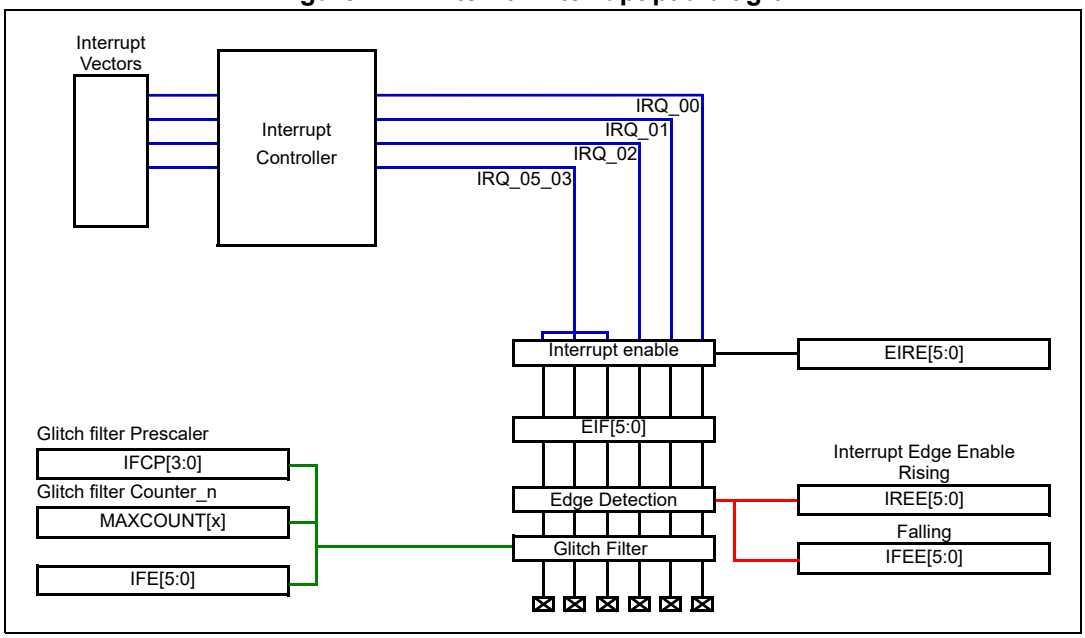
14.3.4 External interrupts/DMA requests (EIRQ pins)

The SIUL2 supports 6 external interrupts which can be allocated to any pad necessary at the MCU level. This allocation is fixed per MCU.

The SIUL2 supports 4 interrupt vectors to the interrupt controller of the MCU. Each interrupt vector can support six external interrupt sources from the device pads.

Refer to *Figure 117* for an overview of the external interrupt implementation.

Figure 117. External interrupt pad diagram



All the external interrupt pads within a single group (maximum 8 pads) have equal priority. It is the responsibility of the user software to search through the group of sources in the most appropriate way for their application.

The priority of the vectors used by the external interrupt pads is fixed based on the platform and the interrupt controller and its priority levels, but the allocation of pads to each group of interrupts can be independently configured by the MCU.

An MCU-specific number of external interrupt lines can have digital glitch filters applied to them. The supported range is 1 to 8. The glitch filters need a running internal oscillator clock to work. If no such clock is available, external interrupts will be effectively disabled when the glitch filter is enabled on an interrupt line.

14.3.4.1 External interrupt initialization

When an external interrupt pin is first enabled, it is possible to get a false interrupt flag. To prevent a false interrupt request, during pin interrupt initialization, the user must do the following:

- Mask interrupts by clearing the EIREn bits in DIRER0.
- Select the pin polarity by setting the appropriate IREEn bits in IREER0 and the appropriate IFEEn bits in IFEER0 as desired.
- Configure the appropriate bits in the MSCR[0–511] register for the external interrupt pin(s) desired as follows:
  - Clear the ODC bits to do disable output.
  - Set the IBE bit to enable the pin's input buffer.
  - If using the internal weak pullup/pulldown, configure the appropriate WPUE and WPDE bits.

Note: MSCR_SSS bits do not need to be changed for either MSCR[0:511] or MSCR[512:1023] since the external interrupt pin input is directly connected to the SIUL for EIRQ pins.

External interrupt pins should never be configured as outputs (so MSCR_ODC has to be configured as zero) when external interrupt inputs are desired since false interrupts could be detected (such as from a GPIO configuration).

- Select the request desired between DMA or Interrupt by writing the appropriate DIRSn bits in DIRSR0.
- Select the desired glitch filter setup for the pins by writing the following:
  - Write the Filter Counter setting to the desired value by writing the MAXCNT[3:0] bits in the IFMCn register for the respective external interrupt that is being used.
  - Set the Filter Clock Prescaler setting from 0 to 15 to the desired value by writing the IFCP[3:0] bits in the IFCPR.
  - Enable the glitch filter for the desired external interrupt pins by setting the appropriate IFEn bits in IFER0.
- Write to EIFn bits in DISR0 to as desired to clear any flags.
- Enable the interrupt pins by setting the appropriate EIREn bits in DIREER0.

14.3.4.2 External interrupt management

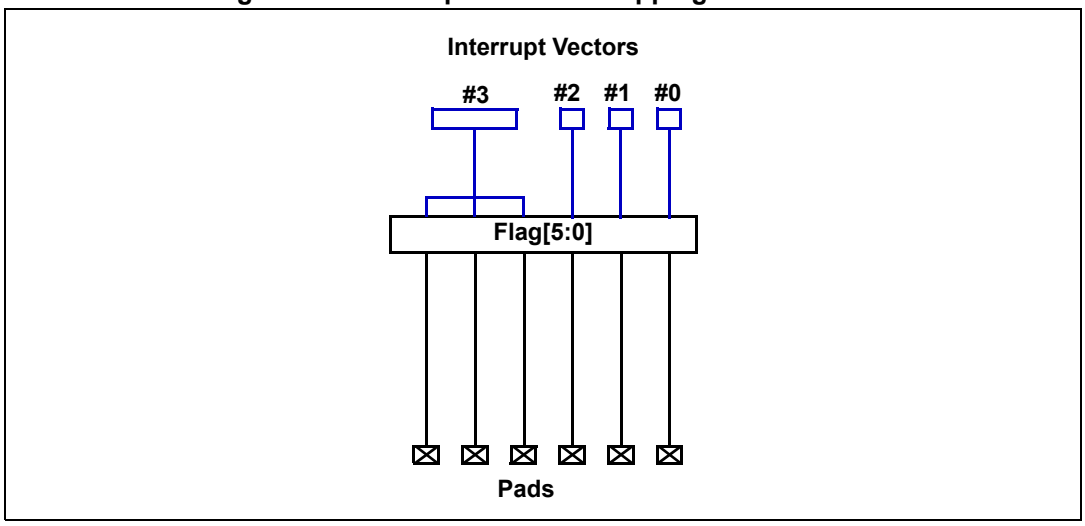
Each interrupt can be enabled or disabled independently. This can be performed using a single rolled up register (SIUL2_DIRER0 – *Figure 101*). A pad defined as an external interrupt can be configured by the user to recognize interrupts with an active rising edge, an active falling edge or both edges being active. A setting of having both edge events disabled is reserved and should not be configured.

The active IRQ edge is controlled by the users through the configuration of the registers SIUL2_IREER and SIUL2_IFEER.

Each external interrupt supports an individual flag which is held in the Flag register (DISR0). This register is a clear-by-write-1 register type, preventing inadvertent overwriting of other flags in the same register.

The SPC574Sx device supports 4 SIUL2 interrupt vectors aggregating a total of 6 external interrupt sources as shown in *Figure 118*.

Figure 118. Interrupt to vector mapping at MCU level



14.3.4.3 External interrupt request

The EIRQ input pins on the device are sources for interrupt or DMA requests. The 6 EIRQ pins for the device map to 4 interrupt requests. The mapping of interrupt requests and EIRQ pins is given in *Table 212*.

Table 212. Interrupt source mapping to SIUL2 interrupt request output

Interrupt requests	EIRQ pins
0	EIRQ0
1	EIRQ1
2	EIRQ2
3	EIRQ3 | EIRQ4 | EIRQ5

14.3.4.4 Interrupt Vector

The 6 EIRQ pins on the device map to an Interrupt Vector in the Interrupt Controller.

Note: See Section 7.3.5: Interrupt controller (INTC) configuration in the Chapter 7: Device configuration for the Interrupt Vector table.

14.3.4.5 DMA requests

The 6 EIRQ pins on the device map to an independent DMA request channel in the DMA controller.

Note: See the DMA controller configuration section in the Chapter Device Configuration for DMA request mapping table.

15 Crossbar Integrity Checker (XBIC)

15.1 Overview

The Crossbar Integrity Checker (XBIC) verifies the integrity of the crossbar transfers. For forward signals (master to slave), it is done by verifying the integrity of the attribute information using an 8-bit Error Detection Code (EDC). The EDC detects any single- or double-bit errors in the attribute information and signals the Fault Collection and Control Unit (FCCU) when an error is detected. For feedback signals (slave to master), it is done by comparing the consistence of the signals during the AHB dataphase.There are three signals from slave to master, hready, hresp0, and hresp2. If any of the master signals is different from the slave signals during dataphase, the error will be reported in the Error Status Register.

During the address phase of the system bus pipelined protocol, the XBIC generates an 8-bit EDC checkbit value for every transfer, passing these checkbits through the crossbar to the targeted slave where it is compared against the 8-bit EDC checkbits calculated from the crossbar slave attributes. The EDC is calculated for 64 bits of attribute information: 32 bits of system bus attribute control plus 32 bits associated with the processor core's decorated storage attribute. The resulting (72,64) code is a distance 4 encoding using a minimum odd weight H matrix definition. The EDC definition has been optimized for timing considerations with the system bus attribute signals exclusively using weight 3 code values and the decoration attribute using a mix of weight 3 and 5 codes.

During the data phase of the system bus pipelined protocol, the XBIC compares the three signals of hready, hresp0, hresp2 from entering the crossbar on the slave side to leaving the crossbar on the master side. Data phase is often more than one clock cycle due to wait cycles. All three signals are compared clock by clock during the whole data cycle.

15.2 Features

The XBIC has the following features:

- Verification of attribute information for all crossbar transfers
  - EDC (72,64) code protects against single and double bit errors
- Verification of feedback information for each data phase during crossbar transfer
  - hready, hresp0, and hresp2 are compared from entering the crossbar on the slave side to leaving the crossbar on the master side
- Error injection for testing
  - Programmable master and slave port specifiers
  - Programmable 8-bit toggle vector to insert error in master EDC checkbit value
  - Address, EDC syndrome, master and slave port information captured on error
- Programmable integrity check enable on a per-slave-port basis
- Programmable integrity feedback check enable on a per-master-port basis

15.3 Block diagram

The crossbar transfer attribute information for all master and slave ports is routed to the XBIC, which calculates and checks the EDC parity over the attribute information as shown in *Figure 119* and *Figure 120*.

Figure 119. XBIC system block diagram

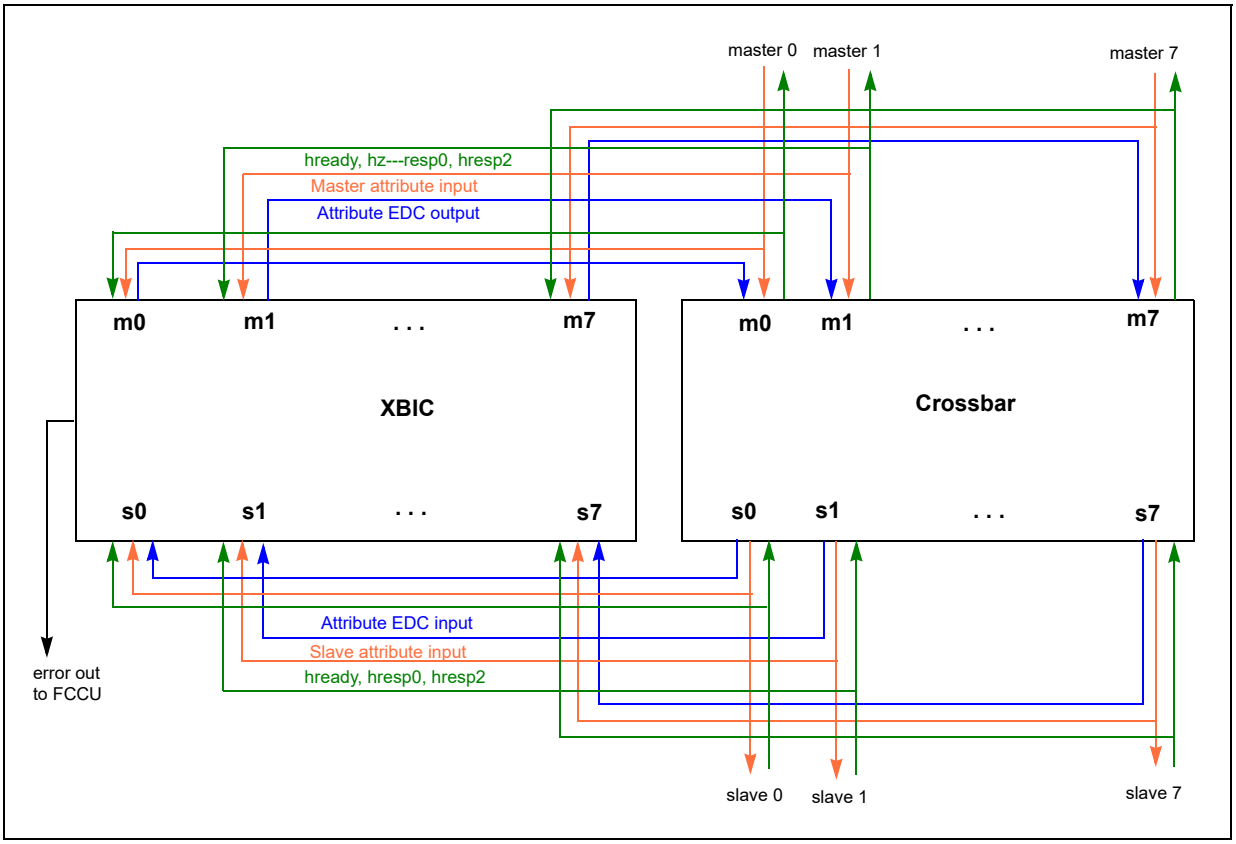
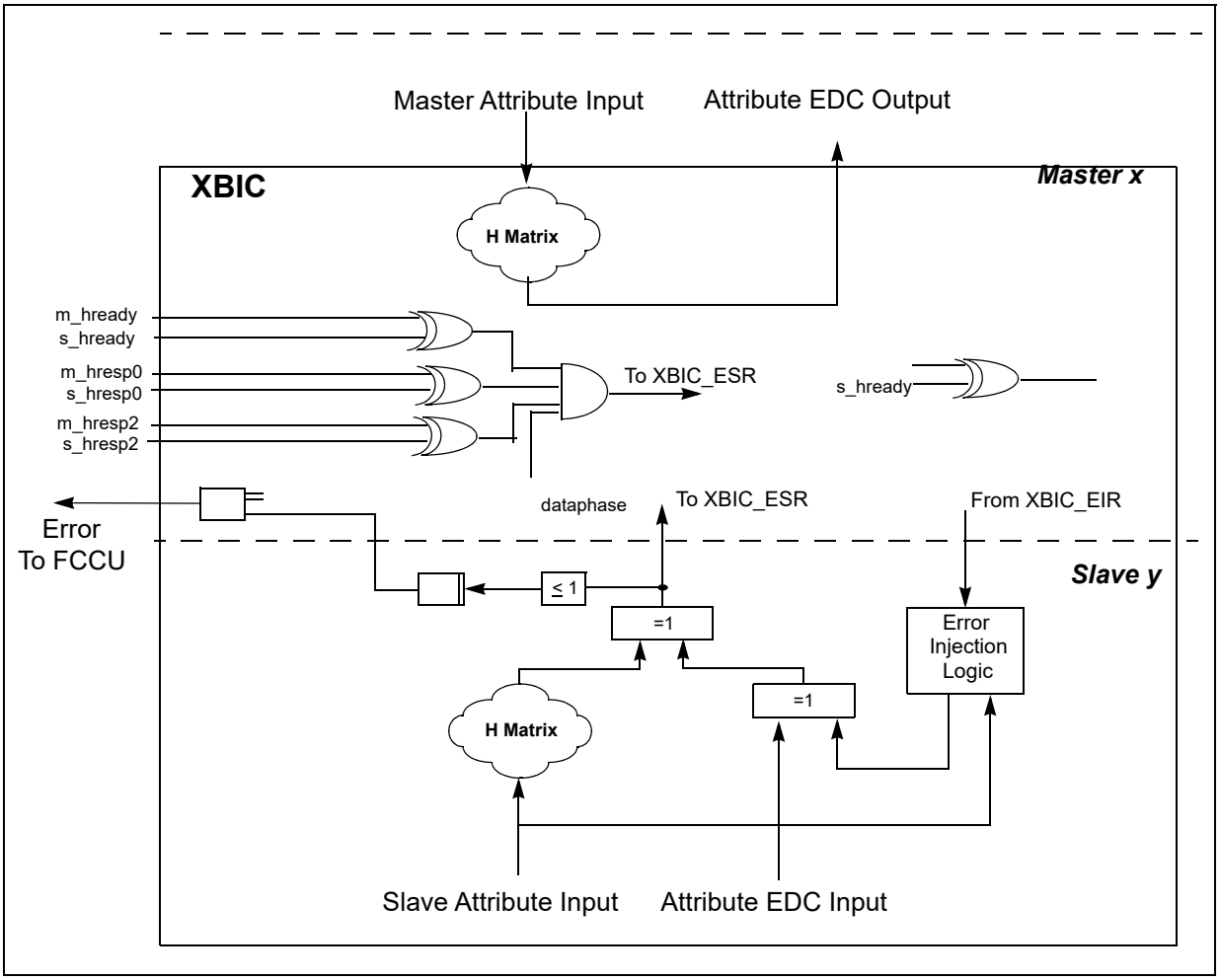


Figure 120. XBIC block diagram



15.4 External signal description

The XBIC has no external interface signals.

15.5 Memory map and register definition

The XBIC programming model has four 32-bit registers. The programming model can only be accessed in supervisor mode using 32-bit (word) accesses. Attempted references with a different access size, to an undefined (reserved) address, with a non-supported access type (a write to a read-only register), or in user mode generate an error termination.

15.5.1 Memory map

The XBIC memory map is shown in *Table 213*.

Table 213. XBIC memory map

Address offset (hex)	Register description	Location
0x0000	XBIC Module Control Register (XBIC_MCR)	*Section 15.5.2.1*
0x0004	XBIC Error Injection Register (XBIC_EIR)	*Section 15.5.2.2*
0x0008	XBIC Error Status Register (XBIC_ESR)	*Section 15.5.2.3*
0x000C	XBIC Error Address Register (XBIC_EAR)	*Section 15.5.2.4*
0x0010–0x3FFF	Reserved	

15.5.2 Register descriptions

The following sections detail the individual registers within the XBIC programming model.

15.5.2.1 XBIC Module Control Register (XBIC_MCR)

The XBIC_MCR allows attribute integrity checking to be enabled on a per-port basis.

Offset: 0x0000 Access: Supervisor Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	SE0	SE1	SE2	SE3	SE4	SE5(1)	SE6(1)	SE7(1)	ME0	ME1	ME2	ME3	ME4(1)	ME5(1)	ME6(1)	ME7(1)
W																
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

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

1. This bit is writable but has no impact.

Figure 121. XBIC Module Control Register (XBIC_MCR)

Table 214. XBIC_MCR field descriptions

Field	Description
0 SE0	Slave Port Enable for EDC error detection 0 Attribute integrity checking disabled for slave port 0 1 Attribute integrity checking enabled for slave port 0
1 SE1	Slave Port Enable for EDC error detection 0 Attribute integrity checking disabled for slave port 1 1 Attribute integrity checking enabled for slave port 1
2 SE2	Slave Port Enable for EDC error detection 0 Attribute integrity checking disabled for slave port 2 1 Attribute integrity checking enabled for slave port 2
3 SE3	Slave Port Enable for EDC error detection 0 Attribute integrity checking disabled for slave port 3 1 Attribute integrity checking enabled for slave port 3

Table 214. XBIC_MCR field descriptions (continued)

Field	Description
4 SE4	Slave Port Enable for EDC error detection 0 Attribute integrity checking disabled for slave port 4 1 Attribute integrity checking enabled for slave port 4
5 SE5	Slave Port Enable for EDC error detection 0 Attribute integrity checking disabled for slave port 5 1 Attribute integrity checking enabled for slave port 5
6 SE6	Slave Port Enable for EDC error detection 0 Attribute integrity checking disabled for slave port 6 1 Attribute integrity checking enabled for slave port 6
7 SE7	Slave Port Enable for EDC error detection 0 Attribute integrity checking disabled for slave port 7 1 Attribute integrity checking enabled for slave port 7
8 ME0	Master Port Enable for slave driven signal safety check 0 Attribute integrity checking disabled for master port 0 1 Attribute integrity checking enabled for master port 0
9 ME1	Master Port Enable for slave driven signal safety check 0 Attribute integrity checking disabled for master port 1 1 Attribute integrity checking enabled for master port 1
10 ME2	Master Port Enable for slave driven signal safety check 0 Attribute integrity checking disabled for master port 2 1 Attribute integrity checking enabled for master port 2
11 ME3	Master Port Enable for slave driven signal safety check 0 Attribute integrity checking disabled for master port 3 1 Attribute integrity checking enabled for master port 3
12 ME4	Master Port Enable for slave driven signal safety check 0 Attribute integrity checking disabled for master port 4 1 Attribute integrity checking enabled for master port 4
13 ME5	Master Port Enable for slave driven signal safety check 0 Attribute integrity checking disabled for master port 5 1 Attribute integrity checking enabled for master port 5
14 ME6	Master Port Enable for slave driven signal safety check 0 Attribute integrity checking disabled for master port 6 1 Attribute integrity checking enabled for master port 6
15 ME7	Master Port Enable for slave driven signal safety check 0 Attribute integrity checking disabled for master port 7 1 Attribute integrity checking enabled for master port 7

15.5.2.2 XBIC Error Injection Register (XBIC_EIR)

The XBIC_EIR contains fields for controlling the XBIC error injection function. When an error is injected, the EDC syndrome associated with the programmed master and slave ports is modified, causing the XBIC to assert the error indication to the FCCU and capturing the transfer information in the XBIC_ESR and XBIC_EAR registers. Otherwise, transfers are not affected by XBIC error injection.

Offset: 0x004 Access: Supervisor Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	EIE	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	SLV			MST				SYN							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 122. XBIC Error Injection Register (XBIC_EIR)

Table 215. XBIC_EIR field descriptions

Field	Description
0 EIE	Error Injection Enable 0 Error injection disabled 1 Error injection enabled
17:19 SLV	Target Slave Port (1) Error injection is enabled for the designated slave port. Other slave ports are unaffected.
20:23 MST	Target Master ID (2) Error injection is enabled for the designated master ID. Transfers with other master IDs are not affected.
24:31 SYN	Syndrome This value is exclusive-ored with the calculated syndrome (which should be zero) to generate an error with the specified syndrome. A value of zero does not generate an error.

1. For the list of the target slave ports refer to the table ‘Crossbar switch slave assignments’ in section ‘Crossbar switch’ from ‘Device Configuration’ chapter.
2. For the list of the target master IDs refer to the table ‘SMPU logical bus master assignments’ in section ‘Crossbar switch’ from ‘Device Configuration’ chapter.

15.5.2.3 XBIC Error Status Register (XBIC_ESR)

The Error Status Register (XBIC_ESR) contains two types of error information about the *last* transfer. If an attribute integrity check error was detected, the slave port, the master port and the syndrome are captured in the SLV, MST and SYN fields. If there is a mismatch among internal signals (hready, hresp0, and hresp2) during the data phase, the slave and master port are captured in the DPSE0*x* and DPME*x* fields.

This register is cleared only on reset.

Offset: 0x008 Access: Supervisor Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	VLD	DPSE0	DPSE1	DPSE2	DPSE3	DPSE4	0	0	0	DPME0	DPME1	DPME2	DPME3	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	SLV			MST				SYN							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 123. XBIC Error Status Register (XBIC_ESR)

Table 216. XBIC_ESR field descriptions

Field	Description
0 VLD	Error Status Valid 0 No error detected, other fields of the ESR and EAR are not valid. 1 Error detected, all fields of the ESR and EAR are valid.
1 DPSE0	Data phase slave port error 0 No error on slave port 0 1 Data phase error on slave port 0
2 DPSE1	Data phase slave port error 0 No error on slave port 1 1 Data phase error on slave port 1
3 DPSE2	Data phase slave port error 0 No error on slave port 2 1 Data phase error on slave port 2
4 DPSE3	Data phase slave port error 0 No error on slave port 3 1 Data phase error on slave port 3
5 DPSE4	Data phase slave port error 0 No error on slave port 4 1 Data phase error on slave port 4
9 DPME0	Data phase master port error 0 No error on master port 0 1 Data phase error on master port 0
10 DPME1	Data phase master port error 0 No error on master port 1 1 Data phase error on master port 1
11 DPME2	Data phase master port error 0 No error on master port 2 1 Data phase error on master port 2

Table 216. XBIC_ESR field descriptions (continued)

Field	Description
12 DPME3	Data phase master port error 0 No error on master port 3 1 Data phase error on master port 3
17:19 SLV	Slave Port The slave port targeted by the last transfer with an error.
20:23 MST	Master ID The master ID value of the last transfer with an error.
24:31 SYN	Syndrome The syndrome calculated for the last transfer with an error. For single bit errors the signal in error can be determined, see *Table 218*.

15.5.2.4 XBIC Error Address Register (XBIC_EAR)

The Error Address Register (XBIC_EAR) contains the address of the last transfer in which an attribute integrity check error was detected on an enabled slave port. This register is only cleared on reset.

Offset: 0x00C Access: Supervisor Read

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

Figure 124. Error Address Register (XBIC_EAR)

Table 217. XBIC_EAR Register field descriptions

Field	Description
0:31 ADDR	The address of the last transfer with an error.

15.6 Functional description

The Crossbar Integrity Checker (XBIC) verifies the integrity of the attribute information for crossbar transfers using an 8-bit Error Detection Code (EDC). The EDC detects any single or double bit errors in the attribute information and signals the Fault Collection and Control Unit (FCCU) when an error is detected. The Module Control Register (XBIC_MCR) allows integrity checking to be enabled on a per-slave-port basis. The EDC parity is generated by the XBIC for each master port during the system bus address phase pipeline stage, routed through the crossbar via a sideband signal and checked against the slave attribute EDC generated for each enabled slave port during the first system bus data phase cycle. Detection of an error does *not* generate a bus error. The XBIC attribute integrity checking is independent from the end-to-end ECC that covers the transfer address and data.

The XBIC can be programmed to intentionally inject EDC errors for testing. Error injection is targeted toward a single slave port and a single master ID at a time as determined by the

settings in the Error Injection Register (XBIC_EIR). When an error is injected, the EDC syndrome is modified which causes the XBIC to assert the error indication to the FCCU. Otherwise transfers are unaffected by XBIC error injection. Error injection using XBIC_EIR register does not update the DPxxx bits in XBIC_ESR register, it only updates the syndrome information in XBIC_ESR register. This approach allows the check logic to be verified without compromising the integrity of the data transfer. Once the error injection function is enabled by setting XBIC_EIR[EIE], errors are induced on all subsequent targeted transactions until XBIC_EIR[EIE] is cleared.

Forwarding errors (master to slave) are detected during the AHB address phase. When a forwarding error is detected, due either to a hardware fault or error injection, information related to the error is captured and reported in the Error Status Register (XBIC_ESR) and Error Address Register (XBIC_EAR). The XBIC_ESR and XBIC_EAR registers contain information about the *last* transfer with a detected error and are only cleared on reset. The cause of a single-bit error can be determined by the syndrome value reported in the XBIC_ESR as shown in *Table 218*. Any other syndrome indicates a multi-bit error.

XBIC also detects backward signal (slave to master) errors during AHB data phase. For the three slave driven signals (hready, hresp0, and hresp2), XBIC performs the comparison between the input of the slave port and output of the master port during the AHB data phase. When a mismatch occurs, the error will be captured. The current slave port will be registered in XBIC_ESR bit field DPSE*x*, and the current master port will be registered in XBIC_ESR bit field DSME*x*.

Table 218. Hexadecimal attribute single-bit error syndromes

Signal	SYN	Signal	SYN	Signal	SYN	Signal	SYN
hwrite	07	hbstrb[7]	70	hdecor[31]	25	hdecor[15]	23
htrans[0]	0b	hbstrb[6]	32	hdecor[30]	68	hdecor[14]	51
hsize[2]	0d	hbstrb[5]	52	hdecor[29]	c7	hdecor[13]	54
hsize[1]	0e	hbstrb[4]	a8	hdecor[28]	83	hdecor[12]	61
hsize[0]	13	hbstrb[3]	43	hdecor[27]	85	hdecor[11]	e3
hprot[5]	15	hbstrb[2]	45	hdecor[26]	86	hdecor[10]	e6
hprot[4]	16	hbstrb[1]	4c	hdecor[25]	89	hdecor[9]	f8
hprot[3]	19	hbstrb[0]	a4	hdecor[24]	8a	hdecor[8]	38
hprot[2]	1a	hmaster[3]	a2	hdecor[23]	8c	hdecor[7]	58
hprot[1]	1c	hmaster[2]	b0	hdecor[22]	49	hdecor[6]	37
hprot[0]	91	hmaster[1]	c1	hdecor[21]	92	hdecor[5]	f1
hburst[2]	a1	hmaster[0]	c2	hdecor[20]	94	hdecor[4]	3b
hburst[1]	64	hslave[2]	c4	hdecor[19]	98	hdecor[3]	3d
hburst[0]	29	hslave[1]	c8	hdecor[18]	46	hdecor[2]	3e
hmastlock	2a	hslave[0]	d0	hdecor[17]	34	hdecor[1]	4f
hunalign	2c	hdecorated	e0	hdecor[16]	4a	hdecor[0]	6e
edc[7]	80	edc[6]	40	edc[5]	20	edc[4]	10
edc[3]	08	edc[2]	04	edc[1]	02	edc[0]	01

The H matrix for the attribute EDC is shown in *Table 219*.

Table 219. XBIC H Matrix Definition

Checkbits [7:0]	Attribute Bit																															
	hwrite	htrans[0]	hsize[2]	hsize[1]	hsize[0]	hprot[5]	hprot[4]	hprot[3]	hprot[2]	hprot[1]	hprot[0]	hburst[2]	hburst[1]	hburst[0]	hmastlock	hunalign	hbstrb[7]	hbstrb[6]	hbstrb[5]	hbstrb[4]	hbsrtb[3]	hbsrtb[2]	hbsrtb[1]	hbsrtb[0]	hmaster[3]	hmaster[2]	hmaster[1]	hmaster[0]	hslave[2]	hslave[1]	hslave[0]	hdecorated
7											*	*								*				*	*	*	*	*	*	*	*	*
6													*				*		*		*	*	*				*	*	*	*	*	*
5												*	*	*	*	*	*	*		*				*	*	*						*
4					*	*	*	*	*	*	*						*	*	*							*					*	
3		*	*	*				*	*	*				*	*	*				*			*							*		
2	*		*	*		*	*			*			*			*						*	*	*					*			
1	*	*		*	*		*		*						*			*	*		*				*			*				
0	*	*	*		*	*		*			*	*		*							*	*					*					
Checkbits [7:0]	hdecor[31]	hdecor[30]	hdecor[29]	hdecor[28]	hdecor[27]	hdecor[26]	hdecor[25]	hdecor[24]	hdecor[23]	hdecor[22]	hdecor[21]	hdecor[20]	hdecor[19]	hdecor[18]	hdecor[17]	hdecor[16]	hdecor[15]	hdecor[14]	hdecor[13]	hdecor[12]	hdecor[11]	hdecor[10]	hdecor[9]	hdecor[8]	hdecor[7]	hdecor[6]	hdecor[5]	hdecor[4]	hdecor[3]	hdecor[2]	hdecor[1]	hdecor[0]	
7			*	*	*	*	*	*	*		*	*	*								*	*	*				*					
6		*	*							*				*		*		*	*	*	*	*	*		*		*				*	*	
5	*	*													*		*			*	*	*	*	*		*	*	*	*	*		*	
4											*	*	*		*			*	*				*	*	*	*	*	*	*	*			
3		*					*	*	*	*			*			*							*	*	*			*	*	*	*	*	
2	*		*		*	*			*			*		*	*				*			*				*			*	*	*	*	
1			*	*		*		*			*			*		*	*				*	*				*		*		*	*	*	
0	*		*	*	*		*			*							*	*		*	*					*	*	*	*		*		

16 Peripheral bridge

16.1 Introduction

The peripheral bridge converts the crossbar switch interface to an interface that can access a majority of peripherals on the device.

The peripheral bridge occupies a 64 MB portion of the address space. (Not all peripheral slots may be used. See *Chapter 7: Device configuration* or *Chapter 5: Memory map* for details on slot assignment.) The bridge includes separate clock enable inputs for each of the slots, to accommodate slower peripherals.

16.1.1 Features

Key features of the peripheral bridge are:

- Supports a pair of 32-bit transactions for selected 64-bit memory accesses
- Each independently configurable peripheral includes a clock enable, which allows peripherals to operate at any speed less than the system clock rate
- Programming model that provides memory protection functionality

16.1.2 General operation

The slave devices connected to the peripheral bridge are modules that contain readable/writable control and status registers. The system masters read and write these registers through the peripheral bridge. The peripheral bridge generates module enables, the module address, transfer attributes, byte enables, and write data as inputs to the peripherals. The peripheral bridge captures read data from the peripheral interface and drives it to the crossbar switch.

Each peripheral is typically allocated one 16-KB block (or slot) of the memory map. Usually, the peripheral bridge uses the size of the addressed peripheral to perform proper data byte lane routing only; no bus decomposition (dynamic bus sizing) is performed. However, there are the following exceptions:

- Aligned 64-bit reads are permitted to 32-bit slots.
- Aligned 64-bit writes may be targeted to 32-bit slots
- Aligned or misaligned 64-bit instruction fetches may also be performed to 32-bit slots

All other 64-bit accesses result in an error response.

16.2 Memory map/register definition

Table 220. Peripheral bridge memory map

Offset address	Register	Location
0x000	Master Privilege Register A (PBRIDGE_MPRA)	*Section 16.2.1*
0x004–0x1F	Reserved(1)	
0x020–0x2F	Reserved(2)	

Table 220. Peripheral bridge memory map (continued)

Offset address	Register	Location
0x030–0x3F	Reserved(1)	
0x040–0x83	Reserved(2)	
0x084–0xFF	Reserved(1)	
0x100	Peripheral Access Control Register A (PBRIDGE_PACRA)	*Section 16.2.2*
0x104	Peripheral Access Control Register B (PBRIDGE_PACRB)	*Section 16.2.2*
0x108	Peripheral Access Control Register C (PBRIDGE_PACRC)	*Section 16.2.2*
0x10C	Peripheral Access Control Register D (PBRIDGE_PACRD)	*Section 16.2.2*
0x110–0x113	Reserved(1)	
0x114	Peripheral Access Control Register F (PBRIDGE_PACRF)	*Section 16.2.2*
0x118	Reserved(1)	
0x11C	Reserved(1)	
0x120–0x12F	Reserved(2)	
0x130–0x133	Reserved(1)	
0x134–0x13F	Reserved(2)	
0x140	Off-platform Peripheral Access Control Register A (PBRIDGE_OPACRA)	*Section 16.2.3*
0x144	Off-platform Peripheral Access Control Register B (PBRIDGE_OPACRB)	*Section 16.2.3*
0x148	Off-platform Peripheral Access Control Register C (PBRIDGE_OPACRC)	*Section 16.2.3*
0x14C	Off-platform Peripheral Access Control Register D (PBRIDGE_OPACRD)	*Section 16.2.3*
0x150	Off-platform Peripheral Access Control Register E (PBRIDGE_OPACRE)	*Section 16.2.3*
0x154	Off-platform Peripheral Access Control Register F (PBRIDGE_OPACRF)	*Section 16.2.3*
0x158	Off-platform Peripheral Access Control Register G (PBRIDGE_OPACRG)	*Section 16.2.3*
0x15C	Off-platform Peripheral Access Control Register H (PBRIDGE_OPACRH)	*Section 16.2.3*
0x160	Off-platform Peripheral Access Control Register I (PBRIDGE_OPACRI)	*Section 16.2.3*
0x164	Off-platform Peripheral Access Control Register J (PBRIDGE_OPACRJ)	*Section 16.2.3*
0x168	Off-platform Peripheral Access Control Register K (PBRIDGE_OPACRK)	*Section 16.2.3*
0x16C	Off-platform Peripheral Access Control Register L (PBRIDGE_OPACRL)	*Section 16.2.3*

Table 220. Peripheral bridge memory map (continued)

Offset address	Register	Location
0x170	Off-platform Peripheral Access Control Register M (PBRIDGE_OPACRM)	*Section 16.2.3*
0x174	Off-platform Peripheral Access Control Register N (PBRIDGE_OPACRN)	*Section 16.2.3*
0x178	Off-platform Peripheral Access Control Register O (PBRIDGE_OPACRO)	*Section 16.2.3*
0x17C	Off-platform Peripheral Access Control Register P (PBRIDGE_OPACRP)	*Section 16.2.3*
0x180	Off-platform Peripheral Access Control Register Q (PBRIDGE_OPACRQ)	*Section 16.2.3*
0x184	Off-platform Peripheral Access Control Register R (PBRIDGE_OPACRR)	*Section 16.2.3*
0x188	Off-platform Peripheral Access Control Register S (PBRIDGE_OPACRS)	*Section 16.2.3*
0x18C	Off-platform Peripheral Access Control Register T (PBRIDGE_OPACRT)	*Section 16.2.3*
0x190	Off-platform Peripheral Access Control Register U (PBRIDGE_OPACRU)	*Section 16.2.3*
0x194	Off-platform Peripheral Access Control Register V (PBRIDGE_OPACRV)	*Section 16.2.3*
0x198	Off-platform Peripheral Access Control Register W (PBRIDGE_OPACRW)	*Section 16.2.3*
0x19C	Off-platform Peripheral Access Control Register X (PBRIDGE_OPACRX)	*Section 16.2.3*
0x1A0	Off-platform Peripheral Access Control Register Y (PBRIDGE_OPACRY)	*Section 16.2.3*
0x1A4	Off-platform Peripheral Access Control Register Z (PBRIDGE_OPACRZ)	*Section 16.2.3*
0x1A8	Off-platform Peripheral Access Control Register AA (PBRIDGE_OPACRAA)	*Section 16.2.3*
0x1AC	Off-platform Peripheral Access Control Register AB (PBRIDGE_OPACRAB)	*Section 16.2.3*
0x1B0	Off-platform Peripheral Access Control Register AC (PBRIDGE_OPACRAC)	*Section 16.2.3*
0x1B4	Off-platform Peripheral Access Control Register AD (PBRIDGE_OPACRAD)	*Section 16.2.3*
0x1B8	Off-platform Peripheral Access Control Register AE (PBRIDGE_OPACRAE)	*Section 16.2.3*
0x1BC	Off-platform Peripheral Access Control Register AF (PBRIDGE_OPACRAF)	*Section 16.2.3*

1. Access to this address space will terminate in an error.

2. The following address spaces when programmed, will not terminate in an error. Software should not program these address spaces

16.2.1 Master Privilege Registers (PBRIDGE_MPRA)

Each MPR specifies eight 4-bit fields defining the access privilege level associated with a bus master in the device to the various peripherals. The register provides one field per bus master. Each master is assigned depending on its logical master number. See *Chapter 7: Device configuration* for details about the master assignments to these registers.

Offset: 0x000 (PBRIDGE_MPRA) Access: Supervisor read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	MPROT0				MPROT1				MPROT2				MPROT3			
Reset	0	1	1	1	0	1	1	1	0	1	1	1	0	1	1	1

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	MPROT4				MPROT5				MPROT6				MPROT7			
Reset	0	1	1	1	0	1	1	1	0	1	1	1	0	1	1	1

Figure 125. Master Privilege Register A (MPRA)

The MPROT*n* fields are defined as shown in *Figure 126*.

	0	1	2	3
R	0	MTR	MTW	MPL
W				

Figure 126. MPROT*n* fields

Table 221. MPROT*n* field descriptions

Field	Description
0	Reserved, should be cleared
1 MTR	Master trusted for read Determines whether the master is trusted for read accesses. 0 This master is not trusted for read accesses 1 This master is trusted for read accesses

Table 221. MPROT*n* field descriptions (continued)

Field	Description
2 MTW	Master trusted for writes Determines whether the master is trusted for write accesses 0 This master is not trusted for write accesses 1 This master is trusted for write accesses.
3 MPL	Master privilege level Determines how the privilege level of the master is determined. 0 Accesses from this master are forced to user mode 1 Accesses from this master are not forced to user mode

16.2.2 Peripheral Access Control Registers (PBRIDGE_PACR*x*)

Each of the on-platform peripherals has a four-bit PACR*n* field which defines the access levels supported by the given module. Eight PACR fields are grouped together to form a 32-bit PACR*x* register.

The peripheral assignments to each PACR field are defined by the memory map slot to which the peripherals are assigned. See *Chapter 5: Memory map* for the assignments for your particular device. If a peripheral is absent, the corresponding PACR field is not implemented. Reads to the location return zeroes, and writes are ignored.

Table 222 shows the top-level structure of the PACRs.

Table 222. PACR memory map

Offset	Register	[0:3]	[4:7]	[8:11]	[12:15]	[16:19]	[20:23]	[24:27]	[28:31]
0x0100	PACRA	PACR0	PACR1	PACR2	PACR3	PACR4	PACR5	PACR6	PACR7
0x0104	PACRB	PACR8	PACR9	PACR10	PACR11	PACR12	PACR13	PACR14	PACR15
0x0108	PACRC	PACR16	PACR17	PACR18	PACR19	PACR20	PACR21	PACR22	PACR23
0x010C	PACRD	PACR24	PACR25	PACR26	PACR27	PACR28	PACR29	PACR30	PACR31
0x0110	Reserved								
0x0114	PACRF	PACR40	PACR41	PACR42	PACR43	PACR44	PACR45	PACR46	PACR47

Offset: 0x100 (PBRIDGE_PACRA)
0x104 (PBRIDGE_PACRB)
0x108 (PBRIDGE_PACRC)
0x10C (PBRIDGE_PACRD)
0x114 (PBRIDGE_PACRF)

Access: Supervisor read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	PACR*n*				PACR(*n*+1)				PACR(*n*+2)				PACR(n+3)			
Reset	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	PACR(*n*+4)				PACR(*n*+5)				PACR(*n*+6)				PACR(*n*+7)			
Reset	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0

Figure 127. Peripheral Access Control Registers (PBRIDGE_PACR*x*)

Each PACR*n* field is defined as shown in *Figure 128*.

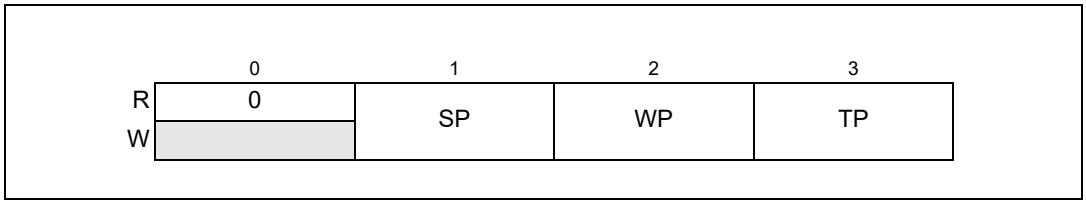


Figure 128. PACR*n* fields

Table 223. PACR*n* field descriptions

Field	Description
0	Reserved, should be cleared
1 SP	Supervisor protect Determines whether the peripheral requires supervisor privilege level for access. 0 This peripheral does not require supervisor privilege level for accesses. 1 This peripheral requires supervisor privilege level for accesses. The master privilege level must indicate supervisor access attribute, and the MPROT*n*[MPL] control bit for the master must be set. If not, the access is terminated with an error response and no peripheral access is initiated.
2 WP	Write protect Determines whether the peripheral allows write accesses. 0 This peripheral allows write accesses. 1 This peripheral is write-protected. If a write access is attempted, the access is terminated with an error response and no peripheral access is initiated.
3 TP	Trusted protect Determines whether the peripheral allows accesses from an untrusted master. 0 Accesses from an untrusted master are allowed. 1 Accesses from an untrusted master are not allowed. If an access is attempted by an untrusted master, the access is terminated with an error response and no peripheral access is initiated.

16.2.3 Off-Platform Peripheral Access Control Registers (PBRIDGE_OPACR*x*)

Each of the off-platform peripherals has a four-bit OPACR*n* field which defines the access levels supported by the given module. Eight OPACRs are grouped together to form a 32-bit OPACR*x* register.

The peripheral assignments to each OPACR field are defined by the memory map slot to which the peripherals are assigned. See *Chapter 5: Memory map* for the assignments for your particular device. If a peripheral is absent, the corresponding OPACR is not implemented. Reads to the location return zeroes and writes are ignored.

Table 224 shows the top-level structure of the OPACR registers.

Table 224. OPACR memory map

Offset	Register	[0:3]	[4:7]	[8:11]	[12:15]	[16:19]	[20:23]	[24:27]	[28:31]
0x0140	OPACRA	OPACR 0	OPACR 1	OPACR 2	OPACR 3	OPACR 4	OPACR 5	OPACR 6	OPACR 7
0x0144	OPACRB	OPACR 8	OPACR 9	OPACR 10	OPACR 11	OPACR 12	OPACR 13	OPACR 14	OPACR 15
0x0148	OPACRC	OPACR 16	OPACR 17	OPACR 18	OPACR 19	OPACR 20	OPACR 21	OPACR 22	OPACR 23
0x014C	OPACRD	OPACR 24	OPACR 25	OPACR 26	OPACR 27	OPACR 28	OPACR 29	OPACR 30	OPACR 31
0x0150	OPACRE	OPACR 32	OPACR 33	OPACR 34	OPACR 35	OPACR 36	OPACR 37	OPACR 38	OPACR 39
0x0154	OPACRF	OPACR 40	OPACR 41	OPACR 42	OPACR 43	OPACR 44	OPACR 45	OPACR 46	OPACR 47
0x0158	OPACRG	OPACR 48	OPACR 49	OPACR 50	OPACR 51	OPACR 52	OPACR 53	OPACR 54	OPACR 55
0x015C	OPACRH	OPACR 56	OPACR 57	OPACR 58	OPACR 59	OPACR 60	OPACR 61	OPACR 62	OPACR 63
0x0160	OPACRI	OPACR 64	OPACR 65	OPACR 66	OPACR 67	OPACR 68	OPACR 69	OPACR 70	OPACR 71
0x0164	OPACRJ	OPACR 72	OPACR 73	OPACR 74	OPACR 75	OPACR 76	OPACR 77	OPACR 78	OPACR 79
0x0168	OPACRK	OPACR 80	OPACR 81	OPACR 82	OPACR 83	OPACR 84	OPACR 85	OPACR 86	OPACR 87
0x016C	OPACRL	OPACR 88	OPACR 89	OPACR 90	OPACR 91	OPACR 92	OPACR 93	OPACR 94	OPACR 95
0x0170	OPACRM	OPACR 96	OPACR 97	OPACR 98	OPACR 99	OPACR 100	OPACR 101	OPACR 102	OPACR 103
0x0174	OPACRN	OPACR 104	OPACR 105	OPACR 106	OPACR 107	OPACR 108	OPACR 109	OPACR 110	OPACR 111

Table 224. OPACR memory map (continued)

Offset	Register	[0:3]	[4:7]	[8:11]	[12:15]	[16:19]	[20:23]	[24:27]	[28:31]
0x0178	OPACRO	OPACR 112	OPACR 113	OPACR 114	OPACR 115	OPACR 116	OPACR 117	OPACR 118	OPACR 119
0x017C	OPACRP	OPACR 120	OPACR 121	OPACR 122	OPACR 123	OPACR 124	OPACR 125	OPACR 126	OPACR 127
0x0180	OPACRQ	OPACR 128	OPACR 129	OPACR 130	OPACR 131	OPACR 132	OPACR 133	OPACR 134	OPACR 135
0x0184	OPACRR	OPACR 136	OPACR 137	OPACR 138	OPACR 139	OPACR 140	OPACR 141	OPACR 142	OPACR 143
0x0188	OPACRS	OPACR 144	OPACR 145	OPACR 146	OPACR 147	OPACR 148	OPACR 149	OPACR 150	OPACR 151
0x018C	OPACRT	OPACR 152	OPACR 153	OPACR 154	OPACR 155	OPACR 156	OPACR 157	OPACR 158	OPACR 159
0x0190	OPACRU	OPACR 160	OPACR 161	OPACR 162	OPACR 163	OPACR 164	OPACR 165	OPACR 166	OPACR 167
0x0194	OPACRV	OPACR 168	OPACR 169	OPACR 170	OPACR 171	OPACR 172	OPACR 173	OPACR 174	OPACR 175
0x0198	OPACRW	OPACR 176	OPACR 177	OPACR 178	OPACR 179	OPACR 180	OPACR 181	OPACR 182	OPACR 183
0x019C	OPACRX	OPACR 184	OPACR 185	OPACR 186	OPACR 187	OPACR 188	OPACR 189	OPACR 190	OPACR 191
0x01A0	OPACRY	OPACR 192	OPACR 193	OPACR 194	OPACR 195	OPACR 196	OPACR 197	OPACR 198	OPACR 199
0x01A4	OPACRZ	OPACR 200	OPACR 201	OPACR 202	OPACR 203	OPACR 204	OPACR 205	OPACR 206	OPACR 207
0x01A8	OPACRAA	OPACR 208	OPACR 209	OPACR 210	OPACR 211	OPACR 212	OPACR 213	OPACR 214	OPACR 215
0x01AC	OPACRAB	OPACR 216	OPACR 217	OPACR 218	OPACR 219	OPACR 220	OPACR 221	OPACR 222	OPACR 223
0x01B0	OPACRAC	OPACR 224	OPACR 225	OPACR 226	OPACR 227	OPACR 228	OPACR 229	OPACR 230	OPACR 231
0x01B4	OPACRAD	OPACR 232	OPACR 233	OPACR 234	OPACR 235	OPACR 236	OPACR 237	OPACR 238	OPACR 239
0x01B8	OPACRAE	OPACR 240	OPACR 241	OPACR 242	OPACR 243	OPACR 244	OPACR 245	OPACR 246	OPACR 247
0x01BC	OPACRAF	OPACR 248	OPACR 249	OPACR 250	OPACR 251	OPACR 252	OPACR 253	OPACR 254	OPACR 255

Offset:
0x140 (PBRIDGE_OPACRA) 0x180 (PBRIDGE_OPACRQ)
0x144 (PBRIDGE_OPACRB) 0x184 (PBRIDGE_OPACRR)
0x148 (PBRIDGE_OPACRC) 0x188 (PBRIDGE_OPACRS)
0x14C (PBRIDGE_OPACRD) 0x18C (PBRIDGE_OPACRT)
0x150 (PBRIDGE_OPACRE) 0x190 (PBRIDGE_OPACRU)
0x154 (PBRIDGE_OPACRF) 0x194 (PBRIDGE_OPACRV)
0x158 (PBRIDGE_OPACRG) 0x198 (PBRIDGE_OPACRW)
0x15C (PBRIDGE_OPACRH) 0x19C (PBRIDGE_OPACRX)
0x160 (PBRIDGE_OPACRI) 0x1A0 (PBRIDGE_OPACRIY)
0x164 (PBRIDGE_OPACRJ) 0x1A4 (PBRIDGE_OPACRZ)
0x168 (PBRIDGE_OPACRK) 0x1A8 (PBRIDGE_OPACRAA)
0x16C (PBRIDGE_OPACRL) 0x1AC (PBRIDGE_OPACRAB)
0x170 (PBRIDGE_OPACRM) 0x1B0 (PBRIDGE_OPACRAC)
0x174 (PBRIDGE_OPACRN) 0x1B4 (PBRIDGE_OPACRAD)
0x178 (PBRIDGE_OPACRO) 0x1B8 (PBRIDGE_OPACRAE)
0x17C (PBRIDGE_OPACRP) 0x1BC (PBRIDGE_OPACRAF)

Access: Supervisor read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	OPACR*n*				OPACR(*n*+1)				OPACR(*n*+2)				OPACR(*n*+3)			
Reset	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	OPACR(*n*+4)				OPACR(*n*+5)				OPACR(*n*+6)				OPACR(*n*+7)			
Reset	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0

Figure 129. Off-Platform Peripheral Access Control Registers (PBRIDGE_OPACR*x*)

The OPACR*n* field is defined as in *Figure 130*.

	0	1	2	3
R	0	SP	WP	TP
W				

Figure 130. OPACR*n* fields

Table 225. OPACR*n* field descriptions

Field	Description
0	Reserved, should be cleared
1 SP	Supervisor protect Determines whether the peripheral requires supervisor privilege level for access. 0 This peripheral does not require supervisor privilege level for accesses. 1 This peripheral requires supervisor privilege level for accesses. The master privilege level must indicate supervisor access attribute, and the MPROT*n*[MPL] control bit for the master must be set. If not, the access is terminated with an error response and no peripheral access is initiated.

Table 225. OPACR*n* field descriptions (continued)

Field	Description
2 WP	Write protect Determines whether the peripheral allows write accesses. 0 This peripheral allows write accesses. 1 This peripheral is write-protected. If a write access is attempted, the access is terminated with an error response and no peripheral access is initiated.
3 TP	Trusted protect Determines whether the peripheral allows accesses from an untrusted master. 0 Accesses from an untrusted master are allowed. 1 Accesses from an untrusted master are not allowed. If an access is attempted by an untrusted master, the access is terminated with an error response and no peripheral access is initiated.

16.3 Functional description

The peripheral bridge serves as an interface between the crossbar switch and the slave peripheral bus. It functions as a protocol translator. Support is provided for generating a pair of 32-bit slave accesses when performing certain 64-bit peripheral accesses.

Accesses which fall within the address space of the peripheral bridge are decoded to provide individual module selects for peripheral devices on the slave bus interface.

16.3.1 Access support

Aligned and misaligned 32-bit and 16-bit accesses, as well as byte accesses, are supported for 32-bit peripherals. Misaligned accesses are supported to allow memory to be placed on the slave peripheral bus. Peripheral registers must not be misaligned, although no explicit checking is performed by the peripheral bridge. The peripheral bridge performs two slave peripheral bus transfers for allowed 64-bit transactions, to 32-bit sized peripheral slots only. All other accesses are performed with a single transfer.

All accesses to the peripheral slots must be sized less than or equal to the designated peripheral slot size. If an access is attempted which is larger (in size) than the targeted port, an error response is generated without a peripheral access.

17 System Memory Protection Unit (SMPU)

17.1 Overview

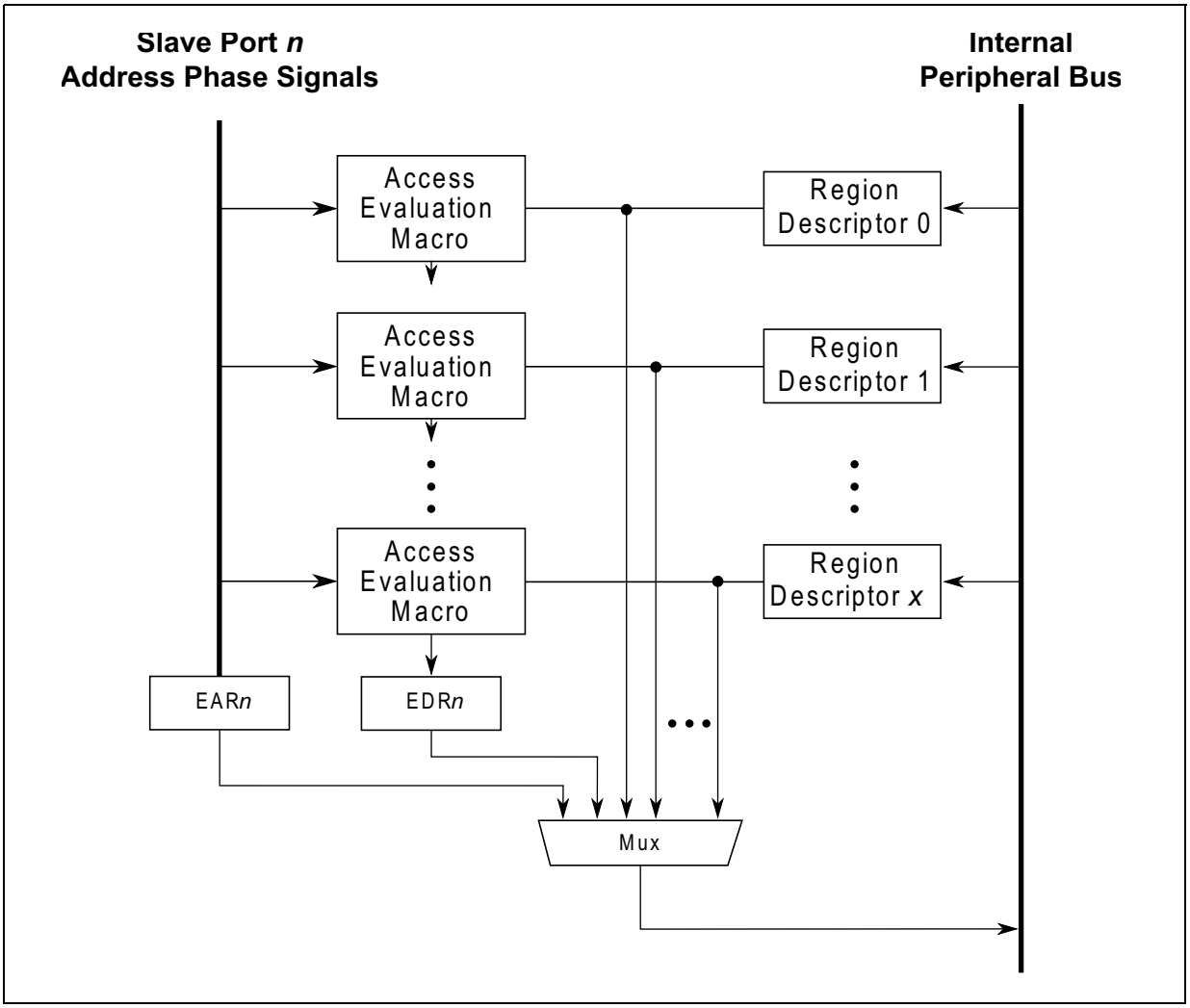
The System Memory Protection Unit (SMPU) provides hardware access control for system bus memory references. The SMPU concurrently monitors and evaluates system bus transactions using pre-programmed region descriptors that define memory spaces and their access rights. Memory references that have sufficient access control rights are allowed to complete, while references that are not mapped to any region descriptor or have insufficient rights are terminated with an access error response.

17.2 Block diagram

A simplified block diagram of the SMPU module is shown in *Figure 131*. The hardware's two-dimensional connection matrix is clearly visible with the basic access evaluation macro shown as the replicated submodule block. The crossbar switch slave ports are shown on the left, the region descriptor registers are in the middle, and the peripheral bus interface is on the right side. The evaluation macro contains two magnitude comparators connected to the start and end address registers from each region descriptor as well as the combinational logic blocks to determine the region hit and detect protection violations.

For details of the access evaluation macro, see *Section 17.5.1: Access evaluation macro*.

Figure 131. SMPU block diagram



17.3 Features

The SMPU feature set includes:

- Supports up to 8 program-visible 128-bit region descriptors, accessible as four 32-bit words each. (Specific module instances may support fewer than 24.)
  - Each region descriptor defines an arbitrarily sized space, aligned anywhere in memory
    Region sizes can vary from a minimum of 1 byte to a maximum of (4 GB minus 1 byte)
  - Read/write access control permissions defined in region descriptor
  - Cache-inhibit attribute indicator per region descriptor
  - Hardware-assisted maintenance of the descriptor valid bit minimizes coherency issues
  - Priority given to granting permission over denying access for overlapping region descriptors
- Supports as many as eight crossbar slave ports

- Supports up to 24 program-visible 128-bit region descriptors, accessible as four 32-bit words each. (Specific module instances may support fewer than 24.)
  - Each region descriptor defines an arbitrarily sized space, aligned anywhere in memory
    Region sizes can vary from a minimum of 1 byte to a maximum of (4 GB minus 1 byte)
  - Read/write access control permissions defined in region descriptor
  - Cache-inhibit attribute indicator per region descriptor
  - Hardware-assisted maintenance of the descriptor valid bit minimizes coherency issues
  - Priority given to granting permission over denying access for overlapping region descriptors
- Supports as many as eight crossbar slave ports

- Detects access errors if a memory reference does not hit in any memory region, or if the reference is illegal in all hit memory regions. If an access error occurs, the reference is terminated with an error response, and the SMPU inhibits the bus cycle being sent to the targeted slave device.
- Aborted core accesses generate instruction storage (ISI) or data storage (DSI) interrupts (see the documentation related to the core for details).

- Error registers (per bus master ID) capture the last faulting address, attributes, and other information.
- Global SMPU enable/disable control bit

17.4 Memory map and register definition

17.4.1 Memory map

The programming model is partitioned into three groups: control registers, error capture registers, and the data structure containing the region descriptors.

The programming model can only be referenced using 32-bit accesses. Attempted references using different access sizes, to undefined (reserved) addresses, or with a non-

supported access type (a write to a read-only register, or a read of a write-only register) generate an error termination.

The programming model can be accessed only in supervisor mode.

Note: See the device configuration details for any chip-specific register information for this module. For example, a specific instance of a module may support only up to 8 region descriptors.

Table 226. SMPU memory map

Address offset	Register name	Section
0x0000	CESR0	*Section 17.4.2.1*
0x0004	CESR1	*Section 17.4.2.2*
(n=0 to 15)		
0x8*n+0x0100	EARn	*Section 17.4.2.3*
0x8*n+0x0104	EDRn	*Section 17.4.2.4*
(n=0 to 2311)		
0x10*n+0x0400	RGDn_WORD0	*Section 17.4.2.5*
0x10*n+0x0404	RGDn_WORD1	*Section 17.4.2.6*
0x10*n+0x0408	RGDn_WORD2_FMT0	*Section 17.4.2.7*
0x10*n+0x040C	RGDn_WORD3	*Section 17.4.2.8*

17.4.2 Register descriptions

17.4.2.1 Control/Error Status Register 0 (CESR0)

Address: 0x0000 Access: Supervisor read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	MERR															
W	w1c															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	1	0	0	0	0	0	0	0	0	0	0	0	HRL			GVLD
W																
Reset	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0

Figure 132. Control/Error Status Register 0 (CESR0)

Table 227. CESR0 field descriptions

Field	Description
0:15 MERR	Master *n* error, where the bus master number matches the bit number Indicates a captured error in EAR*n* and EDR*n*. A bit is set when the hardware detects an error and records the faulting address and attributes. It is cleared by writing '1' to it. If another error is captured at the exact same cycle as the write, the flag remains set. A find-first-one instruction (or equivalent) can detect the presence of a captured error. 0 No error has occurred for bus master *n.* 1 An error has occurred for bus master *n.*
16	Reserved This read-only bit is reserved and always has the value '1'.
28:30 HRL	Hardware revision level Specifies the SMPU's hardware and definition revision level. It can be read by software to determine the functional definition of the module.
31 GVLD	Global Valid (global enable/disable for the SMPU) 0 SMPU is disabled. All accesses from all bus masters are allowed. 1 SMPU is enabled.

17.4.2.2 Control/Error Status Register 1 (CESR1)

Address: 0x0004 Access: Supervisor read-only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	MEOVR															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	1	0	0	0	0	0	0	0	0	0	0	0	NRGD			
W																
Reset	1	0	0	0	0	0	0	0	0	0	0	0	0	01	10	0

Figure 133. Control/Error Status Register 1 (CESR1)

Table 228. CESR1 field descriptions

Field	Description
0:15 MEOVR	Master *n* error overrun, where the bus master number matches the bit number Each bit in this field signals that another SMPU error for bus master *n* has occurred before the previous error was processed. The details of the first error are recorded in the EAR*n* and EDR*n* registers, and no information on subsequent errors is recorded until the associated CESR0.MERR flag is cleared. 0 No error overrun condition has been detected for bus master *n.* 1 An error overrun condition has been detected for bus master *n.*

Table 228. CESR1 field descriptions (continued)

Field	Description
16	Reserved This read-only bit is reserved and always has the value ‘1’.
28:31 NRGD	Number of region descriptors Indicates the number of region descriptors implemented in the SMPU. 0001 4 region descriptors 0010 8 region descriptors 0011 12 region descriptors 0100 16 region descriptors 0101 20 region descriptors 0110 24 region descriptors

17.4.2.3 Error Address Register, Bus Master *n* (EAR*n*)

When the SMPU detects an access error on bus master *n*, the 32-bit reference address is captured in this read-only register and the corresponding bit in CESR0.MERR is set. Additional information about the faulting access is captured in the corresponding EDR*n* at the same time.

Note: The corresponding EARn and EDRn registers contain information on the original access error; subsequent errors associated with the given master are recorded as overruns in the CESR1.MEOVR field until the original error is processed.

Address: 0x0100 + *n**0x8 (*n* = 0 to 15) Access: Supervisor read-only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	EADDR															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	EADDR															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 134. Error Address Register, Bus Master *n* (EAR*n*)

Table 229. EAR*n* field descriptions

Field	Description
0:31 EADDR	Error address Indicates the reference address from bus master *n* that generated the access error.

17.4.2.4 Error Detail Register, Bus Master *n* (EDR*n*)

When the SMPU detects an access error associated with bus master *n*, 32 bits of error detail are captured in this read-only register and the corresponding bit in CESR0.MERR is

set. Information on the faulting address is captured in the corresponding EAR*n* register at the same time.

Note: The corresponding EARn and EDRn registers contain information on the original access error; subsequent errors associated with the given master are recorded as overruns in the CESR1.MEOVR field until the original error is processed.

Address: 0x0104 + *n**0x8 (*n* = 0 to 15) Access: Supervisor read-only

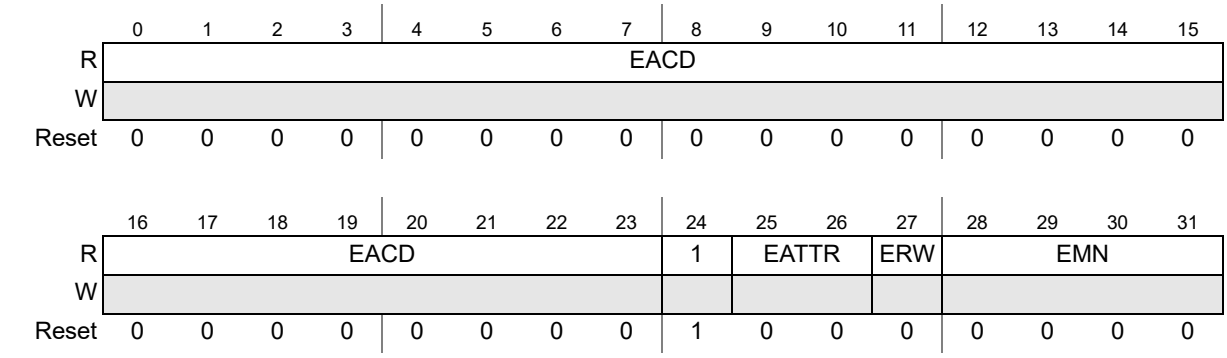


Figure 135. Error Detail Register, Bus Master *n* (EDR*n*)

Table 230. EDR*n* field descriptions

Field	Description
0:23 EACD	Error access control detail Indicates the region descriptor with the access error, where the region descriptor number matches the bit number. If EDR*n* contains a captured error and EACD is all zeroes, an access did not hit in any region descriptor. If only a single EACD bit is set, the access error was caused by a single non-overlapping region descriptor. If two or more EACD bits are set, the access error was caused by an overlapping set of region descriptors.
24	Reserved This read-only bit is reserved and always has the value ‘1’.
25:26 EATTR	Error attributes Indicates attribute information about the faulting reference. 00 User mode, instruction access 01 User mode, data access 10 Supervisor mode, instruction access 11 Supervisor mode, data access
27 ERW	Error read/write Indicates the access type of the faulting reference. 0 Read 1 Write
28:31 EMN	Error master number Indicates the logical bus master number of the faulting reference. This field is used to determine the bus master that generated the access error.

17.4.2.5 Region Descriptor *n*, Word 0 (RGD*n*_WORD0)

Address: 0x0400 + *n**0x10 (*n* = 0 to 1123) Access: Supervisor read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	SRTADDR															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	SRTADDR															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 136. Region Descriptor *n*, Word 0 (RGD*n*_WORD0)

Table 231. RGD*n*_WORD0 field descriptions

Field	Description
0:31 SRTADDR	Start address Defines the byte start address of the memory region.

17.4.2.6 Region Descriptor *n*, Word 1 (RGD*n*_WORD1)

Address: 0x0404 + *n**0x10 (*n* = 0 to 1123) Access: Supervisor read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	ENDADDR															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	ENDADDR															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 137. Region Descriptor *n*, Word 1 (RGD*n*_WORD1)

Table 232. RGD*n*_WORD1 field descriptions

Field	Description
0:31 ENDADDR	End address Defines the byte end address of the memory region. **Note:** The SMPU does not verify that ENDADDR ≥ SRTADDR.

17.4.2.7 Region Descriptor *n*, Word 2 Format 0 (RGD*n*_WORD2_FMT0)

RGD_WORD2 has two formats as determined by the RGD_WORD3.FMT field.

Note: RGD_WORD2_FMT0 applies when RGD_WORD3[FMT] = 0.

RGD_WORD2_FMT0 defines the access control rights of the memory region on a per master basis. The access control rights are defined by separate read and write permissions. For these fields, the bus master number refers to the *logical* bus master number.

For the access control rights, there are two flags per logical bus master:

- Read (r) permission refers to the ability to access the referenced memory address using an operand (data) fetch or an instruction fetch.
- Write (w) permission refers to the ability to update the referenced memory address using a store (data) instruction.

Each field consists of the two flags, with (r) being in the more significant position. For example, M0P has (r) as bit 0 and (w) as bit 1.

The bit settings are as follows:

- If set, the corresponding flag allows the specific access type (r = memory read, w = memory write).
- If cleared, the specific access type is not allowed.

Writes to this word clear the region descriptor's valid bit.

Address: 0x0408 + *n**0x10 (*n* = 0 to 1123) Access: Supervisor read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	M0P		M1P		M2P		M3P		M4P		M5P		M6P		M7P	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	M8P		M9P		M10P		M11P		M12P		M13P		M14P		M15P	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 138. Region Descriptor *n*, Word 2 Format 0 (RGD*n*_WORD2_FMT0)

Table 233. RGD*n*_WORD2_FMT0 field descriptions

Field	Description
0:1 M0P	Bus master 0 permissions
2:3 M1P	Bus master 1 permissions
4:5 M2P	Bus master 2 permissions
6:7 M3P	Bus master 3 permissions
8:9 M4P	Bus master 4 permissions
10:11 M5P	Bus master 5 permissions

Table 233. RGD*n*_WORD2_FMT0 field descriptions (continued)

Field	Description
12:13 M6P	Bus master 6 permissions
14:15 M7P	Bus master 7 permissions
16:17 M8P	Bus master 8 permissions
18:19 M9P	Bus master 9 permissions
20:21 M10P	Bus master 10 permissions
22:23 M11P	Bus master 11 permissions
24:25 M12P	Bus master 12 permissions
26:27 M13P	Bus master 13 permissions
28:29 M14P	Bus master 14 permissions
30:31 M15P	Bus master 15 permissions

17.4.2.8 Region Descriptor *n*, Word 3 (RGD*n*_WORD3)

The final word of the SMPU region descriptor contains the valid bit, the format select (FMT), plus other configuration bits.

Address: 0x040C + *n**0x10 (*n* = 0 to 1123) Access: Supervisor read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0												RO	0	CI	VLD
W																
R	0	0	0	0	0	0	0	0	0	0	0	FMT	RO	0	CI	VLD
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 139. Region Descriptor n, Word 3 (RGDn_WORD3)

Table 234. RGD*n*_WORD3 field descriptions

Field	Description
27 FMT	Region Descriptor Format This bit selects the configuration format (FMT0 or FMT1) for this region descriptor. **Note:** A specific module instance of the SMPU may support only the FMT0 format. If so, the FMT field is read-only with a fixed value of 0 and only RGD_WORD2_FMT0 applies. 0 Use format 0 (RGD_WORD2_FMT0) 1 Use format 1 (RGD_WORD2_FMT1)
28 RO	Read-Only This bit is intended to prevent accidental writes of an SMPU region descriptor. **Note:** Setting RO in an RGD locks all four words of the RGD until a system reset; the valid bit of the RGD and the global valid bit have no effect. 0 The region descriptor can be read or written. 1 Attempted writes to any location in the region descriptor are ignored with an error-free data transfer termination.
30 CI	Cache Inhibit Defines the cacheability attribute of the memory region. This bit is returned to the bus master that initiated the memory reference. **Note:** An address range specified in an SMPU region descriptor for a cacheable space (that is, CI = 0) must be defined with a starting address aligned on a 0-modulo-32 byte address and with a multiple of the 32 byte cache line size. 0 References to this region can be cached. 1 References to this region cannot be cached.
31 VLD	Valid Signals that the region descriptor is valid. Any write to RGD*n*_WORD0–2 clears this bit. 0 Region descriptor is invalid. 1 Region descriptor is valid.

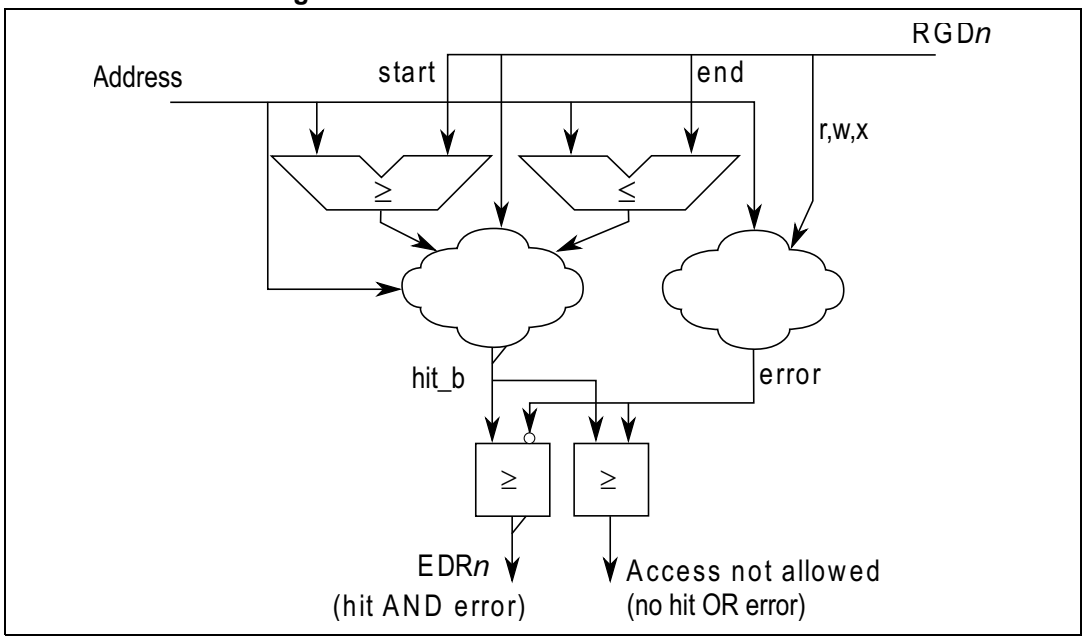
17.5 Functional description

In this section, the functional operation of the SMPU is detailed, including the operation of the access evaluation macro and the handling of error-terminated bus cycles.

17.5.1 Access evaluation macro

The basic operation of the SMPU is performed in the access evaluation macro, a hardware structure replicated in the two-dimensional connection matrix. As shown in *Figure 140*, the access evaluation macro inputs the crossbar bus address phase signals and the contents of a region descriptor (RGD*n*), then performs two major functions: region hit determination and detection of an access protection violation. *Figure 140* shows a functional block diagram.

Figure 140. SMPU access evaluation macro



17.5.1.1 Hit determination

To determine if the current reference hits in the given region, two magnitude comparators are used with the region's start and end addresses. The boolean equation for this portion of the hit determination is:

```
region_hit = ((addr[0:31] >= RGDn_Word0[SRTADDR]) &
     (addr[0:31] <= RGDn_Word1[ENDADDR])) &
RGDn_Word3[VLD]
```

where addr is the current reference address, RGDn_Word0.SRTADDR and RGDn_Word1.ENDADDR are the start and end addresses, and RGDn_Word3.VLD is the valid bit.

Note: The SMPU does not verify that ENDADDR ≥ SRTADDR.

Region hit determination is based only on an address comparison of the first byte being accessed; that is, the SMPU does not check the size of the access to make sure it fits within an allowed region.

17.5.1.2 Privilege violation determination

While the access evaluation macro is determining the region hit, the logic is also evaluating whether the current access is allowed by the permissions defined in the region descriptor. Using the master signal, a set of effective permissions is generated from the relevant fields in the region descriptor. The protection violation logic then evaluates the access against the effective permissions, using the specification shown in *Table 235*.

Table 235. Protection violation definition

Access type	Access permissions		Protection violation?
	r	w	
Instruction fetch read	0	—	Yes, no read permission
	1	—	No, access is allowed
Data read	0	—	Yes, no read permission
	1	—	No, access is allowed
Data write	—	0	Yes, no write permission
	—	1	No, access is allowed

17.5.2 Putting it all together and error terminations

For each slave port monitored, the SMPU performs a reduction-AND of all the individual terms from each access evaluation macro. This expression then terminates the bus cycle with an error and reports an access error for three conditions:

1. The access does not hit in any region descriptor.
2. The access hits in a single region descriptor and that region has a protection violation.
3. The access hits in multiple (overlapping) regions and all regions have protection violations.

As shown in the third condition, granting permission is a higher priority than denying access for overlapping regions. This approach is more flexible to system software in region descriptor assignments. For an example of the use of overlapping region descriptors, see *Section 17.7*.

17.6 Initialization information

At system startup, load the required number of region descriptors, including setting RGD*n*_Word3.VLD. Setting CESR0.GVLD enables the module.

If the system requires that all the loaded region descriptors be enabled simultaneously, first ensure that the entire SMPU is disabled (CESR0.GVLD = 0). Next, load the appropriate region descriptors (RGD*n*), including the setting of the RGDn_Word3.VLD bits. Finally, set CESR0.GVLD to enable the entire module.

A region descriptor must be set to allow access to the SMPU registers if further changes are needed.

17.7 Application information

In an operational system, interfacing with the SMPU is generally classified into the following activities:

- Creating a new memory region: Load the appropriate region descriptor into an available RGD*n*, using four sequential 32-bit writes. The hardware assists in the

maintenance of the valid bit, so if this approach is followed, there are no coherency issues with the multi-cycle descriptor writes.

- Modifying a region descriptor: Load the updates into the region descriptor using sequential 32-bit writes. Writing to Word3 re-enables the region descriptor valid bit. The hardware assists in the maintenance of the valid bit, so if this approach is followed, there are no coherency issues with the multi-cycle descriptor writes.
- Removing a region descriptor: Clearing RGD*n*_Word3.VLD deletes an existing region descriptor.
- Accessing the SMPU: Allocate a region descriptor to restrict SMPU access to supervisor mode from a specific master.
- Detecting an access error: The current bus cycle is terminated with an error response and EAR*n* and EDR*n* capture information on the faulting reference. The error-terminated bus cycle typically initiates an error response in the originating bus master. For example, a processor core may respond with a bus error exception, while a data movement bus master may respond with an error interrupt. The processor can retrieve the captured error address and detail information simply by reading E{A,D}R*n*. CESR0.MERR signals. The error registers contain the captured fault data.
- Overlapping region descriptors: Applying overlapping regions often reduces the number of descriptors required for a given set of access controls. In the overlapping memory space, the protection rights of the corresponding region descriptors are logically summed together (the boolean OR operator).

The following dual-core system example contains four bus masters: the two processors (CP0, CP1) and two DMA engines (DMA1, a traditional data movement engine transferring data between RAM and peripherals, and DMA2, a second engine transferring data to/from the RAM only). Consider the following region descriptor assignments:

Table 236. Overlapping region descriptor example

Region description	RGDn	CP0	CP1	DMA1	DMA2	System memory map space
CP0 code	0	rwx	r--	—	—	Flash
CP1 code	1	r--	rwx	—	—	
CP0 data & stack	2	rw-	—	—	—	RAM
CP0 → CP1 shared data	2–3	r--	r--	—	—	
CP1 → CP0 shared data	4					
CP1 data & stack	4	—	rw-	—	—	
Shared DMA data	5	rw-	rw-	rw	rw	
SMPU	6	rw-	rw-	—	—	Peripheral space
Peripherals	7	rw-	rw-	rw	—	

In this example, there are eight descriptors used to span nine regions in the three main spaces of the system memory map (flash, RAM, and peripheral space). Each region indicates the specific permissions for each of the four bus masters, and this definition provides an assigned set of shared, private, and executable memory spaces.

Of particular interest are the two overlapping spaces: region descriptors 2 & 3 and 3 & 4.

The space defined by RGD2 with no overlap is a private data and stack area that provides read/write access to CP0 only. The overlapping space between RGD2 and RGD3 defines a shared data space for passing data from CP0 to CP1, and the access controls are defined by the logical OR of the two region descriptors. Thus, CP0 has (rw- | r--) = (rw-) permissions, while CP1 has (--- | r--) = (r--) permission in this space. Both DMA engines are excluded from this shared processor data region. The overlapping spaces between RGD3 and RGD4 defines another shared data space, this one for passing data from CP1 to CP0. For this overlapping space, CP0 has (r-- | ---) = (r--) permission, while CP1 has (rw- | r--) = (rw-) permission. The non-overlapped space of RGD4 defines a private data and stack area for CP1 only.

The space defined by RGD5 is a shared data region, accessible by all four bus masters. Finally, the slave peripheral space mapped onto the peripheral bus is partitioned into two regions: one containing the SMPU's programming model accessible only to the two processor cores and the remaining peripheral region accessible to both processors and the traditional DMA1 master.

This simple example is intended to show one possible application of the capabilities of the SMPU in a typical system.

18 Intelligent AHB Gasket (IAHBG)

18.1 Introduction

This section details the intelligent bridging gasket between the fast and slow clock domains. The intelligent operation of the gasket initiates pending reads early and optimizes bursts to recover some of the performance that is lost on a registered interface between different clock domains. Pending reads begin even if there is not an hready to the master. Non-bursted writes only begin when the slave is IDLE, pending writes wait until IDLE.

The IAHBG complies with the AHBV6lite protocol, including an extension to support masters that do address retraction during the error termination sequence or abort a burst read due to an error termination or another non-error termination like cache inhibit termination.

Table 237 lists the available operation modes.

Table 237. Operation modes available

Operating mode (Master:Slave)	Comments
0:0	Flowthru (no register wall)
1:1	Master and slave at same operating frequency; gasket performance optimizations on by default
2:1	Master at twice frequency of slave; gasket performance optimizations on by default
1:2	Slave at twice frequency of master; gasket performance optimizations on by default
4:1	Master at four times frequency of slave; gasket performance optimizations disabled at this time
1:4	Slave at four times frequency of master; gasket performance optimizations disabled at this time

18.2 Timing modes

18.2.1 1:1

In the 1:1 timing mode the master is running at the same clock frequency as the slave. The pending read can either be a single or burst transaction.

18.2.2 2:1

In the 2:1 timing mode the master is running at twice the clock frequency of the slave. The 4:1 mode is very similar to the 2:1 mode. The pending read can either be a single or burst transaction.

18.2.3 1:2

In the 1:2 timing mode, the slave is running at twice the clock frequency of the master.The 1:4 mode is very similar to the 1:2 mode. The pending read can either be a single or burst transaction.

Because the slave is operating at a faster clock frequency than the master, the transactions from the master must be stored into a queue of data and response attribute registers. At a certain point during the master burst transactions, the queue is either emptied into the master for a burst read or it is emptied into the slave for a burst write.

19 Interrupt Controller (INTC)

19.1 Introduction

Note: For the chip-specific implementation details of this module's instances, please refer to the Section 7.3.5.

The INTC:

- Provides priority-based preemptive scheduling of interrupt requests
- Schedules interrupt requests (IRQs) from software and internal peripherals to the processor (PRC)
- Provides interrupt prioritization and preemption, interrupt masking, interrupt priority elevation, and protocol support

This scheduling scheme is suitable for statically scheduled hard real-time systems.

The INTC is targeted to work with a Power Architecture processor and is capable of processing high-demand interrupt sources where the Interrupt Service Routines (ISRs) nest to multiple levels, but it also can be used with other processors and applications.

For high-priority interrupt requests in these target applications, it is necessary to minimize the interval between the assertion of the interrupt request from the peripheral and the point at which the processor is performing useful work to service the interrupt request. The INTC supports this goal by providing a unique vector for each interrupt request source. It also provides 64 priorities so that lower priority ISRs do not delay the execution of higher priority ISRs. Since each individual application will have different priorities for each source of interrupt request, the priority of each interrupt request is configurable.

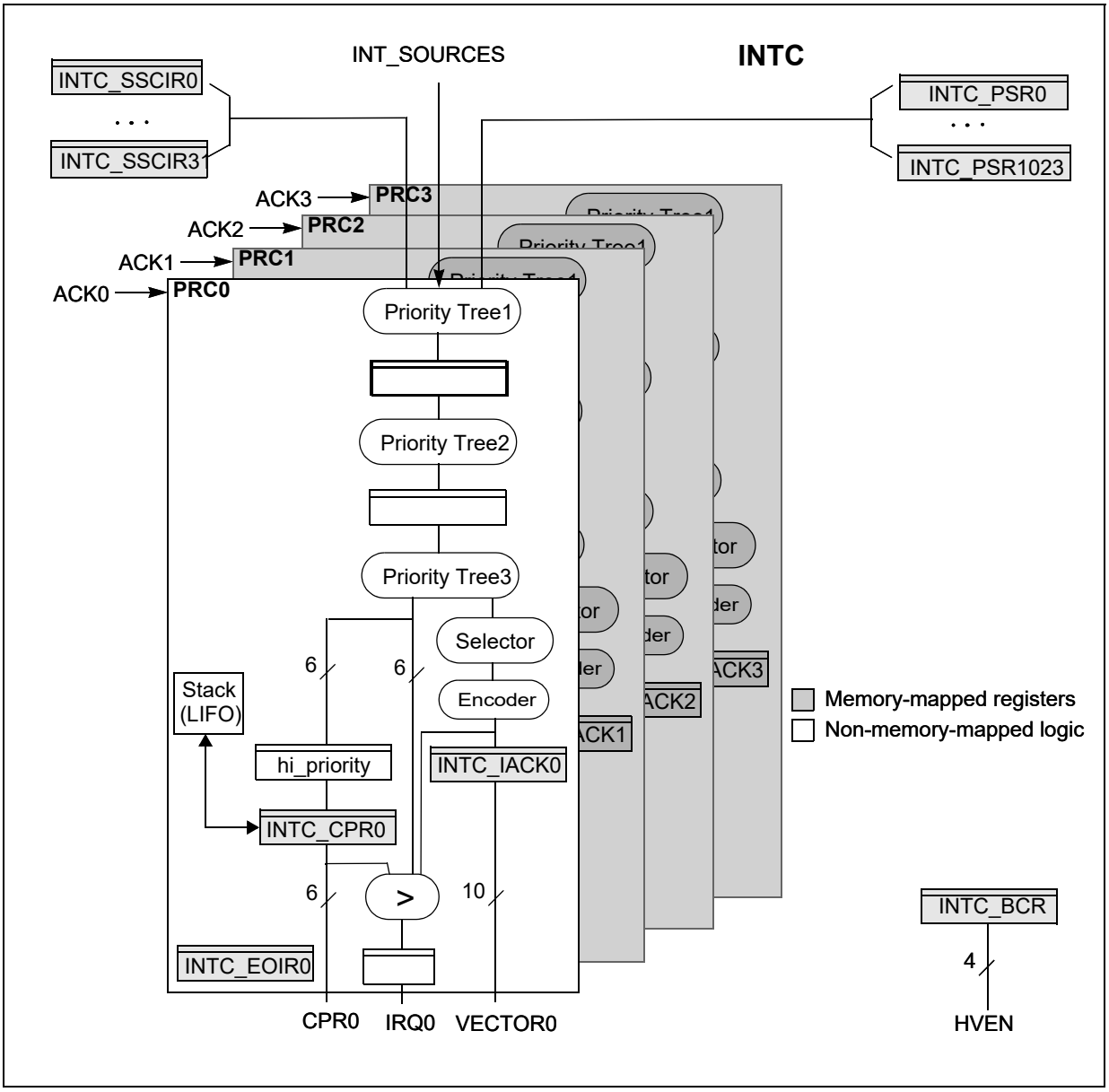
When multiple tasks share a resource, coherent accesses to that resource need to be supported. The INTC supports the Priority Ceiling Protocol for coherent accesses. By providing a modifiable priority mask, the priority can be raised temporarily so that all tasks which share the resource are unable to preempt each other.

The software-settable interrupt requests can also be used to separate the work involved in servicing an interrupt request into two parts, a high-priority portion and a low-priority portion. The high-priority portion is initiated by a peripheral interrupt request, but then the ISR can assert a software-settable interrupt request to finish the servicing in a lower priority ISR. Therefore these software-settable interrupt requests can be used instead of having the peripheral ISR schedule a task through the RTOS.

19.2 Block diagram

The block diagram for the INTC is shown in *Figure 141*.

Figure 141. INTC block diagram



19.3 Features

- Each peripheral interrupt source is software-steerable to processor 0
- 32 software-settable interrupt request sources
- 10-bit vector
  - Unique vector for each interrupt request source
  - Hardware connection to processor or read from register
- Each interrupt source can be programmed to one of 64 priorities
- Each interrupt source can be triggered by software
- Preemption
  - Preemptive prioritized interrupt requests to processor
  - ISR with higher priority preempts ISRs or tasks with lower priorities
  - Automatic pushing or popping of preempted priority to or from a LIFO
  - Ability to modify the ISR or task priority; modifying the priority can be used to implement the Priority Ceiling Protocol for accessing shared resources
  - 3 INTC clock cycles from interrupt request into interrupt request to CPU
- Low latency
  - 3 INTC clock cycles from receipt of interrupt request from peripheral to interrupt request to processor; four clock cycles from receipt of software request

19.4 Modes of operation

The interrupt controller operates with the following handshaking modes: software vector mode and hardware vector mode. The state of the hardware vector enable bit, INTC_BCR[HVEN0], determines which mode is used. In Power Architecture devices, software vector mode uses the autovector mode of the Processor for a single entry point for the ISR (16 bytes of vector space available), whereas the hardware vector mode uses the non-autovector mode where the vector offset from the INTC is provided to the Processor (4 bytes for each IRQ). In debug mode the interrupt controller operation is identical to its normal operation of software vector mode or hardware vector mode.

19.4.1 Software vector mode

In software vector mode, software (that is, the interrupt exception handler) must read a register in the INTC to obtain the vector associated with the interrupt request to the processor. The INTC will use software vector mode for the processor when the HVEN0 bit in INTC_BCR (see *Section 19.5.2.1*) is negated. The hardware vector enable signal to the processor is driven as negated when the HVEN0 bit is negated. The vector is read from the INTC_IACKR0 register (see *Section 19.5.2.3*). Reading the INTC_IACKR0 negates the interrupt request to the processor. Even if a higher priority interrupt request has arrived while waiting for this interrupt acknowledge, the interrupt request to the processor will negate for at least one clock. The reading also pushes the PRI value in INTC_CPR0 (see *Section 19.5.2.2*) onto the LIFO, and updates PRI in INTC_CPR0 with the new priority.

19.4.2 Hardware vector mode

In hardware vector mode, the hardware causes the first instruction that will be executed (when handling the interrupt request to the processor) to be an instruction that is specific to that vector. Therefore the interrupt exception handler is specific to a peripheral or software-settable interrupt request, rather than being common to all of them. The INTC will use hardware vector mode for the processor when the HVEN0 bit in INTC_BCR is asserted. The hardware vector enable signal to the processor is driven as asserted.

When the interrupt request to the processor asserts, the interrupt vector signal is updated. The value of that interrupt vector is the unique vector associated with the preempting peripheral or software-settable interrupt request. The vector value matches the value of the INTVEC field in the INTC_IACKR0.

The assertion of the interrupt acknowledge signal for the processor pushes the PRI value in the INTC_CPR0 register onto the LIFO and updates the PRI in the INTC_CPR0 register with the new priority. This pushing of the PRI value onto the LIFO and updating PRI in INTC_CPR0 does not occur when the associated interrupt acknowledge signal asserts and the INTC_SSCIRs (see *Section 19.5.2.5*) is written at a time such that the PRI value in the INTC_CPR0 register would need to be pushed and the previously last pushed PRI value would need to be popped simultaneously. In this case, PRI in INTC_CPR0 is updated with the new priority, and the LIFO is neither pushed or popped.

19.5 Memory map and register definition

19.5.1 Memory map

Table 238 is the INTC memory map.

The actual availability and number of registers are chip-specific. For this information, see the chapter that describes how modules are configured and connected.

Table 238. INTC memory map

Address offset	Register	Location
0x0000	INTC Block Configuration Register (INTC_BCR)	*Section 19.5.2.1*
0x0004–0x0007	Reserved(1)	
0x0008–0x000F	Reserved	
0x0010	INTC Current Priority Register for Processor 0 (INTC_CPR0)	*Section 19.5.2.2*
0x0014-0x0017	Reserved	
0x0018-0x001B	Reserved	
0x001C–0x001F	Reserved	
0x0020	INTC Interrupt Acknowledge Register for Processor 0 (INTC_IACKR0)	*Section 19.5.2.3*
0x0024-0x0027	Reserved	
0x0028-0x002B	Reserved	
0x002C–0x002F	Reserved	
0x0030	INTC End Of Interrupt Register for Processor 0 (INTC_EOIR0)	*Section 19.5.2.4*

Table 238. INTC memory map (continued)

Address offset	Register	Location
0x0034-0x0037	Reserved	
0x0038-0x003B	Reserved	
0x003C–0x003F	Reserved	
0x0040–0x005F	INTC Software Set/Clear Interrupt Register 0–31 (INTC_SSCIR0–INTC_SSCIR31)	*Section 19.5.2.5*
0x0060–0x085F	INTC Priority Select Registers 0–1023 (INTC_PSR0–INTC_PSR1023)	*Section 19.5.2.6*
0x0860–0x0FFF	Reserved	

1. Access to this location will not result in transfer error.

Note: A transfer error will be asserted if an access is attempted outside of the memory map or for any unimplemented registers. For example, if the design has only 512 interrupt sources and a source > 512 is accessed, a transfer error is asserted.

19.5.2 Register descriptions

With the exception of the INTC_SSCIR*n* and INTC_PSR*n* registers, all of the registers are 32-bit. Any combination of accessing the four bytes of a register with a single access is supported, provided that the access does not cross a register boundary. These supported accesses include types and sizes of 8 bits, aligned 16 bits, misaligned 16 bits to the middle two bytes, and aligned 32 bits.

Although INTC_SSIR*n* are 8 bits wide and INTC_PSR*n* are 16 bits wide, they can be accessed with a single 16-bit or 32-bit access, provided that the access does not cross a 32-bit boundary.

Some registers have specific exceptions to these rules, as outlined in their definitions.

In software vector mode, the effects of a read of the INTC Interrupt Acknowledge register (INTC_IACKR0) are the same regardless of the size of the read. In software or hardware vector mode, the size of a write to the INTC end-of-interrupt register (INTC_EOIR0) does not affect the operation of the write.

19.5.2.1 INTC Block Configuration Register (INTC_BCR)

The Block Configuration Register is used to configure options of the INTC.

Offset: 0x0000 Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	Reserved	0	0	0	Reserved	0	0	0	Reserved	0	0	0	HVEN0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 142. INTC Block Configuration Register (INTC_BCR)

Table 239. INTC_BCR field descriptions

Field	Description
31 HVEN0	Hardware vector enable Controls whether the INTC is in hardware vector mode or software vector mode. See *Section 19.4*, for the details of the handshaking with the processor in each mode. 0 Software vector mode 1 Hardware vector mode

19.5.2.2 INTC Current Priority Register for Processor 0 (INTC_CPR0)

The INTC_CPR0 masks any peripheral or software-settable interrupt request that is set at the same or lower priority as the current value of the INTC_CPR0[PRI] field from generating an interrupt request to the processor. When the INTC Interrupt Acknowledge register (INTC_IACKR0) is read in software vector mode or the interrupt acknowledge signal from the processor is asserted in hardware vector mode, the value of PRI is pushed onto the LIFO, and PRI is updated with the priority of the preempting interrupt request. When the INTC end-of-interrupt register (INTC_EOIR0) is written, the LIFO is popped into the INTC_CPR0 PRI field. An exception case in hardware vector mode to this behavior is described in *Section 19.4.2*.

The masking priority can be raised or lowered by writing to the PRI field, supporting the priority ceiling protocol. See *Section 19.7.5*.

Note: A store to modify the PRI field that closely precedes or follows an access to a shared resource may result in a non-coherent access to that resource. See Section 19.7.5.2, for example code to ensure coherency.

Offset: 0x0010 (INTC_CPR0) Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	PRI					
W																
Reset	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1

Figure 143. INTC Current Priority Register for Processor 0 (INTC_CPR0)

Table 240. INTC_CPR0 field descriptions

Field	Description
26:31 PRI	Priority of the currently executing ISR, according to the field values 63 (highest priority) down to 0 (lowest priority).

19.5.2.3 INTC Interrupt Acknowledge Register for Processor 0 (INTC_IACKR0)

The Interrupt Acknowledge Register for Processor 0 provides a value which can be used to load the address of an ISR from a vector table. The vector table can be composed of addresses of the ISRs specific to their respective interrupt vectors.

Also, in software vector mode, the INTC_IACKR0 has side effects from reads. Therefore, it must not be read speculatively while in this mode. The side effects are the same regardless of the size of the read. Reading the INTC_IACKR0 does not have side effects in hardware vector mode.

Offset: 0x0020 (INTC_IACKR0) Access: User write always; non-speculative read(1)

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	VTBA															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	VTBA				INTVEC										0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. When INTC_BCR[HVEN0] = 1, a read of the INTC_IACKR0 has no side effects.

Figure 144. INTC Interrupt Acknowledge Register for Processor 0 (INTC_IACKR0)

Table 241. INTC_IACKR0 field descriptions

Field	Description
0:19 VTBA	Vector table base address Must be set with the base address of the vector table prior to enabling interrupts.
20:29 INTVEC	Interrupt vector Vector of the peripheral or software-settable interrupt request that caused the interrupt request to the processor. When the interrupt request to the processor asserts, the INTVEC is updated, whether the INTC is in software or hardware vector mode.

19.5.2.4 INTC End of Interrupt Register for Processor 0 (INTC_EOIR0)

Writing to the INTC_EOIR0 signals the end of the servicing of the interrupt request. When the INTC_EOIR0 is written, the priority last pushed on the LIFO is popped into INTC_CPR0. The values and size of data written to the INTC_EOIR0 are ignored. Those values and sizes written to this register neither update the INTC_EOIR0 contents nor affect whether the LIFO pops. Typically, when you write to this register, write four all-zero bytes.

Offset: 0x0030 (INTC_EOIR0) Access: User write-only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R																
W	EOI															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R																
W	EOI															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 145. INTC End of Interrupt Register for Processor 0 (INTC_EOIR0)

Table 242. INTC_EOIR0 field descriptions

Field	Description
0:31 EOI	End of Interrupt Write four all-zero bytes to this field to signal the end of the servicing of an interrupt request.

19.5.2.5 INTC Software Set/Clear Interrupt Register *n* (INTC_SSCIR*n*)

The INTC_SSCIR registers support the setting or clearing of software-settable interrupt requests. These registers contain independent sets of bits to set and clear a corresponding flag bit by software. Unlike the SWT bit in the PSRs, the INTC_SSCIR registers do not require a read-modify-write. With the exception of being set by software, this flag bit behaves the same as a flag bit set within a peripheral. This flag bit generates an interrupt request within the INTC just like a peripheral interrupt request. Writing a 1 to SET will leave SET unchanged at 0 but will set CLR. Writing a 0 to SET will have no effect. CLR is the flag bit. Writing a 1 to CLR will clear it. Writing a 0 to CLR will have no effect. If a 1 is written to a

pair of SET and CLR bits at the same time, CLR is asserted, regardless of whether CLR was asserted before the write.

Figure 146 shows the structure of the first INTC_SSCIR. The other registers follow the same structure.

Offset: 0x0040 + *n**0x1 (*n* = 0 to 31) Access: User read/write

	0	1	2	3	4	5	6	7
R	0	0	0	0	0	0	0	CLR
W							SET	w1c
Reset	0	0	0	0	0	0	0	0

Figure 146. INTC Software Set/Clear Interrupt Register *n* (INTC_SSCIR*n*)

Table 243. INTC_SSCIR*n* field descriptions

Field	Description
6 SET	Set flag bit Writing a 1 will set the corresponding CLR bit. Writing a 0 will have no effect. Each SET is read as a 0.
7 CLR	Clear flag bit CLR is the flag bit. Writing a 1 to CLR will clear it provided that a 1 is not written simultaneously to its corresponding SET bit. Writing a 0 to CLR will have no effect. 0 Interrupt request not pending within INTC. 1 Interrupt request pending within INTC.

19.5.2.6 INTC Priority Select Register *n* (INTC_PSR*n*)

The Priority Select Registers support the selection of an individual priority for each source of interrupt request, and the routing of the request to one or more processors. The unique vector of each peripheral or software-settable interrupt request determines which INTC_PSR*n* is assigned to that interrupt request. The software-settable interrupt requests are assigned the lowest numbered vectors, and their priorities are configured in the lowest numbered INTC_PSR registers. The peripheral interrupt requests are assigned to vectors immediately following the software-settable interrupt requests, and their priorities are configured in the immediately following INTC_PSR registers. The peripheral interrupt requests are assigned vectors 32–1023, and their priorities are configured in INTC_PSR32–1023, respectively.

Figure 147 and *Figure 148* show the structure and numbering scheme of the first and last PSRs, respectively. The other registers follow the same numbering scheme.

Note: Unlike the peripheral interrupt request PSRs, there is no SWT bit in the PSRs corresponding to the software-settable interrupt requests.

Interrupts are not detected while the PSRs are being written.

Offset: 0x0060 + *n* * 0x2 (*n* = 0 to 31) Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	PRC_SEL0	Reserved	Reserved	Reserved	0	0	0	0	0	0	PRI					
W																
Reset	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 147. INTC Priority Select Register *n* (INTC_PSR*n*)

Table 244. INTC_PSR*n* field descriptions

Field	Description
0 PRC_SEL0	Processor 0 select bit If an interrupt source is enabled, PRC_SEL0 selects whether the interrupt request is to be sent to processor 0.
10:15 PRI	Priority select Selects the priority for the interrupt requests. PRI has values 63 (highest priority) down to 0 (lowest priority).

Offset: 0x0060 + *n* * 0x2 (*n*=32 to 1023) Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	PRC_SEL0	Reserved	Reserved	Reserved	0	0	0	SWT	0	0	PRI					
W																
Reset	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 148. INTC Priority Select Register *n* (INTC_PSR*n*)

Table 245. INTC_PSR*n* field descriptions

Field	Description
0 PRC_SEL0	Processor 0 select bit If an interrupt source is enabled, PRC_SEL0 selects whether the interrupt request is to be sent to processor 0.
7 SWT	This bit is only available in PSRs that do not correspond to the SSCIRs. The SWT bit supports triggering an interrupt by software rather than hardware. This is an alternative to setting a flag bit in a peripheral. This flag bit generates an interrupt request within the INTC just like a peripheral interrupt request. Writing a '1' to SWT*n* sets the bit. On the interrupt acknowledge cycle, the SWT*n* bit is cleared.
10:15 PRI	Priority select Selects the priority for the interrupt requests. PRI has values 63 (highest priority) down to 0 (lowest priority).

19.6 Functional description

The functional description involves the areas of interrupt request sources, priority management, and handshaking with the processor. In addition, spaces in the memory map have been reserved for other possible implementations of the INTC.

19.6.1 Interrupt request sources

The INTC has two types of interrupt requests, peripheral and software-settable. These interrupt requests can assert on any clock cycle.

The INTC has no spurious vector support. Therefore, if an asserted peripheral or software-settable interrupt request, whose PRI value in INTC_PSR*n* is higher than the PRI value in INTC_CPR0, negates before the interrupt request to the processor for that peripheral or software-settable interrupt request is acknowledged, then the interrupt request to the processor still can assert (or will remain asserted) for that peripheral or software-settable interrupt request. In this case, the interrupt vector will correspond to that peripheral or software-settable interrupt request. Also, the PRI value in INTC_CPR0 is updated with the corresponding PRI value in INTC_PSR*n*.

Furthermore, clearing the peripheral interrupt request's enable bit in the peripheral, or (alternatively) setting its mask bit, has the same consequences as clearing its flag bit. Setting its enable bit or clearing its mask bit while its flag bit is asserted has the same effect on the INTC as an interrupt event setting the flag bit.

19.6.1.1 Peripheral interrupt requests

An interrupt event in a peripheral's hardware sets a flag bit which resides in that peripheral. The interrupt request from the peripheral is driven by that flag bit.

The time from when the peripheral starts to drive its peripheral interrupt request to the INTC, to the time that the INTC starts to drive the interrupt request to the processor, is three clocks.

19.6.1.2 Software-settable interrupt requests

The software set/clear interrupt registers (INTC_SSCIR*n*) support the setting or clearing of software-settable interrupt requests. These registers contain independent sets of bits to set and clear a corresponding flag bit by software. An interrupt request is triggered by software by writing a 1 to a SET bit in INTC_SSCIR*n*. This write sets the corresponding CLR bit, which is a flag bit, resulting in the interrupt request. The interrupt request is cleared by writing a 1 to the CLR bit. Specific behavior includes the following:

- Writing a 1 to SET leaves SET unchanged at 0 but sets the flag bit (which is the CLR bit).
- Writing a 0 to SET has no effect.
- Writing a 1 to CLR clears the flag (CLR) bit.
- Writing a 0 to CLR has no effect.
- If a 1 is written to a pair of SET and CLR bits at the same time, the flag (CLR) is set, regardless of whether CLR was asserted before the write.

The time from the write to the SET bit, to the time that the INTC starts to drive the interrupt request to the processor, is four clocks.

The processor will be allowed to write to any of the SET bits in the Software Set/Clear Interrupt registers. The CLR bit will only be writable by the processor which has been assigned to handle that interrupt request, as determined by the setting of the INTC_PSR*n*[PRC_SEL0] bit.

19.6.1.3 Unique vector for each interrupt request source

Each peripheral and software-settable interrupt request is assigned a hardwired unique 10-bit vector. Software-settable interrupts 0–31 are assigned vectors 0–31, respectively. The peripheral interrupt requests are assigned vectors from 32 to a number as high as needed to cover all peripheral interrupt requests.

19.6.2 Priority management

The asserted interrupt requests are compared to each other based on their PRI*n* and PRC_SEL0 values set in INTC_PSR*n*. The result of that comparison also is compared to PRI in INTC_CPR0. The results of those comparisons are used to manage the priority of the ISR being executed by the processor. The LIFO also assists in managing that priority.

19.6.2.1 Current priority and preemption

The priority arbitrator, selector, encoder, and comparator logic shown in *Figure 141* are used to compare the priority of the asserted interrupt requests to the current priority. If the priority of any asserted peripheral or software-settable interrupt request is higher than the current priority for the processor, then the interrupt request to the processor is asserted. Also, a unique vector for the preempting peripheral or software-settable interrupt request is generated for INTC_IACKR0, and if in hardware vector mode, for the interrupt vector provided to the processor.

19.6.2.1.1 Priority tree

The priority tree for the processor compares all the priorities of all of the asserted interrupt requests assigned to the processor, both peripheral and software-settable. The output of the priority tree is the highest of those priorities assigned to the processor. Also, any interrupt requests which have this highest priority are output as asserted interrupt requests to the selector logic.

19.6.2.1.2 Selector

If only one interrupt request from the priority tree is asserted, then it is passed as asserted to the encoder logic. If multiple interrupt requests from the priority tree are asserted, then only the one with the lowest vector is passed as asserted to the encoder logic. The lower vector is chosen regardless of the time order of the assertions of the peripheral or software-settable interrupt requests.

19.6.2.1.3 Encoder

The encoder logic generates the unique 10-bit vector for the asserted interrupt request from the request selector subblock for the processor.

19.6.2.1.4 Comparator

The comparator logic compares the highest priority output from the priority arbitrator subblock with PRI in INTC_CPR0. If the comparator detects that this highest priority is

higher than the current priority, then it asserts the interrupt request to the processor. This interrupt request to the processor asserts whether this highest priority is raised above the value of PRI in INTC_CPR0, or the PRI value in INTC_CPR0 is lowered below this highest priority. This highest priority then becomes the new priority which is written to PRI in INTC_CPR0 when the interrupt request to the processor is acknowledged. Interrupt requests whose PRI in INTC_PSR*n* are zero will not cause a preemption because their PRI will not be higher than PRI in INTC_CPR0.

Another function of the comparator is to signal an update of INTC_IACKR0 with the vector number of the first interrupt that arrives that has a priority higher than the current priority. Once the vector number and priority are captured, they cannot be superseded by a higher priority interrupt until an update of INTC_CPR0 occurs. In software vector mode, higher priority interrupts can supersede the previously captured interrupt vector number and priority until the time a hardware or software interrupt acknowledge is processed.

19.6.2.2 Stack (LIFO)

The LIFO stores the preempted PRI values from INTC_CPR0. Therefore, because these priorities are stacked within the INTC, if interrupts need to be enabled during the ISR, at the beginning of the interrupt exception handler the PRI value in INTC_CPR0 does not need to be loaded from INTC_CPR0 and stored onto the context stack. Likewise at the end of the interrupt exception handler, the priority does not need to be loaded from the context stack and stored in INTC_CPR0.

The PRI value in INTC_CPR0 is pushed onto the LIFO when INTC_IACKR0 is read in software vector mode or the interrupt acknowledge signal from the processor is asserted in hardware vector mode. The priority is popped into PRI in INTC_CPR0 whenever INTC_EOIR0 is written. An exception case in hardware vector mode to this behavior is described in *Section 19.4.2*.

Although the INTC supports up to 64 priorities, an ISR executing with PRI in INTC_CPR0 equal to 63 will not be preempted. Therefore, the LIFO supports the stacking of 63 priorities. However, the LIFO is only 62 entries deep. An entry for a priority of 0 is not needed because of the ways that pushing onto a full LIFO and popping an empty LIFO are treated. If the LIFO is pushed 63 or more times than it is popped, the priorities first pushed are overwritten. A priority of 0 would be an overwritten priority. However, the LIFO will pop zeroes if it is popped more times than it is pushed. Therefore, although a priority of 0 was overwritten, it is regenerated with the popping of an empty LIFO.

19.6.3 Handshaking with processor

19.6.3.1 Software vector mode handshaking

19.6.3.1.1 Acknowledging interrupt request to processor

The software vector mode handshaking can be used with processors that only support an interrupt request to them, or processors that support both an interrupt request by itself as well as an interrupt request with an interrupt vector. The software vector mode handshaking even supports processors that always expect an interrupt vector with the interrupt request to them. Refer to *Figure 149*.

A timing diagram of the interrupt request and acknowledge handshaking in software vector mode, along with the handshaking near the end of the interrupt exception handler, is shown in *Figure 149*. The INTC examines the peripheral and software-settable interrupt requests.

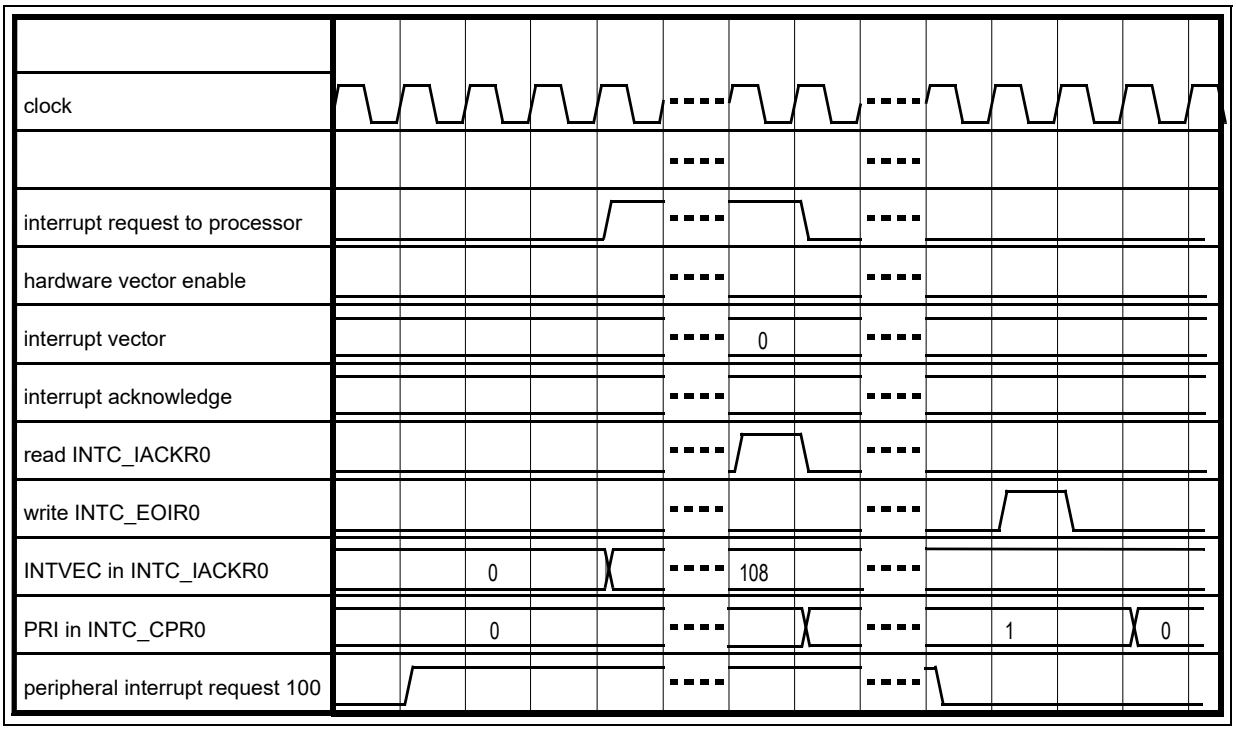
When it finds an asserted peripheral or software-settable interrupt request with a higher priority than PRI in INTC_CPR0, it asserts the interrupt request to the processor. The INTVEC field in INTC_IACKR0 is updated with the preempting interrupt request's vector when the interrupt request to the processor is asserted. The INTVEC field retains that value until the next time the interrupt request to the processor is asserted. The rest of the handshaking is described in *Section 19.4.1*.

19.6.3.1.2 End of interrupt exception handler

Before the interrupt exception handling completes, INTC_EOIR0 must be written. When it is written, the LIFO is popped so that the preempted priority is restored into PRI of INTC_CPR0. Before it is written, the peripheral or software-settable flag bit must be cleared so that the peripheral or software-settable interrupt request is negated.

When returning from the preemption, the INTC does not search for the peripheral or software-settable interrupt request whose ISR was preempted. Depending on how much the ISR has progressed, that interrupt request may no longer even be asserted. When PRI in INTC_CPR0 is lowered to the priority of the preempted ISR, the interrupt request for the preempted ISR or any other asserted peripheral or software-settable interrupt request at or below that priority will not cause a preemption. Instead, after the restoration of the preempted context, the processor will return to the instruction address that it had been going to execute next, before it was preempted. This next instruction is part of the preempted ISR or the interrupt exception handler's prolog or epilog.

Figure 149. Timing diagram of software vector mode handshaking

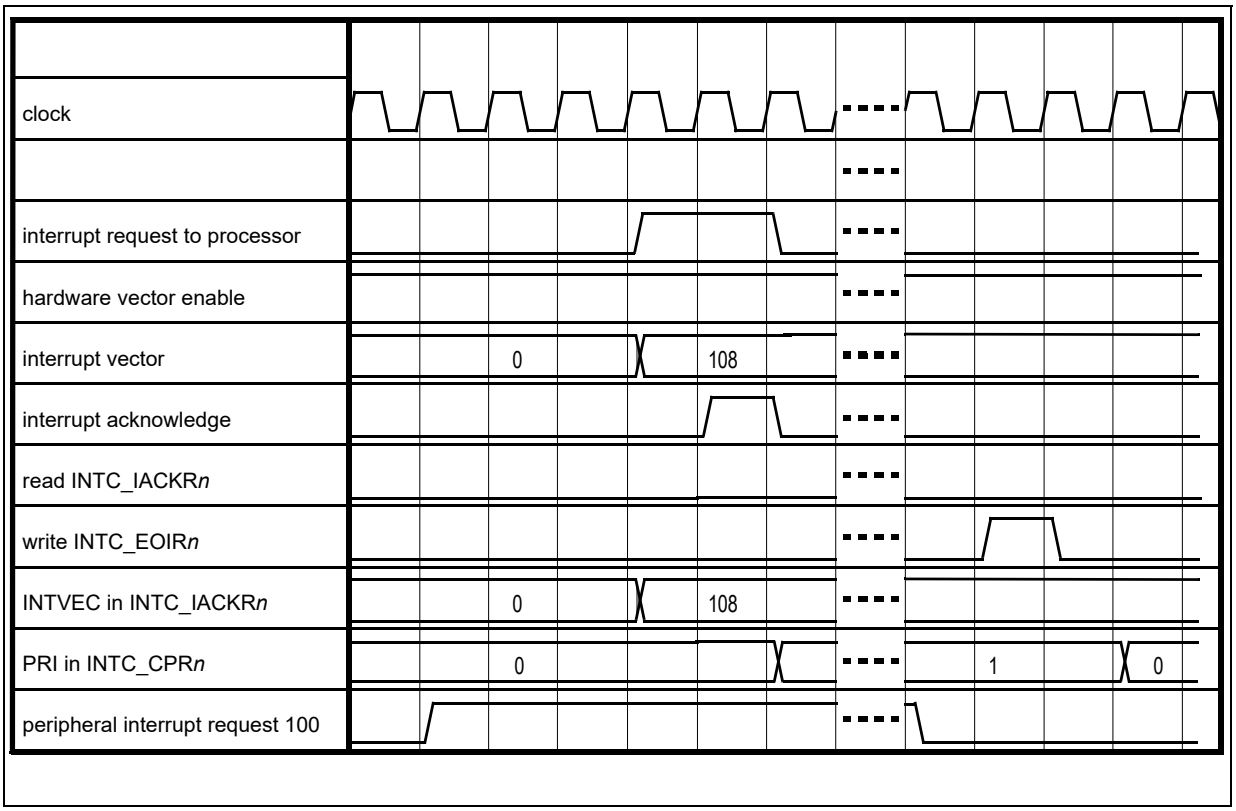


19.6.3.2 Hardware vector mode handshaking

A timing diagram of the interrupt request and acknowledge handshaking in hardware vector mode, along with the handshaking near the end of the interrupt exception handler, is shown in *Figure 150*. As in software vector mode, the INTC examines the peripheral and software-settable interrupt requests, and when it finds an asserted interrupt request with a higher priority than PRI in INTC_CPR0, it asserts the interrupt request to the processor. The INTVEC field in INTC_IACKR0 is updated with the preempting peripheral or software-settable interrupt request's vector when the interrupt request to the processor is asserted. The INTVEC field retains that value until the next time the interrupt request to the processor is asserted. In addition, the value of the interrupt vector to the processor matches the value of the INTVEC field in INTC_IACKR0. The rest of the handshaking is described in *Section 19.4.2*.

The handshaking near the end of the interrupt exception handler, that is the writing to INTC_EOIR0, is the same as in software vector mode. See *Section 19.6.3.1.2*.

Figure 150. Timing diagram for hardware vector mode handshaking



19.7 Initialization/application information

19.7.1 Initialization flow

After exiting reset, all of the PRI and PRC_SEL0 fields in INTC_PSR*n* are zero, and PRI in the INTC_CPR0 register is 31. These reset values will prevent the INTC from asserting the interrupt request to the processor. The enable or mask bits in the peripherals are reset such

that the peripheral interrupt requests are negated. An initialization sequence for allowing the peripheral and software-settable interrupt requests to cause an interrupt request to the processor is:

- Interrupt_request_initialization:
  - Configure HVEN0 in INTC_BCR.
  - Configure VTBA in INTC_IACKR0.
  - Raise the PRI fields and set the PRC_SEL0 fields in INTC_PSR*n*.
  - Set the enable bits or clear the mask bits for the peripheral interrupt requests.
  - Lower PRI in INTC_CPR0 to zero.
  - Enable processor recognition of interrupts.

19.7.2 Interrupt exception handler

These example interrupt exception handlers excerpts use Power Architecture assembly code.

19.7.2.1 Software vector mode

In software vector mode for Power Architecture, there are 16 bytes of vector space available at the single ISR entry point.

```
interrupt_exception_handler:
b  interrupt_exception_handler_continued
   # 16 bytes available, branch to continue

interrupt_exception_handler_continued:
code to create stack frame, save working register, and save SRR0 and SRR1
(Power Architecture Save/Restore registers) (not shown)

lis     r3,INTC_IACKR0@ha    # form adjusted upper half of INTC_IACKR0
                             # address
lwz     r3,INTC_IACKR0@l(r3) # load INTC_IACKR0, which clears request to
                             # processor
lwz     r3,0x0(r3)           # load address of ISR from vector table
wrteei  1                    # enable processor recognition of interrupts

code to save rest of context required by Power Architecture EABI (Embedded
Binary Application Interface) (not shown)

mtlr    r3           # move the INTC_IACKR0 address into the link register
blrl                 # branch to ISR; link register updated with epilog
                     # address

epilog:
code to restore most of context required by Power Architecture EABI (not
shown)

# Popping the LIFO after the restoration of most of the context and the
```

```
# disabling of processor recognition of interrupts eases the calculation of
# the maximum stack depth at the cost of postponing the servicing of the
# next interrupt request.
mbar                          # ensure store to clear flag bit has completed
lis     r3,INTC_EOIR0@ha      # form adjusted upper half of INTC_EOIR0
                              # address
li      r4,0x0                # form 0 to write to INTC_EOIR0
wrteei  0                     # disable processor recognition of interrupts
stw     r4,INTC_EOIR0@l(r3)   # store to INTC_EOIR0, informing INTC to lower
                              # priority

code to restore SRR0 and SRR1, restore working registers, and delete stack
frame (not shown)

rfi

vector_table_base_address:
address of ISR for interrupt with vector 0
address of ISR for interrupt with vector 1
   .
   .
   .
address of ISR for interrupt with vector 1022
address of ISR for interrupt with vector 1023

ISRn:
code to service the interrupt event (not shown)
code to clear flag bit which drives interrupt request to INTC (not shown)

blr        # return to epilog
```

19.7.2.2 Hardware vector mode

This interrupt exception handler is useful with processor and system bus implementations that support a hardware vector.

```
interrupt_exception_handlern:
b  interrupt_exception_handler_continuedn  # 16 bytes available, branch to
                                           # continue

interrupt_exception_handler_continuedn:
code to create stack frame, save working register, and save SRR0 and SRR1
(not shown)

wrteei  1       # enable processor recognition of interrupts

code to save rest of context required by Power Architecture EABI (not shown)
```

```
bl      ISRn    # branch to ISR for interrupt with vector n

epilog:
code to restore most of context required by Power Architecture EABI (not
shown)

# Popping the LIFO after the restoration of most of the context and the
# disabling of processor recognition of interrupts eases the calculation of
# the maximum stack depth at the cost of postponing the servicing of the
# next interrupt request.
mbar                        # ensure store to clear flag bit has completed
lis     r3,INTC_EOIR0@ha    # form adjusted upper half of INTC_EOIR0
                            # address
li      r4,0x0              # form 0 to write to INTC_EOIR0
wrteei  0                   # disable processor recognition of interrupts
stw     r4,INTC_EOIR0@l(r3) # store to INTC_EOIRn, informing INTC to lower
priority

SRR0 and SRR1, restore working registers, and delete stack frame (not shown)

rfi

ISRn:
code to service the interrupt event
code to clear flag bit which drives interrupt request to INTC

blr# branch to epilog
```

19.7.3 ISR, RTOS, and task hierarchy

The RTOS and all of the tasks under its control typically execute with PRI in INTC_CPR0 having a value of 0. The RTOS will execute the tasks according to whatever priority scheme it may have, but that priority scheme is independent and has a lower priority of execution than the priority scheme of the INTC. In other words, the ISRs execute above INTC_CPR0 priority 0 and outside the control of the RTOS while RTOS executes at INTC_CPR0 priority 0, and while the tasks execute at different priorities under the control of the RTOS, they also execute at INTC_CPR0 priority 0.

If a task shares a resource with an ISR and the Priority Ceiling Protocol (PCP) is being used to manage that shared resource, then the task's priority can be elevated in the INTC_CPR0 while the shared resource is being accessed.

An ISR whose PRI in INTC_PSR*n* has a value of 0 will not cause an interrupt request to the processor, even if its peripheral or software-settable interrupt request is asserted. For a peripheral interrupt request, not setting its enable bit or disabling the mask bit will cause it to remain negated, which consequently also will not cause an interrupt request to the processor. Since the ISRs are outside the control of the RTOS, this ISR will not run unless

called by another ISR or the interrupt exception handler, perhaps after executing another ISR.

19.7.4 Order of execution

An ISR with a higher priority can preempt an ISR with a lower priority, regardless of the unique vectors associated with each of their peripheral or software-settable interrupt requests. However, if multiple peripheral or software-settable interrupt requests are asserted, more than one of these interrupt requests has the highest priority and that priority is high enough to cause preemption, the INTC selects the one with the lowest unique vector regardless of the order in time that they asserted. However, the ability to meet deadlines with this scheduling scheme is no less than if the ISRs execute in the time order that their peripheral or software-settable interrupt requests asserted.

The example in *Table 246* shows the order of execution of both cases, ISRs with different priorities and ISRs with the same priority.

Table 246. Order of ISR execution example

		Code executing at end of step						
Step#	**Step Description**	**RTOS**	**ISR108(1)**	**ISR208**	**ISR308**	**ISR408**	**interrupt exception handler**	**PRI in INTC_CPR at end of step**
1	RTOS at priority 0 is executing.	X						0
2	Peripheral interrupt request 100 at priority 1 asserts. Interrupt taken.		X					1
3	Peripheral interrupt request 400 at priority 4 asserts. Interrupt taken.					X		4
4	Peripheral interrupt request 300 at priority 3 asserts.					X		4
5	Peripheral interrupt request 200 at priority 3 asserts.					X		4
6	ISR408 completes. Interrupt exception handler writes to INTC_EOIR0.						X	1
7	Interrupt taken. ISR208 starts to execute, even though peripheral interrupt request 300 asserted first.			X				3
8	ISR208 completes. Interrupt exception handler writes to INTC_EOIR0.						X	1
9	Interrupt taken. ISR308 starts to execute.				X			3

Table 246. Order of ISR execution example (continued)

		Code executing at end of step						
Step#	**Step Description**	**RTOS**	**ISR108[1]**	**ISR208**	**ISR308**	**ISR408**	**interrupt exception handler**	**PRI in INTC_CPR at end of step**
10	ISR308 completes. Interrupt exception handler writes to INTC_EOIR0.						X	1
11	ISR108 completes. Interrupt exception handler writes to INTC_EOIR0.						X	0
12	RTOS continues execution.	X						0

1. ISR108 executes for peripheral interrupt request 100 because the first eight ISRs are for software-settable interrupt requests.

19.7.5 Priority ceiling protocol

19.7.5.1 Elevating priority

The PRI field in INTC current priority register (INTC_CPR0) is elevated in the OSEK PCP to the ceiling of all of the priorities of the ISRs that share a resource. This protocol therefore allows coherent accesses of the ISRs to that shared resource.

For example, ISR1 has a priority of 1, ISR2 has a priority of 2, and ISR3 has a priority of 3. They all share the same resource. Before ISR1 or ISR2 can access that resource, they must raise the PRI value in INTC_CPR2, the ceiling of all of the ISR priorities. After they release the resource, they must lower the PRI value in INTC_CPR0 to prevent further priority inversion. If they do not raise their priority, then ISR2 can preempt ISR1, and ISR3 can preempt ISR1 or ISR2, possibly corrupting the shared resource. Another possible failure mechanism is deadlock if the higher priority ISR needs the lower priority ISR to release the resource before it can continue, but the lower priority ISR cannot release the resource until the higher priority ISR completes and execution returns to the lower priority ISR.

Using the PCP instead of disabling processor recognition of all interrupts eliminates the time when accessing a shared resource that all higher priority interrupts are blocked. For example, while ISR3 cannot preempt ISR1 while it is accessing the shared resource, all of the ISRs with a priority higher than 3 can preempt ISR1.

19.7.5.2 Ensuring coherency

19.7.5.2.1 Interrupt with blocked priority

A scenario can exist that can cause non-coherent accesses to the shared resource. As an example, ISR1 and ISR2 both share a resource. ISR1 has a lower priority than ISR2. ISR1 is executing, and it writes to the INTC_CPR0. The instruction following this store is a store to a value in a shared coherent data block. Either just before or at the same time as the first store, the INTC asserts the interrupt request to the processor because the peripheral interrupt request for ISR2 has asserted. As the processor is responding to the interrupt request from the INTC, and as it is aborting transactions and flushing its pipeline, it is possible (in some implementations) that both of these stores are executed. ISR2 thereby

assumes that it can access the data block coherently, but the data block has been corrupted.

OSEK uses the GetResource and ReleaseResource system services to manage access to a shared resource. To prevent this corruption of a coherent data block, modifications to PRI in INTC_CPR0 can be made by those system services with the code sequence:

Figure 151. OSEK Code Sequence

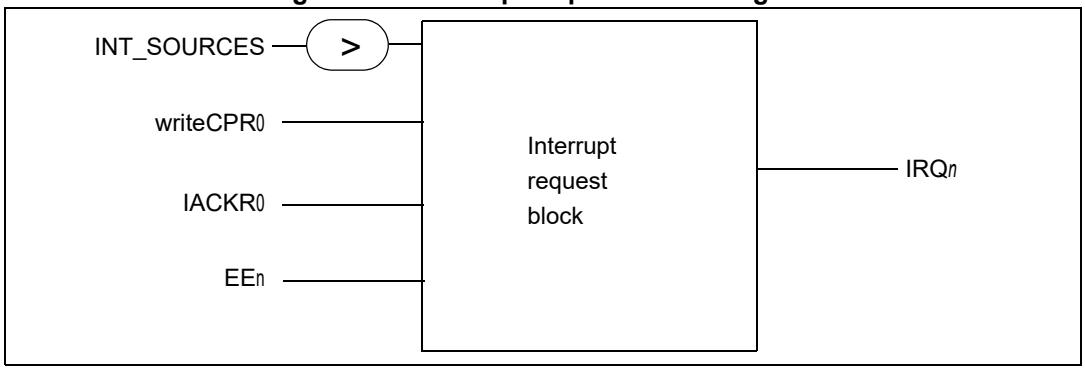
```
GetResource:
   wrteei  0       # disable external interrupts to the Processor
   raise   PRI     # Write to CPR, cache inhibit, guarded
   mbar            # flush out writes from store buffer
   wrteei  1       # enable external interrupts to the Processor
   isync           # re-fetch Processor pipeline

ReleaseResource:
   mbar            # flush out writes from store buffer
   wrteei  0       # disable external interrupts to the Processor
   lower   PRI     # Write to CPR, cache inhibit, guarded
   wrteei  1       # enable external interrupts to the Processor
```

19.7.5.2.1.1 Interrupt request to processor

Referencing *Figure 141*, the interrupt request logic to the processor is shown in *Figure 152*.

Figure 152. Interrupt request block diagram



Assuming that there are no IRQ*n* deasserted (no pending interrupt requests to the processor), if one of the external interrupt sources (INT_SOURCES) has a priority higher than INTC_CPR0, then interrupt request to the processor (IRQ*n*) is asserted. It will stay asserted until one of the following conditions is true:

- Interrupt acknowledge (IACKR0) is asserted
- If the EE*n* bit has been cleared by the wrteei instruction (see *Figure 151*), IRQ*n* will be re-evaluated when the processor writes to INTC_CPR0 while the processor ignores IRQ*n*

This provides a safe way to guarantee that no interrupts will be recognized by the processor while updating INTC_CPR0.

19.7.5.2.2 Raised priority preserved

Before the instruction and after the GetResource system service executes, all pending transactions are complete. These pending transactions can include an ISR for a peripheral or software-settable interrupt request whose priority was equal to or lower than the raised priority. Also, during the epilog of the interrupt exception handler for this preempting ISR, the raised priority has been restored from the LIFO to PRI in INTC_CPR0. The shared coherent data block now can be accessed coherently. *Figure 153* shows the timing diagram for this scenario, and *Table 247* explains the events. The example is for software vector mode. Except for the method of retrieving the vector and acknowledging the interrupt request to the processor, hardware vector mode is identical.

Figure 153. Timing diagram of raised priority preserved

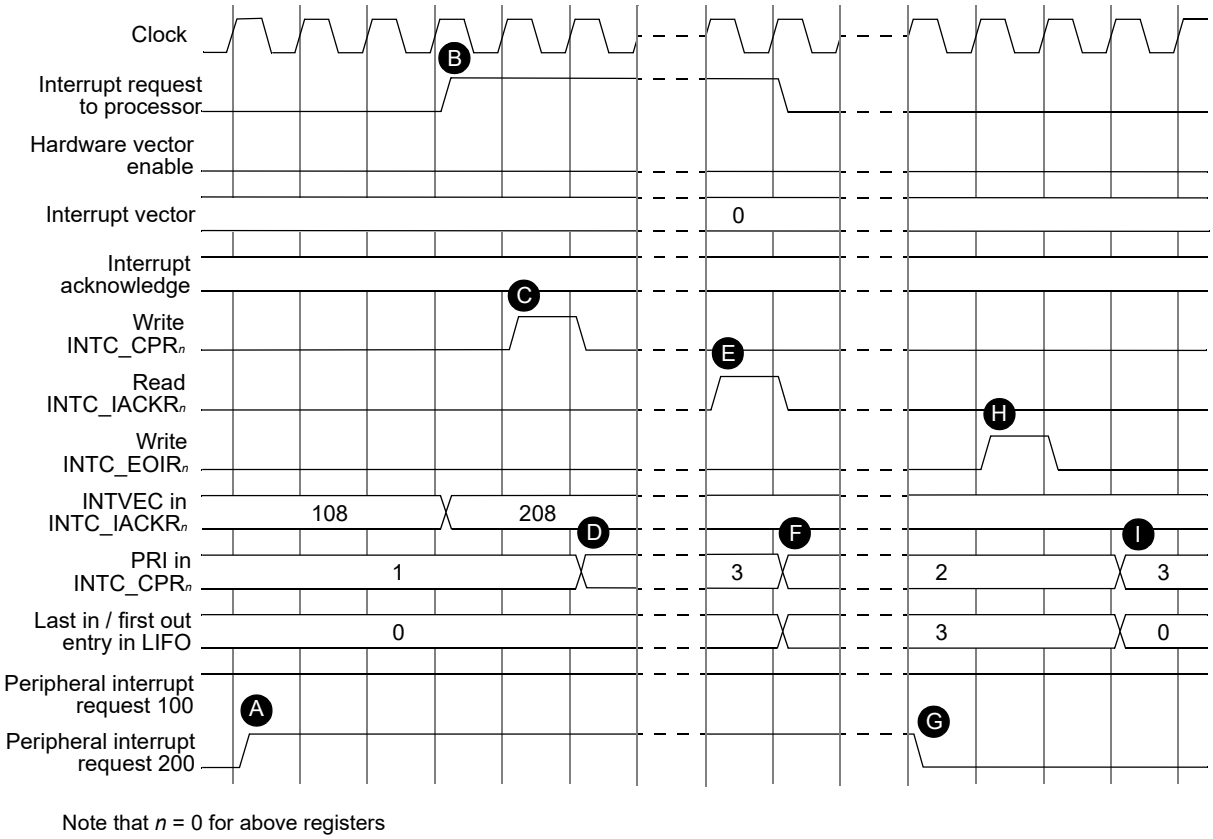


Table 247. Raised priority preserved events

Event	Description
A	Peripheral interrupt request with name equal to 200, asserts during execution of ISR108 running at priority 1.
B	Interrupt request to processor asserts. INTVEC in INTC_IACKR0 updates with vector for that peripheral interrupt request.
C	ISR108 writes to INTC_CPR0 to raise priority to 3 before accessing shared coherent data block.

Table 247. Raised priority preserved events (continued)

Event	Description
D	PRI in INTC_CPR0 now at 3, reflecting the write. This write, just before accessing data block, is the last instruction the processor executes before being interrupted.
E	Interrupt exception handler prolog acknowledges interrupt by reading INTC_IACKR0.
F	PRI of 3 pushed onto LIFO. PRI in INTC_CPR0 updates to 2, the priority of ISR208.
G	ISR208 clears its flag bit, deasserting its peripheral interrupt request.
H	Interrupt exception handler epilog writes to INTC_EOIR0.
I	LIFO pops 3, restoring the raised priority onto PRI in INTC_CPR0. Next value to pop from LIFO is the priority from before peripheral interrupt request with name equal to 100 interrupted. ISR108 now can access data block coherently after interrupt exception handler executes rfi instruction.

19.7.6 Selecting priorities according to request rates and deadlines

The selection of the priorities for the ISRs can be made using Rate Monotonic Scheduling (RMS) or a superset of it, Deadline Monotonic Scheduling (DMS). In RMS, the ISRs which have higher request rates have higher priorities. In DMS, if the deadline is before the next time the ISR is requested, then the ISR is assigned a priority according to the time from the request for the ISR to the deadline, not from the time of the request for the ISR to the next request for it.

For example, ISR1 executes every 100 µs, ISR2 executes every 200 µs, and ISR3 executes every 300 µs. ISR1 has a higher priority than ISR2 which has a higher priority than ISR3. However, if ISR3 has a deadline of 150 µs, then it has a higher priority than ISR2.

The INTC supports 64 priorities, which may be many fewer than the number of ISRs. In this case, the ISRs should be grouped with other ISRs that have similar deadlines. For example, a priority could be allocated for every time the request rate doubles. ISRs with request rates around 1 ms would share a priority, ISRs with request rates around 500 µs would share a priority, ISRs with request rates around 250 µs would share a priority, and so on. With this approach, a range of ISR request rates of 2^{16} could be covered, regardless of the number of ISRs.

Reducing the number of priorities does cause some priority inversion, which reduces the processor's ability to meet its deadlines. However, reducing the number of priorities can reduce the size and latency through the interrupt controller. It also allows easier management of ISRs with similar deadlines that share a resource. They can be placed at the same priority without any further priority inversion, and they do not need to use the PCP to access the shared resource.

19.7.7 Software-settable interrupt requests

The software-settable interrupt requests can be used in two ways. They can be used to schedule a lower priority portion of an ISR.

19.7.7.1 Scheduling a lower priority portion of an ISR

A portion of an ISR needs to be executed at the PRI value in INTC_PSR*n*, which becomes the PRI value in INTC_CPR0 with the interrupt acknowledge. The ISR, however, can have a portion of it which need not be executed at this higher priority. Therefore, executing this later

portion which need not be executed at this higher priority can prevent the execution of ISRs which do not have a higher priority than the earlier portion of the ISR but do have a higher priority than the later portion of the ISR needs. This preemptive scheduling inefficiency reduces the processor's ability to meet its deadlines.

One option is for the ISR to complete the earlier higher priority portion, but then schedule through the RTOS a task to execute the later lower priority portion. However, some RTOSs can require a large amount of time for an ISR to schedule a task. Therefore, a second option is for the ISR, after completing the higher priority portion, to set a SET bit in INTC_SSCIR*n*. Writing a 1 to SET causes a software-settable interrupt request. This software-settable interrupt request, which usually will have a lower PRI value in the INTC_PSR*n*, therefore will not cause preemptive scheduling inefficiencies.

19.7.8 Lowering priority within an ISR

In implementations without the software-settable interrupt requests in INTC_SSCIR*n*, one way (besides scheduling a task through an RTOS) to prevent preemptive scheduling inefficiencies with an ISR whose work spans multiple priorities (as described in *Section 19.7.7.1*) is to lower the current priority. However, the INTC has a LIFO whose depth is determined by the number of priorities.

Note: Lowering the PRI value in INTC_CPR0 within an ISR to below the ISR's corresponding PRI value in INTC_PSRn allows more preemptions than the depth of the LIFO can support.

Therefore, through its use of the LIFO the INTC does not support lowering the current priority within an ISR as a way to avoid preemptive scheduling inefficiencies.

19.7.9 Negating an interrupt request outside of its ISR

19.7.9.1 Negating an interrupt request as a side effect of an ISR

Some peripherals have flag bits which can be cleared as a side effect of servicing a peripheral interrupt request. For example, reading a specific register can clear the flag bits, and consequently their corresponding interrupt requests too. This clearing as a side effect of servicing a peripheral interrupt request can cause the negation of other peripheral interrupt requests besides the peripheral interrupt request whose ISR presently is executing. This negating of a peripheral interrupt request outside of its ISR can be a desired effect.

19.7.9.2 Negating multiple interrupt requests in one ISR

An ISR can clear other flag bits besides its own flag bit. One reason that an ISR clears multiple flag bits is because it serviced those other flag bits, and therefore the ISRs for these other flag bits do not need to be executed.

19.7.9.3 Proper setting of interrupt request priority

Whether an interrupt request negates outside of its own ISR due to the side effect of an ISR execution or the intentional clearing of a flag bit, the priorities of the peripheral or software-settable interrupt requests for these other flag bits must be selected properly. Their PRI values in INTC_PSR*n* must be selected to be at or lower than the priority of the ISR that cleared their flag bits. Otherwise, those flag bits still can cause the interrupt request to the processor to assert. Furthermore, the clearing of these other flag bits also has the same timing relationship to the writing to INTC_SSCIR*n* as the clearing of the flag bit that caused the present ISR to be executed. Refer to *Section 19.6.3.1.2* for more information.

A flag bit whose enable bit or mask bit is negating its peripheral interrupt request can be cleared at any time, regardless of the peripheral interrupt request's PRI value in INTC_PSR*n*.

19.7.10 Examining LIFO contents

Normally the user does not need to know the contents of the LIFO, or even how deep the LIFO is nested. Although the LIFO contents are not memory-mapped, the user can read the contents by popping the LIFO and reading the PRI field in the INTC current priority register (INTC_CPR0). Disabling processor recognition of interrupts while examining the LIFO contents provides a coherent view of the preempted priorities. The code sequence is:

```
pop_lifo:
   store to INTC_EOIR0
   load INTC_CPR0, examine PRI, and store onto stack
   if PRI is not zero or value when interrupts were enabled, branch to
pop_lifo
```

When the examination is complete, the LIFO can be restored using this code sequence:

```
push_lifo:
   load stacked PRI value and store to INTC_CPR0
   load INTC_IACKR0
   if stacked PRI values are not depleted, branch to push_lifo
```

Note: Reading the INTC_IACKR0 acknowledges the interrupt request to the processor and updates the INTC_CPR0[PRI] with the priority of the preempting interrupt request. If the processor recognition of interrupts is disabled during the LIFO restoration, interrupt requests to the processor can go undetected. However, because the peripheral or software-settable interrupt requests are not cleared, the peripheral interrupt request to the processor re-asserts when INTC_CPR0[PRI] is lower than the priorities of those peripheral or software-settable interrupt requests.

19.8 Interrupt sources

The list of interrupt sources is chip-specific. For this list, see the chapter that describes how modules are configured and connected.

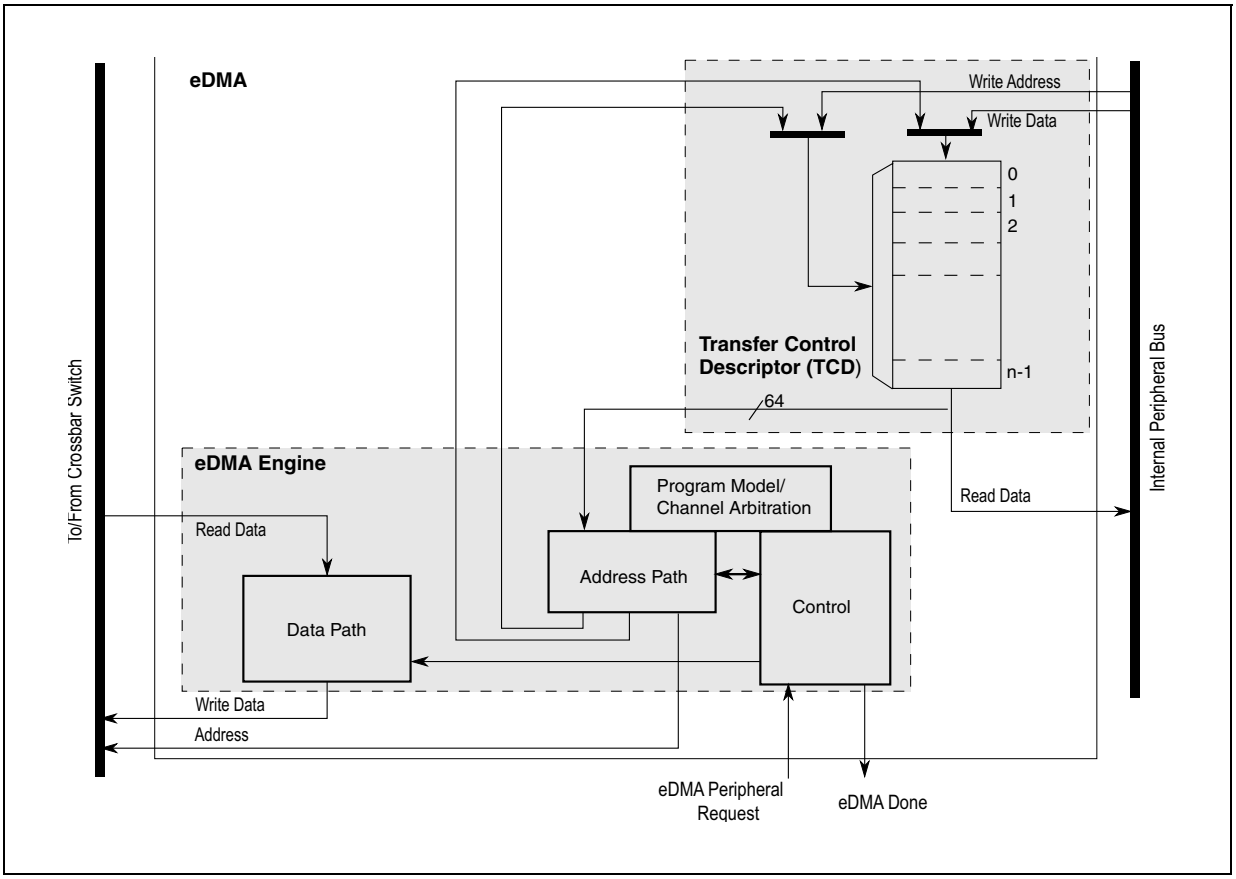
20 Enhanced Direct Memory Access (eDMA)

20.1 Introduction

The enhanced direct memory access (eDMA) controller is a second-generation module capable of performing complex data transfers with minimal intervention from a host processor. The hardware microarchitecture includes a DMA engine that performs source- and destination-address calculations, and the actual data-movement operations, along with local memory containing transfer control descriptors for each of the 32 channels.

Figure 154 illustrates the eDMA module.

Figure 154. eDMA block diagram



20.1.1 Features

The eDMA is a highly-programmable data-transfer engine optimized to minimize the intervention from the host processor. It is intended for use in applications where the data

size to be transferred is statically known and not defined within the data packet itself. The eDMA module features:

- All data movement via dual-address transfers: read from source, write to destination:
  Programmable source and destination addresses and transfer size, plus support for enhanced addressing modes
- 32-channel implementation that performs complex data transfers with minimal intervention from a host processor
  – Internal data buffer, used as temporary storage to support 16- and 32-byte burst transfers
  – Connections to the crossbar switch for bus mastering the data movement
- Transfer control descriptor (TCD) organized to support two-deep, nested transfer operations
  – 32-byte TCD stored in local memory for each channel
  – An inner data transfer loop defined by a minor byte transfer count
  – An outer data transfer loop defined by a major iteration count
- Channel activation via one of three methods:
  – Explicit software initiation
  – Initiation via a channel-to-channel linking mechanism for continuous transfers
  – Peripheral-paced hardware requests (one per channel)
- Fixed-priority and round-robin channel arbitration
- Channel completion reported via optional interrupt requests
  – One interrupt per channel, optionally asserted at completion of major iteration count
  – Optional error terminations per channel and logically summed together to form one error interrupt to the interrupt controller
- Optional support for scatter/gather DMA processing
- Support for complex data structures
- Support to cancel transfers via software

Throughout this chapter, *n* is used to reference the channel number.

20.2 Modes of operation

The eDMA supports the following modes of operation.

20.2.1 Normal mode

In Normal mode, the eDMA transfers data between a source and a destination. The source and destination can be a memory block or an I/O block capable of operation with the eDMA.

A service request initiates a transfer of a specific number of bytes (NBYTES) as specified in the transfer control descriptor (TCD). The minor loop is the sequence of read-write operations that transfers these NBYTES per service request. Each service request executes one iteration of the major loop, which transfers NBYTES of data.

20.2.2 Debug mode

DMA operation is configurable in Debug mode via the control register:

- If CR[EDBG] is cleared, the DMA continues to operate.
- If CR[EDBG] is set, the eDMA stops transferring data. If Debug mode is entered while a channel is active, the eDMA continues operation until the channel retires.

20.2.3 Wait mode

Before entering Wait mode the DMA attempts to complete its current transfer. After the transfer completes the device enters Wait mode.

20.3 Memory map/register definition

The eDMA's programming model is partitioned into two regions:

- The first region defines a number of registers providing control functions.
- The second region corresponds to the local transfer control descriptor memory.

Some registers are implemented as two 32-bit registers, and include an H and L suffix, signaling the high and low portions of the control function. *Table 248* provides a 32-bit view of the eDMA's memory map.

Each channel requires a 32-byte transfer control descriptor for defining the desired data movement operation. The channel descriptors are stored in the local memory in sequential order: channel 0 to channel 31. Each TCD*n* definition is presented as 11 registers of 16 or 32 bits.

Reading reserved bits in a register returns the value of zero and writes to reserved bits in a register are ignored. Reading or writing a reserved memory location generates a bus error.

Table 248. DMA memory map

Address offset	Register name	Width (in bits)	Location
0x0000	Control Register (DMA_CR)	32	*Section 20.3.1*
0x0004	Error Status Register (DMA_ES)	32	*Section 20.3.2*
0x0008-0x000B	Reserved		
0x000C	Enable Request Register Low (DMA_ERQL)	32	*Section 20.3.3*
0x0010-0x0013	Reserved		
0x0014	Enable Error Interrupt Register Low (DMA_EEIL)	32	*Section 20.3.4*
0x0018	Set Enable Request Register (DMA_SERQ)	8	*Section 20.3.5*
0x0019	Clear Enable Request Register (DMA_CERQ)	8	*Section 20.3.6*
0x001A	Set Enable Error Interrupt Register (DMA_SEEI)	8	*Section 20.3.7*
0x001B	Clear Enable Error Interrupt Register (DMA_CEEI)	8	*Section 20.3.8*
0x001C	Clear Interrupt Request Register (DMA_CINT)	8	*Section 20.3.9*
0x001D	Clear Error Register (DMA_CERR)	8	*Section 20.3.10*

Table 248. DMA memory map (continued)

Address offset	Register name	Width (in bits)	Location
0x001E	Set START Bit Register (DMA_SSRT)	8	*Section 20.3.11*
0x001F	Clear DONE Status Bit Register (DMA_CDNE)	8	*Section 20.3.12*
0x0020-0x0023	Reserved		
0x0024	Interrupt Request Register Low (DMA_INTL)	32	*Section 20.3.13*
0x0028-0x002B	Reserved		
0x002C	Error Register Low (DMA_ERRL)	32	*Section 20.3.14*
0x0030-0x0033	Reserved		
0x0034	Hardware Request Status Register Low (DMA_HRSL)	32	*Section 20.3.15*
eDMA channel registers per channel (n=0 to 31)			
0x1*n+0x0100	Channel n Priority Register (DMA_DCHPRIn)	8	*Section 20.3.16*
0x0120-0x013F	Reserved		
0x1*n+0x0140	Channel n Master ID Register (DMA_DCHMIDn)	8	*Section 20.3.17*
0x0160-0x0FFF	Reserved		
0x20*n+0x1000	TCDn Source Address (DMA_TCDn_SADDR)	32	*Section 20.3.18*
0x20*n+0x1004	TCDn Transfer Attributes (DMA_TCDn_ATTR)	16	*Section 20.3.19*
0x20*n+0x1006	TCDn Signed Source Address Offset (DMA_TCDn_SOFF)	16	*Section 20.3.20*
0x20*n+0x1008	TCDn Minor Byte Count (Minor Loop Disabled) (DMA_TCDn_NBYTES_MLNO)	32	*Section 20.3.21*
	(DMA_TCDn_NBYTES_MLOFFNO)	32	*Section 20.3.22*
	(DMA_TCDn_NBYTES_MLOFFYES)	32	*Section 20.3.23*
0x20*n+0x100C	(DMA_TCDn_SLAST)	32	*Section 20.3.24*
0x20*n+0x1010	(DMA_TCDn_DADDR)	32	*Section 20.3.25*
0x20*n+0x1014	(DMA_TCDn_CITER_ELINKYES)	16	*Section 20.3.26*
	TCDn Current Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCDn_CITER_ELINKNO)	16	*Section 20.3.27*
0x20*n+0x1016	(DMA_TCDn_DOFF)	16	*Section 20.3.28*
0x20*n+0x1018	(DMA_TCDn_DLASTSGA)	32	*Section 20.3.29*
0x20*n+0x101C	(DMA_TCDn_BITER_ELINKYES)	16	*Section 20.3.30*
	(DMA_TCDn_BITER_ELINKNO)	16	*Section 20.3.31*
0x20*n+0x101E	(DMA_TCDn_CSR)	16	*Section 20.3.32*
0x1400-0x17FF	Reserved		

20.3.1 Control Register (DMA_CR)

The CR defines the basic operating configuration of the DMA.

The DMA arbitrates channel service requests in two groups (0, 1) of 16 channels each: Group 1 contains channels 31 to 16, and group 0 contains channels 15 to 0.

Arbitration within a group can be configured to use a fixed-priority or a round-robin scheme. In fixed-priority arbitration, the highest priority channel requesting service is selected to execute. The channel priority registers assign the priorities (see the DCHPRI*n* registers). In Round-Robin Arbitration mode, the channel priorities are ignored, and channels within each group are cycled through without regard to priority.

Note: Writes to the DMA_CR must be performed only when the DMA channels are inactive (TCDn_CSR[ACTIVE] bits are cleared).

The group priorities operate in a similar fashion. In Group Fixed-Priority Arbitration mode, channel service requests in the highest priority group are executed first where priority level 3 is the highest and priority level 0 is the lowest. The group priorities are assigned in the GRP*n*PRI fields of the DMA Control Register (CR). All group priorities must have unique values prior to the occurrence of any channel service requests; otherwise, a configuration error will be reported. In Group Round-Robin mode, the group priorities are ignored and the groups are cycled through without regard to priority.

Address: 0x0000 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	CX	ECX
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	GRP1PRI(1)		GRP0PRI(2)		EMLM	CLM	HALT	HOE	ERGA	ERCA	EDBG	EBW
W																
Reset	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0

Figure 155. Control Register (DMA_CR)

1. Bit 20 is always set to 0 and not writable.
2. Bit 22 is always set to 0 and not writable.

Table 249. DMA_CR field descriptions

Field	Description
14 CX	Cancel transfer 0 Normal operation 1 Cancel the remaining data transfer. Stop the executing channel and force the minor loop to finish. The cancel takes effect after the last write of the current read/write sequence. The CXFR bit clears itself after the cancel has been honored. This cancel retires the channel normally as if the minor loop was completed
15 ECX	Error cancel transfer 0 Normal operation 1 Cancel the remaining data transfer in the same fashion as the CX bit. Stop the executing channel and force the minor loop to finish. The cancel takes effect after the last write of the current read/write sequence. The ECX bit clears itself after the cancel is honored. In addition to cancelling the transfer, ECX treats the cancel as an error condition; thus updating the ES register and generating an optional error interrupt
20:21 GRP1PRI	Channel group 1 priority Group 1 priority level when fixed-priority group arbitration is enabled.
22:23 GRP0PRI	Channel group 0 priority Group 0 priority level when fixed-priority group arbitration is enabled.
24 EMLM	Enable minor loop mapping 0 Disabled. TCD*n*.word2 is defined as a 32-bit NBYTES field 1 Enabled. TCD*n*.word2 is redefined to include individual enable fields, an offset field, and the NBYTES field. The individual enable fields allow the minor loop offset to be applied to the source address, the destination address, or both. The NBYTES field is reduced when either offset is enabled
25 CLM	Continuous Link mode 0 A minor loop channel link made to itself goes through channel arbitration before eing activated again 1 A minor loop channel link made to itself does not go through channel arbitration before being activated again. Upon minor loop completion, the channel activates again if that channel has a minor loop channel link enabled and the link channel is itself. This effectively applies the minor loop offsets and restarts the next minor loop
26 HALT	Halt DMA operations 0 Normal operation 1 Stall the start of any new channels. Executing channels are allowed to complete. Channel execution resumes when this bit is cleared.
27 HOE	Halt on error 0 Normal operation 1 Any error causes the HALT bit to set. Subsequently, all service requests are ignored until the HALT bit is cleared.
28 ERGA	Enable round-robin group arbitration 0 Fixed-priority arbitration is used for selection among the groups. 1 Round-robin arbitration is used for selection among the groups.
29 ERCA	Enable round-robin channel arbitration 0 Fixed-priority arbitration is used for channel selection. 1 Round-robin arbitration is used for channel selection.

Table 249. DMA_CR field descriptions (continued)

Field	Description
30 EDBG	Enable debug 0 When in Debug mode the DMA continues to operate. 1 When in Debug mode, the DMA stalls the start of a new channel. Executing channels are allowed to complete. Channel execution resumes when the system or the EDBG bit is cleared.
31 EBW	Enable Buffered Writes 0 The bufferable write signal (hprot[2]) is not asserted during AMBA AHB writes. 1 The bufferable write signal (hprot[2]) is asserted on all AMBA AHB writes except for the last write sequence.

20.3.2 Error Status Register (DMA_ES)

The ES provides information concerning the last recorded channel error. Channel errors can be caused by a configuration error (an illegal setting in the transfer-control descriptor or an illegal priority register setting in Fixed Arbitration mode) or an error termination to a bus master read or write cycle.

See *Section 20.4.3*, for more details.

Address: 0x0004 Access: User read-only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	VLD	0	0	0	0	0	0	0	0	0	0	0	0	0	UCE	ECX
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	GPE	CPE	ERRCHN						SAE	SOE	DAE	DOE	NCE	SGE	SBE	DBE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 156. Error Status Register (DMA_ES)

Table 250. DMA_ES field descriptions

Field	Description
0 VLD	ERRL status bits 0 No ERR bits are set 1 At least one ERR bit is set indicating a valid error exists that has not been cleared.
14 UCE	Uncorrectable ECC error 0 No uncorrectable ECC error 1 The last recorded error was an uncorrectable TCD RAM error.
15 ECX	Transfer cancelled 0 No cancelled transfers 1 The last recorded entry was a cancelled transfer by the error cancel transfer input.

Table 250. DMA_ES field descriptions (continued)

Field	Description
16 GPE	Group priority error 0 No group priority error 1 The last recorded error was a configuration error among the group priorities. All group priorities are not unique.
17 CPE	Channel priority error 0 No channel priority error 1 The last recorded error was a configuration error in the channel priorities. Channel priorities are not unique.
18:23 ERRCHN	Error channel number or cancelled channel number GPE and/or last recorded error cancelled transfer.
24 SAE	Source address error 0 No source address configuration error 1 The last recorded error was a configuration error detected in the TCD*n*_SADDR field. TCD*n*_SADDR is inconsistent with TCD*n*_ATTR[SSIZE].
25 SOE	Source offset error 0 No source offset configuration error 1 The last recorded error was a configuration error detected in the TCD*n*_SOFF field. TCD*n*_SOFF is inconsistent with TCD*n*_ATTR[SSIZE].
26 DAE	Destination address error 0 No destination address configuration error 1 The last recorded error was a configuration error detected in the TCD*n*_DADDR field. TCD*n*_DADDR is inconsistent with TCD*n*_ATTR[DSIZE].
27 DOE	Destination offset error 0 No destination offset configuration error 1 The last recorded error was a configuration error detected in the TCD*n*_DOFF field. TCD*n*_DOFF is inconsistent with TCD*n*_ATTR[DSIZE].
28 NCE	NBYTES/CITER configuration error 0 No NBYTES/CITER configuration error 1 The last recorded error was a configuration error detected in the TCD*n*_NBYTES or TCD*n*_CITER fields.
29 SGE	Scatter/gather configuration error 0 No scatter/gather configuration error 1 The last recorded error was a configuration error detected in the TCD*n*_DLASTSGA field. This field is checked at the beginning of a scatter/gather operation after major loop completion if TCD*n*_CSR[ESG] is enabled. TCD*n*_DLASTSGA is not on a 32-byte boundary.
30 SBE	Source bus error 0 No source bus error 1 The last recorded error was a bus error on a source read.
31 DBE	Destination bus error 0 No destination bus error 1 The last recorded error was a bus error on a destination write.

20.3.3 Enable Request Register Low (DMA_ERQL)

The ERQL registers provide a bit map for the 32 implemented channels to enable the request signal for each channel. EQRL covers channels 31–00.

The state of any given channel enable is directly affected by writes to this register; it is also affected by writes to the SERQ and CERQ. The {S,C}ERQ registers are provided so the request enable for a single channel can easily be modified without needing to perform a read-modify-write sequence to the ERQL.

DMA request input signals and this enable request flag must be asserted before a channel's hardware service request is accepted. The state of the DMA enable request flag does not affect a channel service request made explicitly through software or a linked channel request.

Address: 0x000C Access: User read/write

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

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

Figure 157. Enable Request Register Low (DMA_ERQL)

Table 251. DMA_ERQL field descriptions

Field	Description
0 ERQ31	Enable DMA Request 31 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
1 ERQ30	Enable DMA Request 30 0 0The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
2 ERQ29	Enable DMA Request 29 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
3 ERQ28	Enable DMA Request 28 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
4 ERQ27	Enable DMA Request 27 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
5 ERQ26	Enable DMA Request 26 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled

Table 251. DMA_ERQL field descriptions (continued)

Field	Description
6 ERQ25	Enable DMA Request 25 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
7 ERQ24	Enable DMA Request 24 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
8 ERQ23	Enable DMA Request 23 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
9 ERQ22	Enable DMA Request 22 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
10 ERQ21	Enable DMA Request 21 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
11 ERQ20	Enable DMA Request 20 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
12 ERQ19	Enable DMA Request 19 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
13 ERQ18	Enable DMA Request 18 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
14 ERQ17	Enable DMA Request 17 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
15 ERQ16	Enable DMA Request 16 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
16 ERQ15	Enable DMA Request 15 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
17 ERQ14	Enable DMA Request 14 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
18 ERQ13	Enable DMA Request 13 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
19 ERQ12	Enable DMA Request 12 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
20 ERQ11	Enable DMA Request 11 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled

Table 251. DMA_ERQL field descriptions (continued)

Field	Description
21 ERQ10	Enable DMA Request 10 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
22 ERQ9	Enable DMA Request 9 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
23 ERQ8	Enable DMA Request 8 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
24 ERQ7	Enable DMA Request 7 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
25 ERQ6	Enable DMA Request 6 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
26 ERQ5	Enable DMA Request 5 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
27 ERQ4	Enable DMA Request 4 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
28 ERQ3	Enable DMA Request 3 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
29 ERQ2	Enable DMA Request 2 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
30 ERQ1	Enable DMA Request 1 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
31 ERQ0	Enable DMA Request 0 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled

20.3.4 Enable Error Interrupt Register Low (DMA_EEIL)

The Enable Error Interrupt Register Low (DMA_EEIL) provides a bit map for the 32 channels to enable the error interrupt signal for each channel. EEIL covers channels 31–00. The state of any given channel's error interrupt enable is directly affected by writes to this register; it is also affected by writes to the SEEI and CEEI. The {S,C}EEI are provided so the error interrupt enable for a single channel can easily be modified without the need to perform a read-modify-write sequence to the EEIL registers.

The DMA error indicator and the error interrupt enable flag must be asserted before an error interrupt request for a given channel is asserted to the interrupt controller.

Address: 0x0014 Access: User read/write

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

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

Figure 158. Enable Error Interrupt Register Low (DMA_EEIL)

Table 252. DMA_EEIL field descriptions

Field	Description
0 EEI31	Enable error interrupt 31 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
1 EEI30	Enable error interrupt 30 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
2 EEI29	Enable error interrupt 29 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
3 EEI28	Enable error interrupt 28 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
4 EEI27	Enable error interrupt 27 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
5 EEI26	Enable error interrupt 26 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
6 EEI25	Enable error interrupt 25 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
7 EEI24	Enable error interrupt 24 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request

Table 252. DMA_EEIL field descriptions (continued)

Field	Description
8 EEI23	Enable error interrupt 23 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
9 EEI22	Enable error interrupt 22 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
10 EEI21	Enable error interrupt 21 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
11 EEI20	Enable error interrupt 20 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
12 EEI19	Enable error interrupt 19 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
13 EEI18	Enable error interrupt 18 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
14 EEI17	Enable error interrupt 17 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
15 EEI16	Enable error interrupt 16 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
16 EEI15	Enable error interrupt 15 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
17 EEI14	Enable error interrupt 14 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
18 EEI13	Enable error interrupt 13 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request

Table 252. DMA_EEIL field descriptions (continued)

Field	Description
19 EEI12	Enable error interrupt 12 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
20 EEI11	Enable error interrupt 11 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
21 EEI10	Enable error interrupt 10 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
22 EEI9	Enable error interrupt 9 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
23 EEI8	Enable error interrupt 8 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
24 EEI7	Enable error interrupt 7 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
25 EEI6	Enable error interrupt 6 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
26 EEI5	Enable error interrupt 5 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
27 EEI4	Enable error interrupt 4 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
28 EEI3	Enable error interrupt 3 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
29 EEI2	Enable error interrupt 2 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request

Table 252. DMA_EEIL field descriptions (continued)

Field	Description
30 EEI1	Enable error interrupt 1 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
31 EEI0	Enable error interrupt 0 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request

20.3.5 Set Enable Request Register (DMA_SERQ)

The Set Enable Request Register (DMA_SERQ) provides a simple memory-mapped mechanism to set a given bit in the ERQL to enable the DMA request for a given channel. The data value on a register write causes the corresponding bit in the ERQL to be set. Setting the SAER bit provides a global set function, forcing the entire contents of ERQL to be set.

If the NOP bit is set, the command is ignored. This allows you to write multiple-byte registers as a 32-bit word. Reads of this register return all zeros.

Address: 0x0018 Access: User write-only

	0	1	2	3	4	5	6	7
R	0	0	0					
W	NOP	SAER	SERQ					
Reset	0	0	0	0	0	0	0	0

Figure 159. Set Enable Request Register (DMA_SERQ)

Table 253. DMA_SERQ field descriptions

Field	Description
0 NOP	0 Normal operation 1 No operation, ignore bits 1:7 of this register
1 SAER	Set all enable requests 0 Set only those ERQ bits specified in the SERQ field 1 Set all bits in ERQL
2:7 SERQ	Set enable request Sets the corresponding bit in ERQL

20.3.6 Clear Enable Request Register (DMA_CERQ)

The Clear Enable Request Register (DMA_CERQ) provides a simple memory-mapped mechanism to clear a given bit in the ERQL to disable the DMA request for a given channel. The data value on a register write causes the corresponding bit in the ERQL to be cleared. Setting the CAER bit provides a global clear function, forcing the entire contents of the ERQL to be cleared, disabling all DMA request inputs.

If NOP is set, the command is ignored. This allows you to write multiple-byte registers as a 32-bit word. Reads of this register return all zeros.

Address: 0x0019 Access: User write-only

	0	1	2	3	4	5	6	7
R	0	0	0					
W	NOP	CAER	CERQ					
Reset	0	0	0	0	0	0	0	0

Figure 160. Clear Enable Request Register (DMA_CERQ)

Table 254. DMA_CERQ field descriptions

Field	Description
0 NOP	0 Normal operation 1 No operation, ignore bits 1:7 of this register
1 CAER	Clear all enable requests 0 Clear only those ERQ bits specified in the CERQ field 1 Clear all bits in ERQ
2:7 CERQ	Clear enable request Clears the corresponding bit in ERQL

20.3.7 Set Enable Error Interrupt Register (DMA_SEEI)

The Set Enable Error Interrupt Register (DMA_SEEI) provides a simple memory-mapped mechanism to set a given bit in the EEIL to enable the error interrupt for a given channel. The data value on a register write causes the corresponding bit in the EEIL to be set. Setting the SAEE bit provides a global set function, forcing the entire EEIL contents to be set.

If the NOP bit is set, the command is ignored. This allows you to write multiple-byte registers as a 32-bit word. Reads of this register return all zeros.

Address: 0x001A Access: User write-only

	0	1	2	3	4	5	6	7
R	0	0	0					
W	NOP	SAEE	SEEI					
Reset	0	0	0	0	0	0	0	0

Figure 161. Set Enable Error Interrupt Register (DMA_SEEI)

Table 255. DMA_SEEI field descriptions

Field	Description
0 NOP	0 Normal operation 1 No operation, ignore bits 1:7 of this register

Table 255. DMA_SEEI field descriptions (continued)

Field	Description
1 SAEE	Sets all enable error interrupts 0 Set only those EEI bits specified in the SEEI field 1 Sets all bits in EEI
2:7 SEEI	Set enable error interrupt Sets the corresponding bit in EEIL

20.3.8 Clear Enable Error Interrupt Register (DMA_CEEI)

The Clear Enable Error Interrupt Register (DMA_CEEI) provides a simple memory-mapped mechanism to clear a given bit in the EEIL to disable the error interrupt for a given channel. The data value on a register write causes the corresponding bit in the EEIL to be cleared. Setting the CAEE bit provides a global clear function, forcing the EEIL contents to be cleared, disabling all DMA request inputs.

If the NOP bit is set, the command is ignored. This allows you to write multiple-byte registers as a 32-bit word. Reads of this register return all zeros.

Address: 0x001B Access: User write-only

	0	1	2	3	4	5	6	7
R	0	0	0					
W	NOP	CAEE	CEEI					
Reset	0	0	0	0	0	0	0	0

Figure 162. Clear Enable Error Interrupt Register (DMA_CEEI)

Table 256. DMA_CEEI field descriptions

Field	Description
0 NOP	0 Normal operation 1 No operation, ignore bits 1:7 of this register
1 CAEE	Clear all enable error interrupts 0 Clear only those EEI bits specified in the CEEI field 1 Clear all bits in EEI
2:7 CEEI	Clear enable error interrupt Clears the corresponding bit in EEIL

20.3.9 Clear Interrupt Request Register (DMA_CINT)

The Clear Interrupt Request Register (DMA_CINT) provides a simple, memory-mapped mechanism to clear a given bit in the INTL to disable the interrupt request for a given channel. The given value on a register write causes the corresponding bit in the INTL to be cleared. Setting the CAIR bit provides a global clear function, forcing the entire contents of the INTL to be cleared, disabling all DMA interrupt requests.

If the NOP bit is set, the command is ignored. This allows you to write multiple-byte registers as a 32-bit word. Reads of this register return all zeros.

Address: 0x001C Access: User write-only

	0	1	2	3	4	5	6	7
R	0	0	0					
W	NOP	CAIR	CINT					
Reset	0	0	0	0	0	0	0	0

Figure 163. Clear Interrupt Request Register (DMA_CINT)

Table 257. DMA_CINT field descriptions

Field	Description
0 NOP	0 Normal operation 1 No operation, ignore bits 1:7 of this register
1 CAIR	Clear all interrupt requests 0 Clear only those INT bits specified in the CINT field 1 Clear all bits in INTL
2:7 CINT	Clear interrupt request Clears the corresponding bit in INTL

20.3.10 Clear Error Register (DMA_CERR)

The CERR provides a simple memory-mapped mechanism to clear a given bit in the ERRL to disable the error condition flag for a given channel. The given value on a register write causes the corresponding bit in the ERRL to be cleared. Setting the CAEI bit provides a global clear function, forcing the ERRL contents to be cleared, clearing all channel error indicators.

If the NOP bit is set, the command is ignored. This allows you to write multiple-byte registers as a 32-bit word. Reads of this register return all zeros.

Address: 0x001D Access: User write-only

	0	1	2	3	4	5	6	7
R	0	0	0					
W	NOP	CAEI	CERR					
Reset	0	0	0	0	0	0	0	0

Figure 164. Clear Error Register (DMA_CERR)

Table 258. DMA_CERR field descriptions

Field	Description
0 NOP	0 Normal operation 1 No operation, ignore bits 1:7 of this register
1 CAEI	Clear all error indicators 0 Clear only those ERR bits specified in the CERR field 1 Clear all bits in ERR
2:7 CERR	Clear error indicator Clears the corresponding bit in ERRL

20.3.11 Set START Bit Register (DMA_SSRT)

The Set START Bit Register (DMA_SSRT) provides a simple memory-mapped mechanism to set the START bit in the TCD of the given channel. The data value on a register write causes the START bit in the corresponding transfer control descriptor to be set. Setting the SAST bit provides a global set function, forcing all START bits to be set.

If the NOP bit is set, the command is ignored. This allows you to write multiple-byte registers as a 32-bit word. Reads of this register return all zeros.

Address: 0x001E Access: User write-only

	0	1	2	3	4	5	6	7
R	0	0	0					
W	NOP	SAST	SSRT					
Reset	0	0	0	0	0	0	0	0

Figure 165. Set START Bit Register (DMA_SSRT)

Table 259. DMA_SSRT field descriptions

Field	Description
0 NOP	0 Normal operation 1 No operation, ignore bits 1:7 of this register
1 SAST	Set all START bits (activates all channels) 0 Set only those TCD*n*_CSR[START] bits specified in the SSRT field 1 Set all bits in TCD*n*_CSR[START]
2:7 SSRT	Set START bit Sets the corresponding bit in TCD*n*_CSR[START].

20.3.12 Clear DONE Status Bit Register (DMA_CDNE)

The Clear DONE Status Bit Register (DMA_CDNE) provides a simple memory-mapped mechanism to clear the DONE bit in the TCD of the given channel. The data value on a register write causes the DONE bit in the corresponding transfer control descriptor to be cleared. Setting the CADN bit provides a global clear function, forcing all DONE bits to be cleared.

If the NOP bit is set, the command is ignored. This allows you to write multiple-byte registers as a 32-bit word. Reads of this register return all zeros.

Address: 0x001F Access: User write-only

	0	1	2	3	4	5	6	7
R	0	0	0					
W	NOP	CADN	CDNE					
Reset	0	0	0	0	0	0	0	0

Figure 166. Clear DONE Status Bit Register (DMA_CDNE)

Table 260. DMA_CDNE field descriptions

Field	Description
0 NOP	0 Normal operation 1 No operation, ignore bits 1:7 of this register
1 CADN	Clears all DONE bits 0 Clears only those TCD*n*_CSR[DONE] bits specified in the CDNE field 1 Clears all bits in TCD*n*_CSR[DONE]
2:7 CDNE	Clear DONE bit Clears the corresponding bit in TCD*n*_CSR[DONE]

20.3.13 Interrupt Request Register Low (DMA_INTL)

The Interrupt Request Register Low (DMA_INTL) provides a bit map for the lower half of the 32 channels signaling the presence of an interrupt request for each channel. Depending on the appropriate bit setting in the transfer-control descriptors, the eDMA engine generates an interrupt at data transfer completion. The outputs of this register are directly routed to the interrupt controller (INTC). During the interrupt-service routine associated with any given channel, it is the software's responsibility to clear the appropriate bit, negating the interrupt request. Typically, a write to the CINT in the interrupt service routine is used for this purpose.

The state of any given channel's interrupt request is directly affected by writes to this register; it is also affected by writes to the CINT. On writes to the INT, a '1' in any bit position clears the corresponding channel's interrupt request. A zero in any bit position has no effect on the corresponding channel's current interrupt status. The CINT is provided so the interrupt request for a single channel can easily be cleared without the need to perform a read-modify-write sequence to the INTL.

Address: 0x0024 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	INT 31	INT 30	INT 29	INT 28	INT 27	INT 26	INT 25	INT 24	INT 23	INT 22	INT 21	INT 20	INT 19	INT 18	INT 17	INT 16
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	INT 15	INT 14	INT 13	INT 12	INT 11	INT 10	INT 9	INT 8	INT 7	INT 6	INT 5	INT 4	INT 3	INT 2	INT 1	INT 0
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 167. Interrupt Request Register Low (DMA_INTL)

Table 261. DMA_INTL field descriptions

Field	Description
0 INT31	Interrupt request 31 0 The interrupt request for corresponding channel is cleared. 1 The interrupt request for corresponding channel is active.
1 INT30	Interrupt request 30 0 The interrupt request for corresponding channel is cleared. 1 The interrupt request for corresponding channel is active.
2 INT29	Interrupt request 29 0 The interrupt request for corresponding channel is cleared. 1 The interrupt request for corresponding channel is active.
3 INT28	Interrupt request 28 0 The interrupt request for corresponding channel is cleared. 1 The interrupt request for corresponding channel is active.
4 INT27	Interrupt request 27 0 The interrupt request for corresponding channel is cleared. 1 The interrupt request for corresponding channel is active.
5 INT26	Interrupt request 26 0 The interrupt request for corresponding channel is cleared. 1 The interrupt request for corresponding channel is active.
6 INT25	Interrupt request 25 0 The interrupt request for corresponding channel is cleared. 1 The interrupt request for corresponding channel is active.
7 INT24	Interrupt request 24 0 The interrupt request for corresponding channel is cleared. 1 The interrupt request for corresponding channel is active.
8 INT23	Interrupt request 23 0 The interrupt request for corresponding channel is cleared. 1 The interrupt request for corresponding channel is active.
9 INT22	Interrupt request 22 0 The interrupt request for corresponding channel is cleared. 1 The interrupt request for corresponding channel is active.
10 INT21	Interrupt request 21 0 The interrupt request for corresponding channel is cleared. 1 The interrupt request for corresponding channel is active.
11 INT20	Interrupt request 20 0 The interrupt request for corresponding channel is cleared. 1 The interrupt request for corresponding channel is active.
12 INT19	Interrupt request 19 0 The interrupt request for corresponding channel is cleared. 1 The interrupt request for corresponding channel is active.
13 INT18	Interrupt request 18 0 The interrupt request for corresponding channel is cleared. 1 The interrupt request for corresponding channel is active.
14 INT17	Interrupt request 17 0 The interrupt request for corresponding channel is cleared. 1 The interrupt request for corresponding channel is active.

Table 261. DMA_INTL field descriptions (continued)

Field	Description
15 INT16	Interrupt request 16 0 The interrupt request for corresponding channel is cleared. 1 The interrupt request for corresponding channel is active.
16 INT15	Interrupt request 15 0 The interrupt request for corresponding channel is cleared. 1 The interrupt request for corresponding channel is active.
17 INT14	Interrupt request 14 0 The interrupt request for corresponding channel is cleared. 1 The interrupt request for corresponding channel is active.
18 INT13	Interrupt request 13 0 The interrupt request for corresponding channel is cleared. 1 The interrupt request for corresponding channel is active.
19 INT12	Interrupt request 12 0 The interrupt request for corresponding channel is cleared. 1 The interrupt request for corresponding channel is active.
20 INT11	Interrupt request 11 0 The interrupt request for corresponding channel is cleared. 1 The interrupt request for corresponding channel is active.
21 INT10	Interrupt request 10 0 The interrupt request for corresponding channel is cleared. 1 The interrupt request for corresponding channel is active.
22 INT9	Interrupt request 9 0 The interrupt request for corresponding channel is cleared. 1 The interrupt request for corresponding channel is active.
23 INT8	Interrupt request 8 0 The interrupt request for corresponding channel is cleared. 1 The interrupt request for corresponding channel is active.
24 INT7	Interrupt request 7 0 The interrupt request for corresponding channel is cleared. 1 The interrupt request for corresponding channel is active.
25 INT6	Interrupt request 6 0 The interrupt request for corresponding channel is cleared. 1 The interrupt request for corresponding channel is active.
26 INT5	Interrupt request 5 0 The interrupt request for corresponding channel is cleared. 1 The interrupt request for corresponding channel is active.
27 INT4	Interrupt request 4 0 The interrupt request for corresponding channel is cleared. 1 The interrupt request for corresponding channel is active.
28 INT3	Interrupt request 3 0 The interrupt request for corresponding channel is cleared. 1 The interrupt request for corresponding channel is active.
29 INT2	Interrupt request 2 0 The interrupt request for corresponding channel is cleared. 1 The interrupt request for corresponding channel is active.

Table 261. DMA_INTL field descriptions (continued)

Field	Description
30 INT1	Interrupt request 1 0 The interrupt request for corresponding channel is cleared. 1 The interrupt request for corresponding channel is active.
31 INT0	Interrupt request 0 0 The interrupt request for corresponding channel is cleared. 1 The interrupt request for corresponding channel is active.

20.3.14 Error Register Low (DMA_ERRL)

The ERRL provides a bit map for the 32 channels, signaling the presence of an error for each channel. ERRL covers channels 31–00. The eDMA engine signals the occurrence of an error condition by setting the appropriate bit in this register. The outputs of this register are enabled by the contents of the EEI, then logically summed across groups of 16 and 32 channels to form several group error interrupt requests that are then routed to the interrupt controller. During the execution of the interrupt-service routine associated with any DMA errors, it is software's responsibility to clear the appropriate bit, negating the error-interrupt request. Typically, a write to the CERR in the interrupt-service routine is used for this purpose. The normal DMA channel completion indicators (setting the transfer control descriptor DONE flag and the possible assertion of an interrupt request) are not affected when an error is detected.

The contents of this register can also be polled because a non-zero value indicates the presence of a channel error regardless of the state of the EEI. The state of any given channel's error indicators is affected by writes to this register; it is also affected by writes to the CERR. On writes to the ERR, a one in any bit position clears the corresponding channel's error status. A zero in any bit position has no effect on the corresponding channel's current error status. The CERR is provided so the error indicator for a single channel can easily be cleared.

Address: 0x002C Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	ERR 31	ERR 30	ERR 29	ERR 28	ERR 27	ERR 26	ERR 25	ERR 24	ERR 23	ERR 22	ERR 21	ERR 20	ERR 19	ERR 18	ERR 17	ERR 16
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	ERR 15	ERR 14	ERR 13	ERR 12	ERR 11	ERR 10	ERR 9	ERR 8	ERR 7	ERR 6	ERR 5	ERR 4	ERR 3	ERR 2	ERR 1	ERR 0
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 168. Error Register Low (DMA_ERRL)

Table 262. DMA_ERRL field descriptions

Field	Description
0 ERR31	Error in channel 31 0 An error in the corresponding channel has not occurred. 1 An error in the corresponding channel has occurred.
1 ERR30	Error in channel 30 0 An error in the corresponding channel has not occurred. 1 An error in the corresponding channel has occurred.
2 ERR29	Error in channel 29 0 An error in the corresponding channel has not occurred. 1 An error in the corresponding channel has occurred.
3 ERR28	Error in channel 28 0 An error in the corresponding channel has not occurred. 1 An error in the corresponding channel has occurred.
4 ERR27	Error in channel 27 0 An error in the corresponding channel has not occurred. 1 An error in the corresponding channel has occurred.
5 ERR26	Error in channel 26 0 An error in the corresponding channel has not occurred. 1 An error in the corresponding channel has occurred.
6 ERR25	Error in channel 25 0 An error in the corresponding channel has not occurred. 1 An error in the corresponding channel has occurred.
7 ERR24	Error in channel 24 0 An error in the corresponding channel has not occurred. 1 An error in the corresponding channel has occurred.
8 ERR23	Error in channel 23 0 An error in the corresponding channel has not occurred. 1 An error in the corresponding channel has occurred.
9 ERR22	Error in channel 22 0 An error in the corresponding channel has not occurred. 1 An error in the corresponding channel has occurred.
10 ERR21	Error in channel 21 0 An error in the corresponding channel has not occurred. 1 An error in the corresponding channel has occurred.
11 ERR20	Error in channel 20 0 An error in the corresponding channel has not occurred. 1 An error in the corresponding channel has occurred.
12 ERR19	Error in channel 19 0 An error in the corresponding channel has not occurred. 1 An error in the corresponding channel has occurred.
13 ERR18	Error in channel 18 0 An error in the corresponding channel has not occurred. 1 An error in the corresponding channel has occurred.
14 ERR17	Error in channel 17 0 An error in the corresponding channel has not occurred. 1 An error in the corresponding channel has occurred.

Table 262. DMA_ERRL field descriptions (continued)

Field	Description
15 ERR16	Error in channel 16 0 An error in the corresponding channel has not occurred. 1 An error in the corresponding channel has occurred.
16 ERR15	Error in channel 15 0 An error in the corresponding channel has not occurred. 1 An error in the corresponding channel has occurred.
17 ERR14	Error in channel 14 0 An error in the corresponding channel has not occurred. 1 An error in the corresponding channel has occurred.
18 ERR13	Error in channel 13 0 An error in the corresponding channel has not occurred. 1 An error in the corresponding channel has occurred.
19 ERR12	Error in channel 12 0 An error in the corresponding channel has not occurred. 1 An error in the corresponding channel has occurred.
20 ERR11	Error in channel 11 0 An error in the corresponding channel has not occurred. 1 An error in the corresponding channel has occurred.
21 ERR10	Error in channel 10 0 An error in the corresponding channel has not occurred. 1 An error in the corresponding channel has occurred.
22 ERR9	Error in channel 9 0 An error in the corresponding channel has not occurred. 1 An error in the corresponding channel has occurred.
23 ERR8	Error in channel 8 0 An error in the corresponding channel has not occurred. 1 An error in the corresponding channel has occurred.
24 ERR7	Error in channel 7 0 An error in the corresponding channel has not occurred. 1 An error in the corresponding channel has occurred.
25 ERR6	Error in channel 6 0 An error in the corresponding channel has not occurred. 1 An error in the corresponding channel has occurred.
26 ERR5	Error in channel 5 0 An error in the corresponding channel has not occurred. 1 An error in the corresponding channel has occurred.
27 ERR4	Error in channel 4 0 An error in the corresponding channel has not occurred. 1 An error in the corresponding channel has occurred.
28 ERR3	Error in channel 3 0 An error in the corresponding channel has not occurred. 1 An error in the corresponding channel has occurred.
29 ERR2	Error in channel 2 0 An error in the corresponding channel has not occurred. 1 An error in the corresponding channel has occurred.

Table 262. DMA_ERRL field descriptions (continued)

Field	Description
30 ERR1	Error in channel 1 0 An error in the corresponding channel has not occurred. 1 An error in the corresponding channel has occurred.
31 ERR0	Error in channel 0 0 An error in the corresponding channel has not occurred. 1 An error in the corresponding channel has occurred.

20.3.15 Hardware Request Status Register Low (DMA_HRSL)

The HRSL provides a bit map for the DMA channels, signaling the presence of a qualified hardware service request for each channel. HRSL covers channels 31–00. The Hardware Request Status bits reflect the current state of the register and qualified (via the ERQL fields) DMA request signals as seen by the DMA's arbitration logic. This view into the hardware request signals may be used for debug purposes.

Note: These bits reflect the state of the request as seen by the arbitration logic. Therefore, this status is affected by the ERQL bits.

Address: 0x0034 Access: User read/write

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

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

Figure 169. Hardware Request Status Register Low (DMA_HRSL)

Table 263. DMA_HRSL field descriptions

Field	Description
0 HRS31	Hardware request status channel 31 0 A hardware service request for the corresponding channel is not present. 1 A hardware service request for the corresponding channel is present.
1 HRS30	Hardware request status channel 30 0 A hardware service request for the corresponding channel is not present. 1 A hardware service request for the corresponding channel is present.
2 HRS29	Hardware request status channel 29 0 A hardware service request for the corresponding channel is not present. 1 A hardware service request for the corresponding channel is present.
3 HRS28	Hardware request status channel 28 0 A hardware service request for the corresponding channel is not present. 1 A hardware service request for the corresponding channel is present.

Table 263. DMA_HRSL field descriptions (continued)

Field	Description
4 HRS27	Hardware request status channel 27 0 A hardware service request for the corresponding channel is not present. 1 A hardware service request for the corresponding channel is present.
5 HRS26	Hardware request status channel 26 0 A hardware service request for the corresponding channel is not present. 1 A hardware service request for the corresponding channel is present.
6 HRS25	Hardware request status channel 25 0 A hardware service request for the corresponding channel is not present. 1 A hardware service request for the corresponding channel is present.
7 HRS24	Hardware request status channel 24 0 A hardware service request for the corresponding channel is not present. 1 A hardware service request for the corresponding channel is present.
8 HRS23	Hardware request status channel 23 0 A hardware service request for the corresponding channel is not present. 1 A hardware service request for the corresponding channel is present.
9 HRS22	Hardware request status channel 22 0 A hardware service request for the corresponding channel is not present. 1 A hardware service request for the corresponding channel is present.
10 HRS21	Hardware request status channel 21 0 A hardware service request for the corresponding channel is not present. 1 A hardware service request for the corresponding channel is present.
11 HRS20	Hardware request status channel 20 0 A hardware service request for the corresponding channel is not present. 1 A hardware service request for the corresponding channel is present.
12 HRS19	Hardware request status channel 19 0 A hardware service request for the corresponding channel is not present. 1 A hardware service request for the corresponding channel is present.
13 HRS18	Hardware request status channel 18 0 A hardware service request for the corresponding channel is not present. 1 A hardware service request for the corresponding channel is present.
14 HRS17	Hardware request status channel 17 0 A hardware service request for the corresponding channel is not present. 1 A hardware service request for the corresponding channel is present.
15 HRS16	Hardware request status channel 16 0 A hardware service request for the corresponding channel is not present. 1 A hardware service request for the corresponding channel is present..
16 HRS15	Hardware request status channel 15 0 A hardware service request for the corresponding channel is not present. 1 A hardware service request for the corresponding channel is present.
17 HRS14	Hardware request status channel 14 0 A hardware service request for the corresponding channel is not present. 1 A hardware service request for the corresponding channel is present.
18 HRS13	Hardware request status channel 13 0 A hardware service request for the corresponding channel is not present. 1 A hardware service request for the corresponding channel is present.

Table 263. DMA_HRSL field descriptions (continued)

Field	Description
19 HRS12	Hardware request status channel 12 0 A hardware service request for the corresponding channel is not present. 1 A hardware service request for the corresponding channel is present.
20 HRS11	Hardware request status channel 11 0 A hardware service request for the corresponding channel is not present. 1 A hardware service request for the corresponding channel is present.
21 HRS10	Hardware request status channel 10 0 A hardware service request for the corresponding channel is not present. 1 A hardware service request for the corresponding channel is present.
22 HRS9	Hardware request status channel 9 0 A hardware service request for the corresponding channel is not present. 1 A hardware service request for the corresponding channel is present.
23 HRS8	Hardware request status channel 8 0 A hardware service request for the corresponding channel is not present. 1 A hardware service request for the corresponding channel is present.
24 HRS7	Hardware request status channel 7 0 A hardware service request for the corresponding channel is not present. 1 A hardware service request for the corresponding channel is present.
25 HRS6	Hardware request status channel 6 0 A hardware service request for the corresponding channel is not present. 1 A hardware service request for the corresponding channel is present.
26 HRS5	Hardware request status channel 5 0 A hardware service request for the corresponding channel is not present. 1 A hardware service request for the corresponding channel is present.
27 HRS4	Hardware request status channel 4 0 A hardware service request for the corresponding channel is not present. 1 A hardware service request for the corresponding channel is present.
28 HRS3	Hardware request status channel 3 0 A hardware service request for the corresponding channel is not present. 1 A hardware service request for the corresponding channel is present.
29 HRS2	Hardware request status channel 2 0 A hardware service request for the corresponding channel is not present.. 1 A hardware service request for the corresponding channel is present
30 HRS1	Hardware request status channel 1 0 A hardware service request for the corresponding channel is not present. 1 A hardware service request for the corresponding channel is present.
31 HRS0	Hardware request status channel 0 0 A hardware service request for the corresponding channel is not present. 1 A hardware service request for the corresponding channel is present.

20.3.16 Channel *n* Priority Register (DMA_DCHPRI*n*)

When the Fixed-Priority Channel Arbitration mode is enabled (CR[ERCA] = 0), the contents of these registers define the unique priorities associated with each channel within a group. The channel priorities are evaluated by numeric value; for example, 0 is the lowest priority, 1 is the next higher priority, then 2, 3, and so on. Software must program the channel priorities

with unique values. Otherwise, a configuration error is reported. The range of the priority value is limited to the values of 0 through 15.

When read, the GRPPRI bits of the DCHPRI*n* register reflect the current priority level of the group of channels in which the corresponding channel resides. GRPPRI bits are not affected by writes to the DCHPRI*n* registers. The group priority is assigned in the DMA control register.

Address: 0x0100 + *n**0x1 (*n* = 0 to 31) Access: User read/write

	0	1	2	3	4	5	6	7
R	ECP	DPA	GRPPRI		CHPRI			
W								
RESET(1)	0	0	X	X	X	X	X	X

1. x = Undefined at reset.

Figure 170. Channel *n* Priority Register (DMA_DCHPRI*n*)

Table 264. DMA_DCHPRI*n* field descriptions

Field	Description
0 ECP	Enable channel preemption This bit resets to zero. 0 Channel *n* cannot be suspended by a higher priority channel's service request 1 Channel *n* can be temporarily suspended by the service request of a higher priority channel
1 DPA	Disable preempt ability This bit resets to zero. 0 Channel *n* can suspend a lower priority channel. 1 Channel *n* cannot suspend any channel, regardless of channel priority.
2:3 GRPPRI	Channel *n* current group priority Group priority assigned to this channel group when fixed-priority arbitration is enabled. These two bits are read only; writes are ignored. **Note:** Reset value for the group and channel priority fields, GRPPRI and CHPRI, is equal to the corresponding channel number for each priority register, that is DCHPRI31[GRPPRI] = 0b01 and DCHPRI31[CHPRI] equals 0b1111.
4:7 CHPRI	Channel *n* arbitration priority Channel priority when fixed-priority arbitration is enabled. **Note:** Reset value for the group and channel priority fields, GRPPRI and CHPRI, is equal to the corresponding channel number for each priority register, that is DCHPRI31[GRPPRI] = 0b01 and DCHPRI31[CHPRI] equals 0b1111.

20.3.17 Channel *n* Master ID Register (DMA_DCHMID*n*)

The DMA Master ID Replication registers allow the DMA to use the same protection level and AHB system bus ID of the master programming the DMA's TCD. When enabled, the DMA uses the master ID and protection level stored in the DCHMID register instead of the DMA's default values. When a master (for example a core) programs a TCD, its master ID and protection level are captured when the TCD Word 7 control attributes are written. Although the scatter/gather operation can change the contents of TCD Word 7, that operation does not affect the DCHMID*n* registers.

Address: 0x0140 + *n**0x1 (*n* = 0 to 31) Access: User read/write

	0	1	2	3	4	5	6	7
R	EMI	PAL	0	0	MID			
W								
Reset	0	0	0	0	0	0	0	0

Figure 171. Channel *n* Master ID Register (DMA_DCHMID*n*)

Table 265. DMA_DCHMID*n* field descriptions

Field	Description
0 EMI	Enable Master ID replication 0 Master ID replication is disabled. 1 Master ID replication is enabled.
1 PAL	Privileged Access Level The protection level captured in this register reflects the level used when writing the channel's control attributes; lower byte of TCD Word 7. 0 User protection level for DMA transfers 1 Privileged protection level for DMA transfers
4:7MID	Master ID DMA's master ID when channel *n* is active and master ID replication is enabled. **Note:** The master ID captured in this register reflects the ID used when writing the channel's control attributes; lower byte of TCD Word 7.

20.3.18 TCD Source Address (DMA_TCD*n*_SADDR)

See TCD Source Address register figure and DMA_TCD*n*_SADDR field descriptions table as follows.

Address: 0x1000 + *n**0x20 (*n* = 0 to 31) Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	SADDR															
W																
Reset(1)	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	SADDR															
W																
Reset(1)	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x

1. x = Undefined at reset.

Figure 172. TCD Source Address (DMA_TCD*n*_SADDR)

Table 266. DMA_TCD*n*_SADDR field descriptions

Field	Description
0:31 SADDR	Source address Memory address pointing to the source data.

20.3.19 TCD Transfer Attributes (DMA_TCD*n*_ATTR)

See TCD Transfer Attributes register figure and DMA_TCD*n*_ATTR field descriptions table as follows.

Address: 0x1004 + *n**0x20 (*n* = 0 to 31) Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	SMOD					SSIZE			DMOD				DSIZE			
Reset(1)	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x

1. x = Undefined at reset.

Figure 173. TCD Transfer Attributes (DMA_TCD*n*_ATTR)

Table 267. DMA_TCD*n*_ATTR field descriptions

Field	Description
0:4 SMOD	Source address modulo 0x00 Source address modulo feature is disabled ≠0 This value defines a specific address range specified to be the value after SADDR + SOFF calculation is performed on the original register value. Setting this field provides the ability to implement a circular data queue easily. For data queues requiring power-of-2 size bytes, the queue should start at a 0-modulo-size address and the SMOD field should be set to the appropriate value for the queue, freezing the desired number of upper address bits. The value programmed into this field specifies the number of lower address bits allowed to change. For a circular queue application, the SOFF is typically set to the transfer size to implement post-increment addressing with the SMOD function constraining the addresses to a 0-modulo-size range
5:7 SSIZE	Source data transfer size Using a reserved encoding causes a configuration error. 000 8-bit 001 16-bit 010 32-bit 011 64-bit 100 Reserved 101 32-byte 110 Reserved 111 Reserved
8:12 DMOD	Destination address modulo See the SMOD definition.
13:15 DSIZE	Destination data transfer size See the SSIZE definition.

20.3.20 TCD Signed Source Address Offset (DMA_TCD*n*_SOFF)

See TCD Signed Source Address Offset register figure and DMA_TCD*n*_SOFF field descriptions table as follows.

Address: 0x1006 + *n**0x20 (*n* = 0 to 31) Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	SOFF															
Reset(1)	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x

1. x = Undefined at reset.

Figure 174. TCD Signed Source Address Offset (DMA_TCD*n*_SOFF)

Table 268. DMA_TCD*n*_SOFF field descriptions

Field	Description
0:15 SOFF	Source address signed offset Sign-extended offset applied to the current source address to form the next-state value as each source read is completed.

20.3.21 TCD Minor Byte Count (Minor Loop Disabled) (DMA_TCD*n*_NBYTES_MLNO)

See TCD Minor Byte Count (Minor Loop Disabled) register figure and DMA_TCD*n*_NBYTES_MLNO field descriptions table below.

If minor loop mapping is disabled (CR[EMLM] = 0), TCD word 2 is defined as follows.

Address: 0x1008 + *n**0x20 (*n* = 0 to 31) Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	NBYTES															
Reset(1)	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	NBYTES															
Reset(1)	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x

1. x = Undefined at reset.

Figure 175. TCD Minor Byte Count (Minor Loop Disabled) (DMA_TCD*n*_NBYTES_MLNO)

Table 269. DMA_TCD*n*_NBYTES_MLNO field descriptions

Field	Description
0:31 NBYTES	Minor byte transfer count Number of bytes to be transferred in each service request of the channel. As a channel activates, the appropriate TCD contents load into the eDMA engine, and the appropriate reads and writes perform until the minor byte transfer count has transferred. This is an indivisible operation and cannot be halted. (Although, it may be stalled by using the bandwidth control field, or via preemption.) After the minor count is exhausted, the SADDR and DADDR values are written back into the TCD memory, the major iteration count is decremented and restored to the TCD memory. If the major iteration count is completed, additional processing is performed. **Note:** An NBYTES value of 0x0000_0000 is interpreted as a 4 GB transfer.

20.3.22 TCD Signed Minor Loop Offset (Minor Loop Enabled and Offset Disabled) (DMA_TCD*n*_NBYTES_MLOFFNO)

TCD Word 2 is defined as follows if Minor loop mapping is enabled (CR[EMLM] = 1) and SMLOE = 0 and DMLOE = 0

Address: 0x1008 + *n**0x20 (*n* = 0 to 31) Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	SMLOE	DMLOE	NBYTES													
W																
Reset(1)	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	NBYTES															
W																
Reset(1)	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x

1. x = Undefined at reset.

Figure 176. TCD Signed Minor Loop Offset (Minor Loop Enabled and Offset Disabled) (DMA_TCD*n*_NBYTES_MLOFFNO)

Table 270. DMA_TCD*n*_NBYTES_MLOFFNO field descriptions

Field	Description
0 SMLOE	Source minor loop offset enable Selects whether the minor loop offset is applied to the source address upon minor loop completion. 0 The minor loop offset is not applied to the SADDR. 1 The minor loop offset is applied to the SADDR.

Table 270. DMA_TCD*n*_NBYTES_MLOFFNO field descriptions (continued)

Field	Description
1 DMLOE	Destination minor loop offset enable Selects whether the minor loop offset is applied to the destination address upon minor loop completion. 0 The minor loop offset is not applied to the DADDR. 1 The minor loop offset is applied to the DADDR.
2:31 NBYTES	Minor byte transfer count Number of bytes to be transferred in each service request of the channel. As a channel activates, the appropriate TCD contents load into the eDMA engine, and the appropriate reads and writes perform until the minor byte transfer count has transferred. This is an indivisible operation and cannot be halted. (Although, it may be stalled by using the bandwidth control field, or via preemption.) After the minor count is exhausted, the SADDR and DADDR values are written back into the TCD memory, the major iteration count is decremented and restored to the TCD memory. If the major iteration count is completed, additional processing is performed.

20.3.23 TCD Signed Minor Loop Offset (Minor Loop and Offset Enabled) (DMA_TCD*n*_NBYTES_MLOFFYES)

TCD Word 2 is defined as follows if Minor loop mapping is enabled (CR[EMLM] = 1) and Minor loop offset enabled (SMLOE or DMLOE = 1)

Address: 0x1008 + *n**0x20 (*n* = 0 to 31) Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	SMLOE	DMLOE	MLOFF													
Reset(1)	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	MLOFF						NBYTES									
Reset(1)	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x

1. x = Undefined at reset.

Figure 177. TCD Signed Minor Loop Offset (Minor Loop and Offset Enabled) (DMA_TCD*n*_NBYTES_MLOFFYES)

Table 271. DMA_TCD*n*_NBYTES_MLOFFYES field descriptions

Field	Description
0 SMLOE	Source minor loop offset enable Selects whether the minor loop offset is applied to the source address upon minor loop completion. 0 The minor loop offset is not applied to the SADDR. 1 The minor loop offset is applied to the SADDR.
1 DMLOE	Destination minor loop offset enable Selects whether the minor loop offset is applied to the destination address upon minor loop completion. 0 The minor loop offset is not applied to the DADDR. 1 The minor loop offset is applied to the DADDR.
2:21 MLOFF	If SMLOE or DMLOE is set, this field represents a sign-extended offset applied to the source or destination address to form the next-state value after the minor loop completes.
22:31 NBYTES	Minor byte transfer count Number of bytes to be transferred in each service request of the channel. As a channel activates, the appropriate TCD contents load into the eDMA engine, and the appropriate reads and writes perform until the minor byte transfer count has transferred. This is an indivisible operation and cannot be halted. (Although, it may be stalled by using the bandwidth control field, or via preemption.) After the minor count is exhausted, the SADDR and DADDR values are written back into the TCD memory, the major iteration count is decremented and restored to the TCD memory. If the major iteration count is completed, additional processing is performed.

20.3.24 TCD Last Source Address Adjustment (DMA_TCD*n*_SLAST)

Address: 0x100C + *n**0x20 (*n* = 0 to 31) Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	SLAST															
Reset(1)	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	SLAST															
Reset(1)	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x

1. x = Undefined at reset.

Figure 178. TCD Last Source Address Adjustment (DMA_TCD*n*_SLAST)

Table 272. DMA_TCD*n*_SLAST field descriptions

Field	Description
0:31 SLAST	Last source address adjustment Adjustment value added to the source address at the completion of the major iteration count. This value can be applied to restore the source address to the initial value, or adjust the address to reference the next data structure.

20.3.25 TCD Destination Address (DMA_TCD*n*_DADDR)

See TCD Destination Address register figure and DMA_TCD*n*_DADDR field descriptions table as follows.

Address: 0x1010 + *n**0x20 (n = 0 to 31) Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	DADDR															
Reset(1)	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	DADDR															
Reset(1)	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x

1. x = Undefined at reset.

Figure 179. TCD Destination Address (DMA_TCD*n*_DADDR)

Table 273. DMA_TCD*n*_DADDR field descriptions

Field	Description
0:31 DADDR	Destination address Memory address pointing to the destination data.

20.3.26 TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD*n*_CITER_ELINKYES)

See TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) register figure and DMA_TCDn_CITER_ELINKYES field descriptions table as follows.

Address: 0x1014 + *n**0x20 (*n* = 0 to 31) Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	ELINK	LINKCH							CITER							
Reset(1)	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x

1. x = Undefined at reset.

Figure 180. TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD*n*_CITER_ELINKYES)

Table 274. DMA_TCD*n*_CITER_ELINKYES field descriptions

Field	Description
0 ELINK	Enable channel-to-channel linking on minor-loop complete As the channel completes the minor loop, this flag enables linking to another channel, defined by the LINKCH field. The link target channel initiates a channel service request via an internal mechanism that sets the TCD*n*_CSR[START] bit of the specified channel. If channel linking is disabled, the CITER value is extended to 15 bits in place of a link channel number. If the major loop is exhausted, this link mechanism is suppressed in favor of the MAJORELINK channel linking. **Note:** This bit must be equal to the BITER[ELINK] bit. Otherwise, a configuration error is reported. 0 The channel-to-channel linking is disabled. 1 The channel-to-channel linking is enabled.
1:6 LINKCH	Link channel number If channel-to-channel linking is enabled (ELINK = 1), then after the minor loop is exhausted, the eDMA engine initiates a channel service request to the channel defined by these six bits by setting that channel's TCD*n*_CSR[START] bit.
7:15 CITER	Current major iteration count This 9-bit (ELINK = 1) or 15-bit (ELINK = 0) count represents the current major loop count for the channel. It is decremented each time the minor loop is completed and updated in the transfer control descriptor memory. After the major iteration count is exhausted, the channel performs a number of operations (for example final source and destination address calculations), optionally generating an interrupt to signal channel completion before reloading the CITER field from the beginning iteration count (BITER) field. **Note:** When the CITER field is initially loaded by software, it must be set to the same value as that contained in the BITER field. If the channel is configured to execute a single service request, the initial values of BITER and CITER should be 0x0001.

20.3.27 TCD Current Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD*n*_CITER_ELINKNO)

See TCD Current Minor Loop Link, Major Loop Count (Channel Linking Disabled) register figure and DMA_TCDn_CITER_ELINKNO field descriptions table as follows.

Address: 0x1014 + *n**0x20 (*n* = 0 to 31) Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	ELINK	CITER														
Reset(1)	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x

1. x = Undefined at reset.

Figure 181. TCD Current Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD*n*_CITER_ELINKNO)

Table 275. DMA_TCD*n*_CITER_ELINKNO field descriptions

Field	Description
0 ELINK	Enable channel-to-channel linking on minor-loop complete As the channel completes the minor loop, this flag enables linking to another channel, defined by the LINKCH field. The link target channel initiates a channel service request via an internal mechanism that sets the TCD*n*_CSR[START] bit of the specified channel.If channel linking is disabled, the CITER value is extended to 15 bits in place of a link channel number. If the major loop is exhausted, this link mechanism is suppressed in favor of the MAJORELINK channel linking. **Note:** This bit must be equal to the BITER[ELINK] bit. Otherwise, a configuration error is reported. 0 The channel-to-channel linking is disabled. 1 The channel-to-channel linking is enabled.
1:15 CITER	Current major iteration count This 9-bit (ELINK = 1) or 15-bit (ELINK = 0) count represents the current major loop count for the channel. It is decremented each time the minor loop is completed and updated in the transfer control descriptor memory. After the major iteration count is exhausted, the channel performs a number of operations (for example, final source and destination address calculations), optionally generating an interrupt to signal channel completion before reloading the CITER field from the beginning iteration count (BITER) field. **Note:** When the CITER field is initially loaded by software, it must be set to the same value as that contained in the BITER field. If the channel is configured to execute a single service request, the initial values of BITER and CITER should be 0x0001.

20.3.28 TCD Signed Destination Address Offset (DMA_TCD*n*_DOFF)

See TCD Signed Destination Address Offset register figure and DMA_TCD*n*_DOFF field descriptions table as follows.

Address: 0x1016 + *n**0x20 (*n* = 0 to 31) Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	DOFF															
Reset(1)	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x

1. x = Undefined at reset.

Figure 182. TCD Signed Destination Address Offset (DMA_TCD*n*_DOFF)

Table 276. DMA_TCD*n*_DOFF field descriptions

Field	Description
0:15 DOFF	Destination address signed offset Sign-extended offset applied to the current destination address to form the next-state value as each destination write is completed.

20.3.29 TCD Last Destination Address Adjustment/ Scatter Gather Address (DMA_TCD*n*_DLASTSGA)

Address: 0x1018 + *n**0x20 (*n* = 0 to 31) Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	DLASTSGA															
Reset(1)	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	DLASTSGA															
Reset(1)	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x

1. x = Undefined at reset.

Figure 183. TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD*n*_DLASTSGA)

Table 277. DMA_TCD*n*_DLASTSGA field descriptions

Field	Description

0:31
DLASTSGA
	Destination last address adjustment or the memory address for the next transfer control descriptor to be loaded into this channel (scatter/gather).
	If (TCD*n*_CSR[ESG] = 0) then
	– Adjustment value added to the destination address at the completion of the major iteration count. This value can apply to restore the destination address to the initial value or adjust the address to reference the next data structure
	else
	– This address points to the beginning of a 0-modulo-32-byte region containing the next transfer control descriptor to be loaded into this channel. This channel reload is performed as the major iteration count completes. The scatter/gather address must be 0-modulo-32-byte, else a configuration error is reported

20.3.30 TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD*n*_BITER_ELINKYES)

Address: 0x101C + *n**0x20 (*n* = 0 to 31) Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	ELINK	LINKCH							BITER							
Reset(1)	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x

1. x = Undefined at reset.

Figure 184. TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD*n*_BITER_ELINKYES)

Table 278. DMA_TCD*n*_BITER_ELINKYES field descriptions

Field	Description
0 ELINK	Enables channel-to-channel linking on minor loop complete As the channel completes the minor loop, this flag enables the linking to another channel, defined by BITER[LINKCH]. The link target channel initiates a channel service request via an internal mechanism that sets the TCD*n*_CSR[START] bit of the specified channel. If channel linking disables, the BITER value extends to 15 bits in place of a link channel number. If the major loop is exhausted, this link mechanism is suppressed in favor of the MAJORELINK channel linking. **Note:** When the software loads the TCD, this field must be set equal to the corresponding CITER field. Otherwise, a configuration error is reported. As the major iteration count is exhausted, the contents of this field is reloaded into the CITER field. 0 The channel-to-channel linking is disabled. 1 The channel-to-channel linking is enabled.
1:6 LINKCH	Link channel number If channel-to-channel linking is enabled (ELINK = 1), then after the minor loop is exhausted, the eDMA engine initiates a channel service request at the channel defined by these six bits by setting that channel's TCD*n*_CSR[START] bit. **Note:** When the software loads the TCD, this field must be set equal to the corresponding CITER field. Otherwise, a configuration error is reported. As the major iteration count is exhausted, the contents of this field is reloaded into the CITER field.
7:15 BITER	Starting major iteration count As the transfer control descriptor is first loaded by software, this 9-bit (ELINK = 1) or 15-bit (ELINK = 0) field must be equal to the value in the CITER field. As the major iteration count is exhausted, the contents of this field are reloaded into the CITER field. **Note:** When the software loads the TCD, this field must be set equal to the corresponding CITER field. Otherwise, a configuration error is reported. As the major iteration count is exhausted, the contents of this field is reloaded into the CITER field. If the channel is configured to execute a single service request, the initial values of BITER and CITER should be 0x0001.

20.3.31 TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD*n*_BITER_ELINKNO)

Address: 0x101C + *n**0x20 (*n* = 0 to 31) Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	ELINK	BITER														
Reset(1)	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x

1. x = Undefined at reset.

Figure 185. TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD*n*_BITER_ELINKNO)

Table 279. DMA_TCD*n*_BITER_ELINKNO field descriptions

Field	Description
0 ELINK	Enables channel-to-channel linking on minor loop complete As the channel completes the minor loop, this flag enables the linking to another channel, defined by BITER[LINKCH]. The link target channel initiates a channel service request via an internal mechanism that sets the TCD*n*_CSR[START] bit of the specified channel. If channel linking disables, the BITER value extends to 15 bits in place of a link channel number. If the major loop is exhausted, this link mechanism is suppressed in favor of the MAJORELINK channel linking. **Note:** When the software loads the TCD, this field must be set equal to the corresponding CITER field. Otherwise, a configuration error is reported. As the major iteration count is exhausted, the contents of this field is reloaded into the CITER field. 0 The channel-to-channel linking is disabled. 1 The channel-to-channel linking is enabled.
1:15 BITER	Starting major iteration count As the transfer control descriptor is first loaded by software, this 9-bit (ELINK = 1) or 15-bit (ELINK = 0) field must be equal to the value in the CITER field. As the major iteration count is exhausted, the contents of this field are reloaded into the CITER field. **Note:** When the software loads the TCD, this field must be set equal to the corresponding CITER field. Otherwise, a configuration error is reported. As the major iteration count is exhausted, the contents of this field is reloaded into the CITER field. If the channel is configured to execute a single service request, the initial values of BITER and CITER should be 0x0001.

20.3.32 TCD Control and Status (DMA_TCD*n*_CSR)

See TCD Control and Status register figure and DMA_TCD*n*_CSR field descriptions table as follows.

Address: 0x101E + *n**0x20 (*n* = 0 to 31) Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	BWC		MAJORLINKCH						DONE	ACTIVE	MAJORELINK	ESG	DREQ	INTHALF	INTMAJOR	START
Reset(1)	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x

1. x = Undefined at reset.

Figure 186. TCD Control and Status (DMA_TCD*n*_CSR)

Table 280. DMA_TCD*n*_CSR field descriptions

Field	Description
0:1 BWC	Bandwidth control Throttles the amount of bus bandwidth consumed by the eDMA. In general, as the eDMA processes the minor loop, it continuously generates read/write sequences until the minor count is exhausted. This field forces the eDMA to stall after the completion of each read/write access to control the bus request bandwidth seen by the crossbar switch (XBS). **Note:** If the source and destination sizes are equal, this field is ignored between the first and second transfers and after the last write of each minor loop. This behavior is a side effect of reducing start-up latency. 00 No eDMA engine stalls 01 Reserved 10 eDMA engine stalls for 4 cycles after each R/W 11 eDMA engine stalls for 8 cycles after each R/W

2:7
MAJORLINKCH
	Link channel number
	If (MAJORELINK = 0) then
	– No channel-to-channel linking (or chaining) is performed after the major loop counter is exhausted
	else
	– After the major loop counter is exhausted, the eDMA engine initiates a channel service request at the channel defined by these six bits by setting that channel's TCD*n*_CSR[START] bit.

8 DONE	Channel done This flag indicates the eDMA has completed the major loop. The eDMA engine sets it as the CITER count reaches zero; The software clears it, or the hardware when the channel is activated. **Note:** This bit must be cleared to write the MAJORELINK or ESG bits. This bit resets to zero.
9 ACTIVE	Channel active This flag signals the channel is currently in execution. It is set when channel service begins, and the eDMA clears it as the minor loop completes or if any error condition is detected.
10 MAJORELINK	Enable channel-to-channel linking on major loop complete As the channel completes the major loop, this flag enables the linking to another channel, defined by MAJORLINKCH. The link target channel initiates a channel service request via an internal mechanism that sets the TCD*n*_CSR[START] bit of the specified channel. **Note:** To support the dynamic linking coherency model, this field is forced to zero when written to while the TCD*n*_CSR[DONE] bit is set. 0 The channel-to-channel linking is disabled 1 The channel-to-channel linking is enabled

Table 280. DMA_TCD*n*_CSR field descriptions (continued)

Field	Description
11 ESG	Enable scatter/gather processing As the channel completes the major loop, this flag enables scatter/gather processing in the current channel. If enabled, the eDMA engine uses DLASTSGA as a memory pointer to a 0-modulo-32 address containing a 32-byte data structure loaded as the transfer control descriptor into the local memory. **Note:** To support the dynamic scatter/gather coherency model, this field is forced to zero when written to while the TCD*n*_CSR[DONE] bit is set. 0 The current channel's TCD is normal format. 1 The current channel's TCD specifies a scatter gather format. The DLASTSGA field provides a memory pointer to the next TCD to be loaded into this channel after the major loop completes its execution.
12 DREQ	Disable request 0 The channel's ERQ bit is not affected 1 The channel's ERQ bit is cleared when the major loop is complete
13 INTHALF	Enable an interrupt when major counter is half complete. If this flag is set, the channel generates an interrupt request by setting the appropriate bit in the INT when the current major iteration count reaches the halfway point. Specifically, the comparison performed by the eDMA engine is (CITER == (BITER >> 1)). This halfway point interrupt request is provided to support double-buffered (aka ping-pong) schemes or other types of data movement where the processor needs an early indication of the transfer's progress. **Note:** If BITER is set, do not use INTHALF. Use INTMAJOR instead. 0 The half-point interrupt is disabled. 1 The half-point interrupt is enabled .
14 INTMAJOR	Enable an interrupt when major iteration count completes If this flag is set, the channel generates an interrupt request by setting the appropriate bit in the INT when the current major iteration count reaches zero. 0 The end-of-major loop interrupt is disabled. 1 The end-of-major loop interrupt is enabled.
15 START	Channel start If this flag is set, the channel is requesting service. The eDMA hardware automatically clears this flag after the channel begins execution. This bit resets to zero. 0 The channel is not explicitly started. 1 The channel is explicitly started via a software initiated service request;

20.4 Functional description

This section provides a description of the DMA request sources, an overview of the microarchitecture, and functional operation of the eDMA module.

20.4.1 eDMA microarchitecture

The eDMA module is partitioned into two major modules: the eDMA engine and the transfer-control descriptor local memory. Additionally, the eDMA engine is further partitioned into four submodules:

- eDMA engine
  - Address path:

    This block implements registered versions of two channel transfer control descriptors, channel x and channel y, and manages all master bus-address calculations. All the channels provide the same functionality. This structure allows data transfers associated with one channel to be preempted after the completion of a read/write sequence if a higher priority channel activation is asserted while the first channel is active. After a channel is activated, it runs until the minor loop is completed, unless preempted by a higher priority channel. This provides a mechanism (enabled by DCHPRI*n*[ECP]) where a large data move operation can be preempted to minimize the time another channel is blocked from execution.

    When any channel is selected to execute, the contents of its TCD are read from local memory and loaded into the address path channel x registers for a normal start and into channel y registers for a preemption start. After the minor loop completes execution, the address path hardware writes the new values for the TCD*n* _{SADDR, DADDR, CITER} back to local memory. If the major iteration count is exhausted, additional processing is performed, including updates to the final address pointer, reloading the TCD*n* _CITER field, and a possible fetch of the next TCD*n* from memory as part of a scatter/gather operation.
  - Data path:

    This block implements the bus master read/write datapath. It includes 32 bytes of register storage and the necessary multiplex logic to support any data alignment. The internal read data bus is the primary input, and the internal write data bus is the primary output.

    The address and data path modules directly support the two-stage pipelined internal bus. The address path module represents the first stage of the bus pipeline (address phase), while the data path module implements the second stage of the pipeline (data phase).
  - Program model/channel arbitration:

    This block implements the first section of the eDMA programming model as well as the channel arbitration logic. The programming model registers are connected to the internal peripheral bus (not shown). The eDMA peripheral request inputs and interrupt request outputs are also connected to this block (via control logic).
  - Control:

    This block provides all the control functions for the eDMA engine. For data transfers where the source and destination sizes are equal, the eDMA engine performs a series of source read/destination write operations until the number of bytes specified in the minor loop byte count has moved. For descriptors where the sizes are not equal, multiple accesses of the smaller size data are required for each reference of the larger size. As an example, if the source size references 16-

bit data and the destination is 32-bit data, two reads are performed, then one 32-bit write.

- Transfer control descriptor memory
  - Memory controller:

    This logic implements the dual-ported controller, managing accesses from the eDMA engine as well as references from the internal peripheral bus.

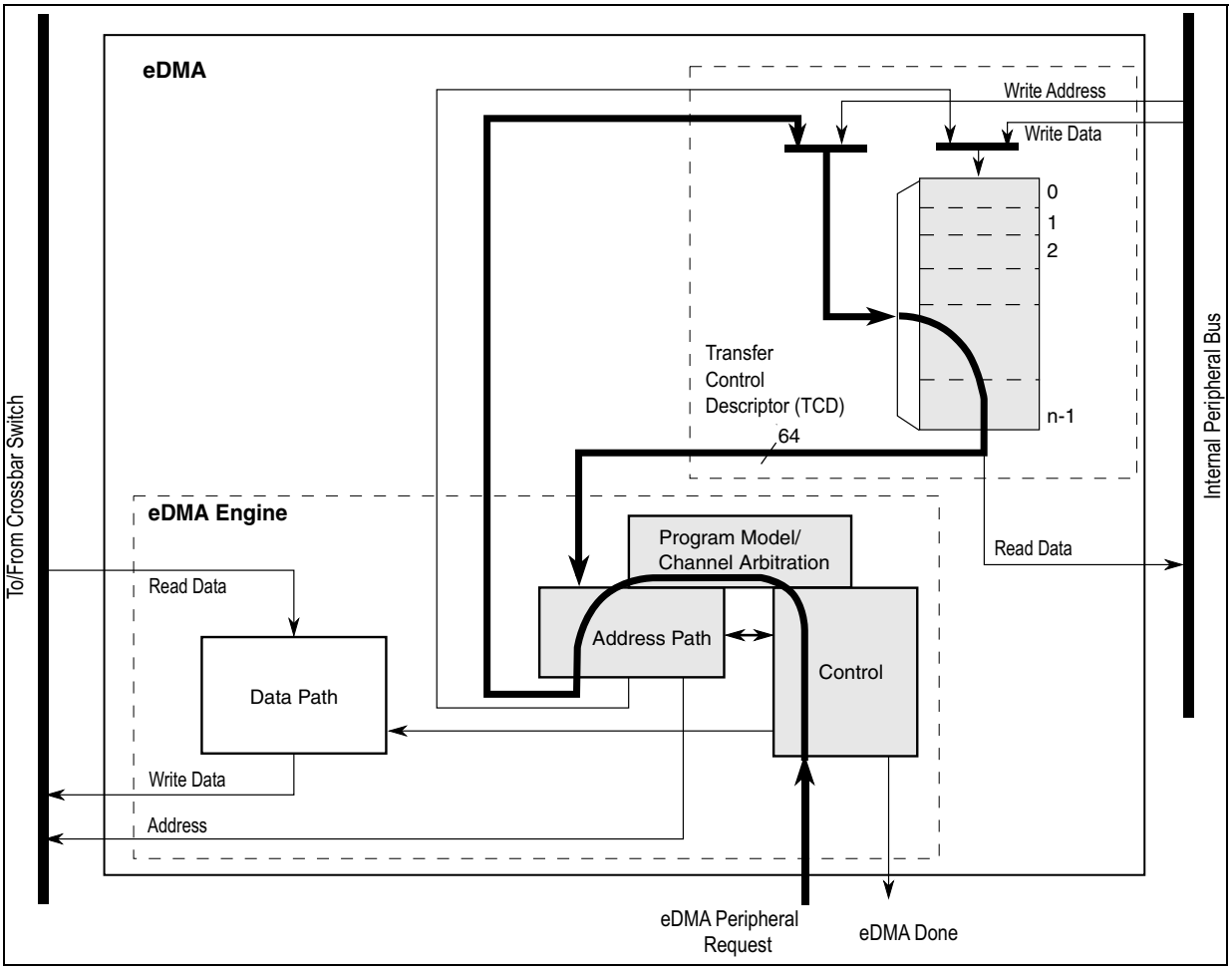
    As noted earlier, in the event of simultaneous accesses, the eDMA engine is given priority and the peripheral transaction is stalled.
  - Memory array:

    TCD storage is implemented using a single-port, synchronous RAM array.

20.4.2 eDMA basic data flow

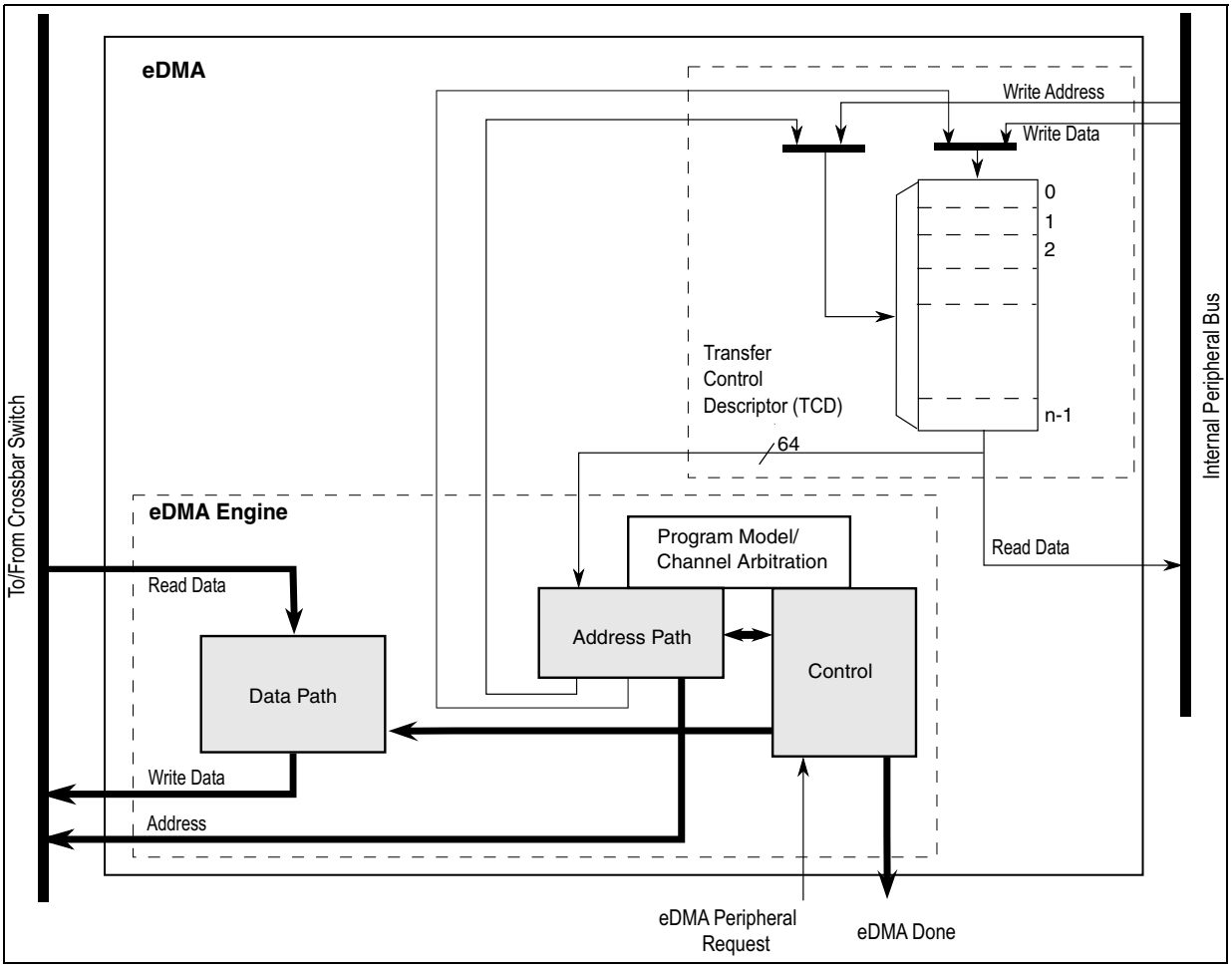
The basic flow of a data transfer can be partitioned into three segments. As shown in *Figure 187*, the first segment involves the channel activation. In the diagram, this example uses the assertion of the eDMA peripheral request signal to request service for channel *n*. Channel activation via software and the TCD*n*_CSR[START] bit follows the same basic flow as peripheral requests. The eDMA request input signal is registered internally and then routed through the eDMA engine: first through the control module, then into the program model and channel arbitration. In the next cycle, channel arbitration is performed, using the fixed-priority or round-robin algorithm. After arbitration is complete, the activated channel number is sent through the address path and converted into the address to access the local memory for TCD*n*. Next, the TCD memory is accessed and the descriptor read from the local memory and loaded into the eDMA engine address path channel x or y registers. The TCD memory is 64 bits wide to minimize the time needed to fetch the activated channel descriptor and load it into the address path channel x or y registers.

Figure 187. DMA operation, part 1



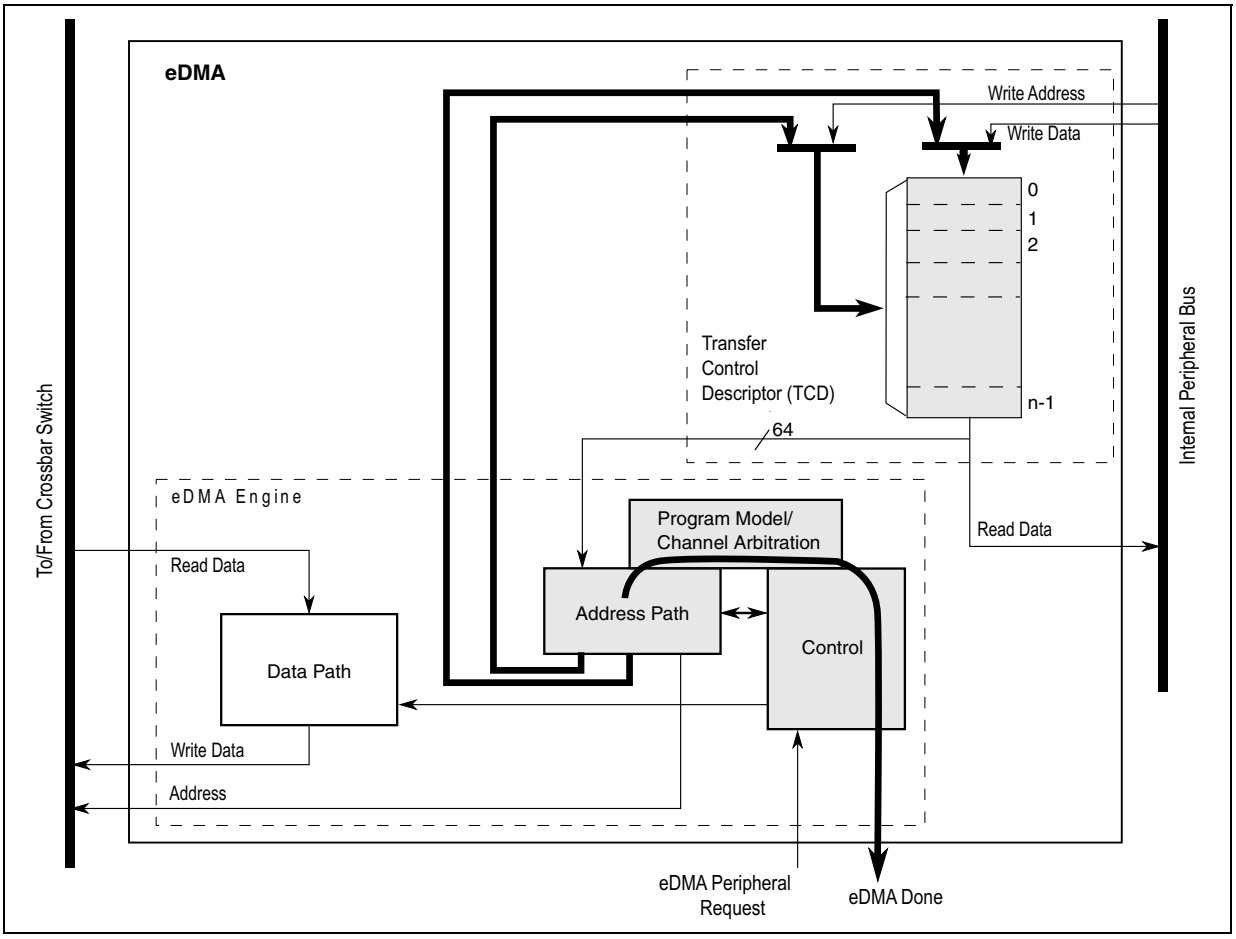
In the second part of the basic data flow as shown in *Figure 188* the modules associated with the data transfer (address path, data path, and control) sequence through source reads and destination writes to perform the actual data movement. The source reads are initiated and the fetched data is temporarily stored in the data path block until it is gated onto the internal bus during the destination write. This source read/destination write processing continues until the minor byte count has transferred.

Figure 188. DMA operation, part 2



After the minor byte count has moved, the final phase of the basic data flow is performed. In this segment, the address path logic performs updates to certain fields in the appropriate TCD, for example, SADDR, DADDR, and CITER. If the major iteration count is exhausted, additional operations are performed. These include the final address adjustments and reloading of the BITER field into the CITER. Assertion of an optional interrupt request also occurs at this time, as does a possible fetch of a new TCD from memory using the scatter/gather address pointer included in the descriptor. The updates to the TCD memory and the assertion of an interrupt request are shown in *Figure 189*.

Figure 189. DMA operation, part 3



20.4.3 Error reporting and handling

Channel errors are reported in the ES register and can be caused by:

- A configuration error (an illegal setting in the transfer-control descriptor or an illegal priority register setting in Fixed Arbitration mode),
- An uncorrectable ECC error, or
- An error termination to a bus master read or write cycle

A configuration error is reported when the starting source or destination address, source or destination offsets, minor loop byte count, or the transfer size represent an inconsistent state. Each of these possible causes are detailed in the list below:

- The addresses and offsets must be aligned on 0-modulo-transfer-size boundaries
- The minor loop byte count must be a multiple of the source and destination transfer sizes
- All source reads and destination writes must be configured to the natural boundary of the programmed transfer size, respectively
- In Fixed Arbitration mode, a configuration error is caused by any two channel priorities being equal within a group, or any group priority levels being equal among the groups.

All channel priority levels within a group must be unique and all group priority levels among the groups must be unique when Fixed Arbitration mode is enabled

- If a scatter/gather operation is enabled upon channel completion, a configuration error is reported if the scatter/gather address (DLAST_SGA) is not aligned on a 32-byte boundary
- If minor loop channel linking is enabled upon channel completion, a configuration error is reported when the link is attempted if the TCD*n*_CITER[E_LINK] bit does not equal the TCD*n*_BITER[E_LINK] bit

If enabled, all configuration error conditions, except the scatter/gather and minor-loop link errors, report as the channel activates and asserts an error interrupt request. A scatter/gather configuration error is reported when the scatter/gather operation begins at major loop completion when properly enabled. A minor loop channel link configuration error is reported when the link operation is serviced at minor loop completion.

If a system bus read or write is terminated with an error, the data transfer is stopped and the appropriate bus error flag set. In this case, the state of the channel's transfer control descriptor is updated by the eDMA engine with the current source address, destination address, and current iteration count at the point of the fault. When a system-bus error occurs, the channel terminates after the read or write transaction (which is already pipelined after errant access) has completed. If a bus error occurs on the last read prior to beginning the write sequence, the write executes using the data captured during the bus error. If a bus error occurs on the last write prior to switching to the next read sequence, the read sequence executes before the channel terminates due to the destination bus error.

If an uncorrectable ECC error occurs during a peripheral bus access, the access is terminated with a bus error. The occurrence of an uncorrectable ECC error during an eDMA engine read causes the eDMA engine to stop the active channel immediately. The ECC error and channel number are recorded in the eDMA's Error Status register.

A transfer may be cancelled by software with the CR[CX] bit. When a cancel transfer request is recognized, the DMA engine stops processing the channel. The current read-write sequence is allowed to finish. If the cancel occurs on the last read-write sequence of a major or minor loop, the cancel request is discarded and the channel retires normally.

The error cancel transfer is the same as a cancel transfer except the ES register is updated with the cancelled channel number and ECX is set. The TCD of a cancelled channel contains the source and destination addresses of the last transfer saved in the TCD. If the channel needs to be restarted, you must re-initialize the TCD since the aforementioned fields no longer represent the original parameters. When a transfer is cancelled by the error cancel transfer mechanism, the channel number is loaded into DMA_ES[ERRCHN] and ECX and VLD are set. In addition, an error interrupt may be generated if enabled.

The occurrence of any error causes the eDMA engine to stop the active channel immediately, and the appropriate channel bit in the eDMA error register is asserted. At the same time, the details of the error condition are loaded into the ES register. The major loop complete indicators, setting the transfer control descriptor DONE flag and the possible assertion of an interrupt request, are not affected when an error is detected. After the error status has been updated, the eDMA engine continues operating by servicing the next appropriate channel. A channel that experiences an error condition is not automatically disabled. If a channel is terminated by an error and then issues another service request before the error is fixed, that channel executes and terminates with the same error condition.

20.4.4 Channel preemption

Channel preemption is enabled on a per-channel basis by setting the DCHPRI*n*[ECP] bit. Channel preemption allows the executing channel's data transfers to temporarily suspend in favor of starting a higher priority channel. After the preempting channel has completed all its minor loop data transfers, the preempted channel is restored and resumes execution. After the restored channel completes one read/write sequence, it is again eligible for preemption. If any higher priority channel is requesting service, the restored channel is suspended and the higher priority channel is serviced. Nested preemption (attempting to preempt a preempting channel) is not supported. After a preempting channel begins execution, it cannot be preempted. Preemption is available only when fixed arbitration is selected for both group and channel arbitration modes.

A channel's ability to preempt another channel can be disabled by setting DCHPRI*n*[DPA]. When a channel's preempt ability is disabled, that channel cannot suspend a lower priority channel's data transfer, regardless of the lower priority channel's ECP setting. This allows for a pool of low-priority, large data-moving channels to be defined. These low-priority channels can be configured so that they do not preempt each other, thus preventing a low-priority channel from consuming the preempt slot normally available to a true, high-priority channel.

20.4.5 eDMA performance

This section addresses the performance of the eDMA module, focusing on two separate metrics. In the traditional data movement context, performance is best expressed as the peak data transfer rates achieved using the eDMA. In most implementations, this transfer rate is limited by the speed of the source and destination address spaces. In a second context where device-paced movement of single data values to/from peripherals is dominant, a measure of the requests that can be serviced in a fixed time is a more interesting metric. In this environment, the speed of the source and destination address spaces remains important, but the microarchitecture of the eDMA also factors significantly into the resulting metric.

The peak transfer rates for several different source and destination transfers are shown in *Table 281* and *Table 282*. The following assumptions apply to those tables:

- Internal SRAM can be accessed with zero wait states when viewed from the system bus data phase
- All internal peripheral bus reads require one wait-state, and internal peripheral bus writes one wait-state, again viewed from the system bus data phase
- All internal peripheral bus accesses are 32 bits in size
- All internal SRAM accesses are 64 bits in size

Table 281 presents a peak transfer rate comparison, measured in megabytes per second.

Table 281. eDMA peak transfer rates (Mbyte/s)

System speed, width	From internal SRAM to internal SRAM	From 32-bit internal peripheral bus to internal SRAM	From internal SRAM to 32-bit internal peripheral bus
100 MHz, 64 bits	400.0	160.0	160.0

Table 281. eDMA peak transfer rates (Mbyte/s) (continued)

System speed, width	From internal SRAM to internal SRAM	From 32-bit internal peripheral bus to internal SRAM	From internal SRAM to 32-bit internal peripheral bus
150 MHz, 64 bits	600.0	240.0	240.0
200 MHz, 64 bits	800.0	320.0	320.0

The second performance metric is a measure of the number of DMA requests that can be serviced in a given amount of time. For this metric, it is assumed the peripheral request causes the channel to move a single internal peripheral bus-mapped operand to/from internal SRAM. The same timing assumptions used in the previous example apply to this calculation. In particular, this metric also reflects the time required to activate the channel. The eDMA design supports the following hardware service request sequence:

- Cycle 1: eDMA peripheral request is asserted
- Cycle 2: The eDMA peripheral request is registered locally in the eDMA module and qualified. (TCD*n* _CSR[START] bit initiated requests start at this point with the registering of the user write to TCD*n* Word 7)
- Cycle 3: Channel arbitration begins
- Cycle 4: Channel arbitration completes. The transfer control descriptor local memory read is initiated
- Cycles 5–6: The first two parts of the activated channel's TCD is read from the local memory. The memory width to the eDMA engine is 64 bits, so the entire descriptor can be accessed in four cycles
- Cycle 7: The first system bus read cycle is initiated, as the third part of the channel's TCD is read from the local memory. Depending on the state of the crossbar switch, arbitration at the system bus may insert an additional cycle of delay here
- Cycles 8 to x[(h)]: The last part of the TCD is read in. This cycle represents the first data phase for the read, and the address phase for the destination write.

  The exact timing from this point is a function of the transfer size and response times for the channel's read and write accesses. In this case of two internal peripheral bus reads and a single 64-bit internal SRAM write, the combined data phase time is five cycles.

h. "*x*" represents an undefined cycle value dependant on transfer sizes and response times for the channel's read and write accesses.

For a single SRAM read and two 64-bit internal peripheral bus writes, the combined data phase time is five cycles

- Cycle $x^{(h)}$ + 1: This cycle represents the data phase of the last destination write
- Cycle $x^{(h)}$ + 2: The eDMA engine completes the execution of the inner minor loop and prepares to write back the required TCD*n* fields into the local memory. The TCD*n* Word 7 is read and checked for channel linking or scatter/gather requests
- Cycle $x^{(h)}$ + 3: The appropriate fields in the first part of the TCD*n* are written back into the local memory
- Cycle $x^{(h)}$ + 4: The fields in the second part of the TCD*n* are written back into the local memory. This cycle coincides with the next channel arbitration cycle start
- Cycle $x^{(h)}$ + 5: The next channel to be activated performs the read of the first part of its TCD from the local memory. This is equivalent to Cycle 4 for the first channel's service request

Assuming zero wait states on the system bus, DMA requests can be processed every nine cycles. Assuming an average of the access times associated with internal peripheral bus-to-SRAM (three cycles) and SRAM-to-internal peripheral bus (three cycles), DMA requests can be processed every 10 cycles (4 + (3 + 3)/2 + 3). This is the time from Cycle 4 to Cycle *x* + 5. The resulting peak request rate, as a function of the system frequency, is shown in *Table 282*. This metric represents millions of requests per second. A single read/write operation with same transfer size for the source and destination is assumed.

Table 282. eDMA peak request rate (MReq/s)

System frequency (MHz)	Request rate	
	zero wait states	with wait states
100.0	11.1	10.0
150.0	16.6	15.0
200.0	22.2	20.0

A general formula to compute the peak request rate (with overlapping requests) is:

Equation 2 PEAKreq = freq / [entry + (1 + read_ws) + (1 + write_ws) + exit]

where:

- PEAKreq = peak request rate
- freq = system frequency
- entry = channel startup (4 cycles)
- read_ws = wait states seen during the system bus read data phase
- write_ws = wait states seen during the system bus write data phase
- exit = channel shutdown (3 cycles)

For example, consider a system with the following characteristics:

- Internal SRAM can be accessed with zero wait states when viewed from the system bus data phase
- All internal peripheral bus reads require one wait state, and internal peripheral bus writes one wait-state, again viewed from the system bus data phase
- System operates at 150 MHz

For an SRAM to internal peripheral bus transfer:

Equation 3 PEAKreq = 150 MHz / [4 + (1 + 0) + (1 + 1) + 3] cycles = 15.0 Mreq/s

For an internal peripheral bus-to-SRAM transfer:

Equation 4 PEAKreq = 150 MHz / [4 + (1 + 1) + (1 + 0) + 3] cycles = 15.0 Mreq/s

The minimum number of cycles to perform a single read/write, with zero wait states on the system bus, from a cold start (where no channel is executing and the eDMA is idle):

- 11 cycles for a software (TCD*n*_CSR[START] bit) request
- 12 cycles for a hardware (eDMA peripheral request signal) request

Two cycles account for the arbitration pipeline and one extra cycle on the hardware request resulting from the internal registering of the eDMA peripheral request signals. For the peak request rate calculations above, the arbitration and request registering are absorbed in or overlap the previous executing channel.

When channel linking or scatter/gather is enabled, a two cycle delay is imposed on the next channel selection and startup. This allows the link channel or the scatter/gather channel to be eligible and considered in the arbitration pool for next channel selection.

20.5 Initialization/application information

The following sections discuss initialization of the eDMA and programming considerations.

20.5.1 eDMA initialization

A typical initialization of the eDMA has the following sequence:

1. Write the CR register if a configuration other than the default is desired
2. Write the channel priority levels into the DCHPRI*n* registers if a configuration other than the default is desired
3. Enable error interrupts in the EEI registers if so desired
4. Write the 32-byte TCD for each channel that may request service
5. Enable any hardware service requests via the ERQ register
6. Request channel service by software (setting the TCD*n*_CSR[START] bit) or hardware (slave device asserting its eDMA peripheral request signal)

After any channel requests service, a channel is selected for execution based on the arbitration and priority levels written into the programmer's model. The eDMA engine reads the entire TCD, including the TCD control and status fields (shown in *Table 283*) for the selected channel into its internal address path module. As the TCD is read, the first transfer is initiated on the internal bus unless a configuration error is detected. Transfers from the source (as defined by the source address, TCD*n*_SADDR) to the destination (as defined by the destination address, TCD*n*_DADDR) continue until the specified number of bytes (TCD*n*_NBYTES) are transferred. When the transfer is complete, the eDMA engine's local TCD*n*_SADDR, TCD*n*_DADDR, and TCD*n*_CITER are written back to the main TCD memory and any minor loop channel linking is performed, if enabled. If the major loop is exhausted, further post processing executes (interrupts, major loop channel linking, and scatter/gather operations) if enabled.

Table 283. TCD control and status fields

TCD*n*_CSR field name	Description
START	Control bit to start channel explicitly when using a software initiated DMA service (automatically cleared by hardware)
ACTIVE	Status bit indicating the channel is currently in execution
DONE	Status bit indicating major loop completion (cleared by software when using a software initiated DMA service)
D_REQ	Control bit to disable DMA request at end of major loop completion when using a hardware initiated DMA service
BWC	Control bits for throttling bandwidth control of a channel
E_SG	Control bit to enable scatter-gather feature
INT_HALF	Control bit to enable interrupt when major loop is half complete
INT_MAJ	Control bit to enable interrupt when major loop completes

Figure 190 shows how each DMA request initiates one minor-loop transfer (iteration) without CPU intervention. DMA arbitration can occur after each minor loop, and one level of minor loop DMA preemption is allowed. The number of minor loops in a major loop is specified by the beginning iteration count (BITER).

Figure 190. Example of multiple loop iterations

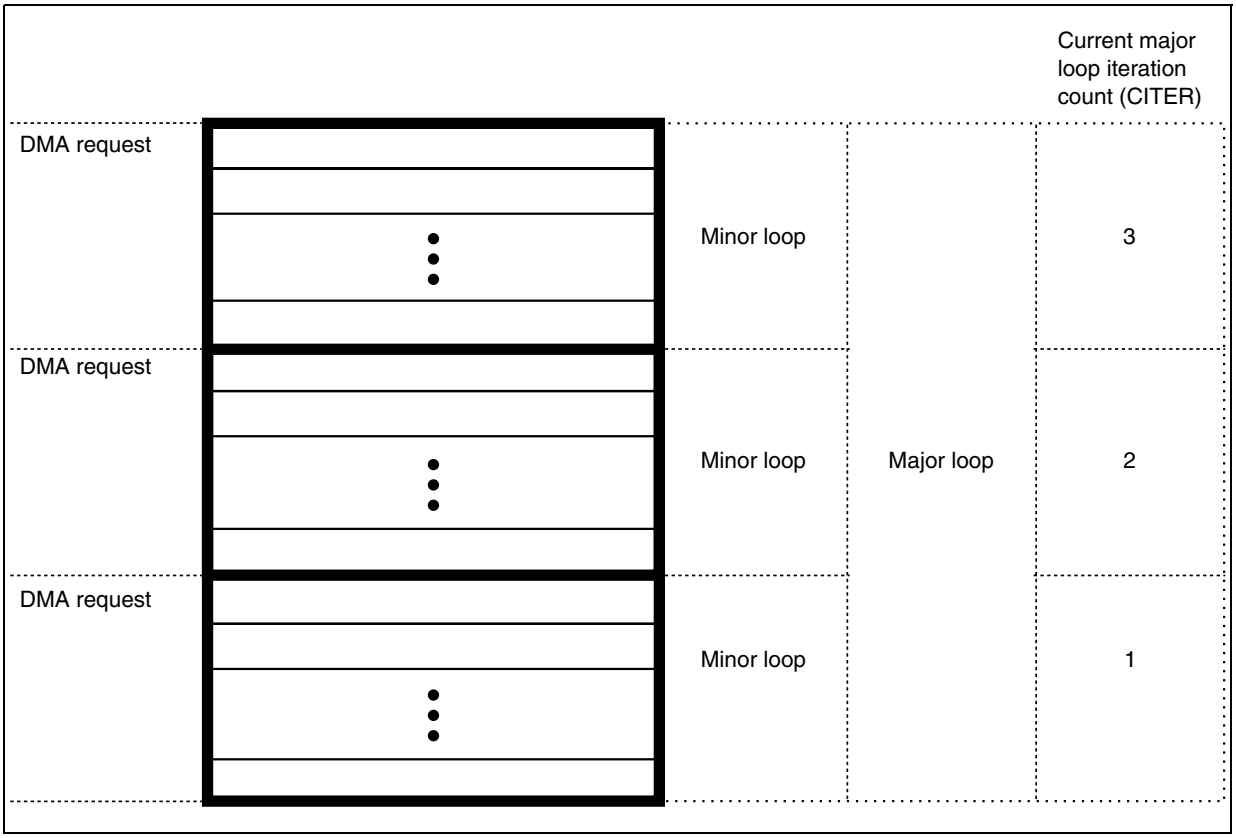
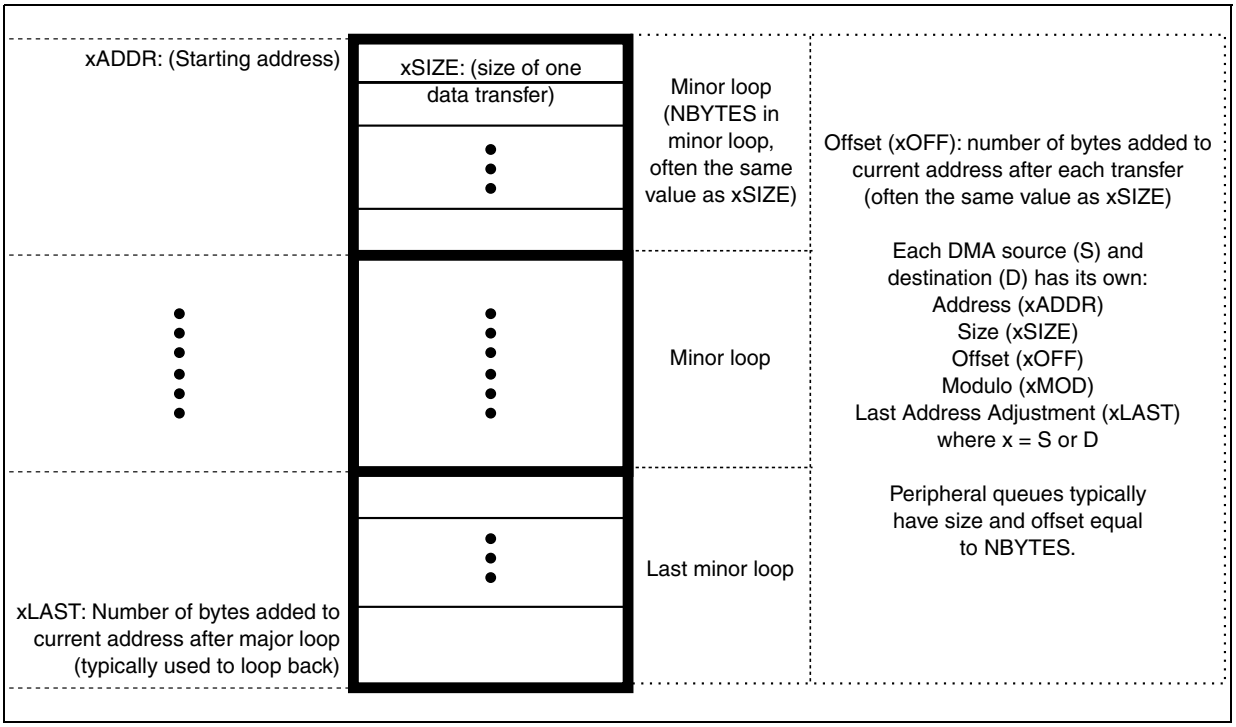


Figure 191 lists the memory array terms and how the TCD settings interrelate.

Figure 191. Memory array terms



20.5.2 DMA programming errors

The eDMA performs various tests on the transfer control descriptor to verify consistency in the descriptor data. Most programming errors are reported on a per-channel basis with the exception of two errors: group priority error (ES[GPE]) and channel priority error (ES[CPE]).

For all error types other than group or channel priority errors, the channel number causing the error is recorded in the ES register. If the error source is not removed before the next activation of the problem channel, the error is detected and recorded again.

Channel priority errors are identified within a group once that group has been selected as the active group. For example:

1. The eDMA is configured for fixed-group and fixed-channel arbitration modes
2. Group 1 is the highest priority and all channels are unique in that group
3. Group 0 is the next highest priority and has two channels with the same priority level
4. If Group 1 has any service requests, those requests will be executed
5. Once all of Group 1 requests have completed, Group 0 will be the next active group.
6. If Group 0 has a service request, then an undefined channel in Group 0 will be selected and a channel priority error will occur
7. This repeats until all of the Group 0 requests have been removed or a higher priority Group 1 request comes in

In this sequence, for item 2, the eDMA Acknowledge lines will assert only if the selected channel is requesting service via the eDMA peripheral request signal. If interrupts are enabled for all channels, the user will get an error interrupt, but the channel number for the ERR register and the error interrupt request line may be wrong because they reflect the

selected channel. A group priority error is global and any request in any group will cause a group priority error.

If priority levels are not unique, when any channel requests service, a channel priority error is reported. The highest (channel/group) priority with an active request is selected, but the lowest numbered (channel/group) with that priority is selected by arbitration and executed by the eDMA engine. The hardware service request handshake signals, error interrupts, and error reporting is associated with the selected channel.

20.5.3 DMA Arbitration mode considerations

This section discusses arbitration considerations for the eDMA.

20.5.3.1 Fixed-group arbitration, fixed-channel arbitration

In this mode, the channel service request from the highest priority channel in the highest priority group is selected to execute. If the eDMA is programmed so the channels within one group use “fixed” priorities, and that group is assigned the highest “fixed” priority of all groups, it is possible for that group to take all the bandwidth of the eDMA controller: that is, no other groups will be serviced if there is always at least one DMA request pending on a channel in the highest priority group when the controller arbitrates the next DMA request. The advantage of this scenario is that latency can be small for channels that need to be serviced quickly. Preemption is available in this scenario only.

20.5.3.2 Round-robin group arbitration, fixed-channel arbitration

The occurrence of one or more DMA requests from one or more groups, the channel with the highest priority from a specific group will be serviced first. Groups are serviced starting with the highest group number with an service request and rotating through to the lowest group number containing a service request.

Once the channel request is serviced, the group round-robin algorithm will select the highest pending request from the next group in the round-robin sequence. Servicing continues round-robin, always servicing the highest priority channel in the next group in the sequence, or just skipping a group if it has no pending requests.

If a channel requests service at a rate that equals or exceeds the round-robin service rate, then that channel will always be serviced before lower priority channels in the same group, and thus the lower priority channels will never be serviced. The advantage of this scenario is that no one group will consume all the eDMA bandwidth. The highest priority channel selection latency is potentially greater than fixed/fixed arbitration. Excessive request rates on high priority channels could prevent the servicing of lower priority channels in the same group.

20.5.3.3 Round-robin group arbitration, round-robin channel arbitration

Groups will be serviced as described in *Section 20.5.3.2: Round-robin group arbitration, fixed-channel arbitration*, but this time channels will be serviced in channel number order. Only one channel is serviced from each requesting group for each round-robin pass through the groups.

Within each group, channels are serviced starting with the highest channel number and rotating through to the lowest channel number without regard to channel priority levels.

Because channels are serviced in round-robin manner, any channel that generates DMA requests faster than a combination of the group round-robin service rate and the channel

service rate for its group will not prevent the servicing of other channels in its group. Any DMA requests that are not serviced are simply lost, but at least one channel will be serviced.

This scenario ensures that all channels will be guaranteed service at some point, regardless of the request rates. However, the potential latency could be quite high. All channels are treated equally. Priority levels are not used in round-robin/round-robin mode.

20.5.3.4 Fixed-group arbitration, round-robin channel arbitration

The highest priority group with a request will be serviced. Lower priority groups will be serviced if no pending requests exist in the higher priority groups.

Within each group, channels are serviced starting with the highest channel number and rotating through to the lowest channel number without regard to the channel priority levels assigned within the group.

This scenario could cause the same bandwidth consumption problem as indicated in *Section 20.5.3.1: Fixed-group arbitration, fixed-channel arbitration* but all the channels in the highest priority group will get serviced. Service latency will be short on the highest priority group, but could potentially get very much longer and longer as the group priority decreases.

20.5.4 DMA transfer

This section discusses how to perform DMA transfers with the eDMA.

20.5.4.1 Single request

To perform a simple transfer of *n* bytes of data with one activation, set the major loop to one (TCD*n*_CITER = TCD*n*_BITER = 1). The data transfer begins after the channel service request is acknowledged and the channel is selected to execute. After the transfer is complete, the TCD*n*_CSR[DONE] bit is set and an interrupt generates if properly enabled.

For example, the following TCD entry is configured to transfer 16 bytes of data. The eDMA is programmed for one iteration of the major loop, transferring 16 bytes per iteration. The source memory has a byte wide memory port located at 0x1000. The destination memory has a 32-bit-wide port located at 0x2000. The address offsets are programmed in increments to match the transfer size: one byte for the source and four bytes for the destination. The final source and destination addresses are adjusted to return to their beginning values.

```
TCDn_CITER = TCDn_BITER = 1
TCDn_NBYTES = 16
TCDn_SADDR = 0x1000
TCDn_SOFF = 1
TCDn_ATTR[SSIZE] = 0
TCDn_SLAST = -16
TCDn_DADDR = 0x2000
TCDn_DOFF = 4
TCDn_ATTR[DSIZE] = 2
TCDn_DLAST_SGA= -16
TCDn_CSR[INT_MAJ] = 1
TCDn_CSR[START] = 1 (Should be written last after all other fields have
been initialized)
```

```
All other TCDn fields = 0
```

This generates the following event sequence:

1. User write to the TCD*n*_CSR[START] bit requests channel service
2. The channel is selected by arbitration for servicing
3. eDMA engine writes: TCD*n*_CSR[DONE] = 0, TCD*n*_CSR[START] = 0, TCD*n*_CSR[ACTIVE] = 1
4. eDMA engine reads: channel TCD data from local memory to internal register file
5. The source-to-destination transfers are executed as follows:
   a) Read byte from location 0x1000, read byte from location 0x1001, read byte from 0x1002, read byte from 0x1003
   b) Write 32 bits to location 0x2000 → first iteration of the minor loop
   c) Read byte from location 0x1004, read byte from location 0x1005, read byte from 0x1006, read byte from 0x1007
   d) Write 32 bits to location 0x2004 → second iteration of the minor loop
   e) Read byte from location 0x1008, read byte from location 0x1009, read byte from 0x100A, read byte from 0x100B
   f) Write 32 bits to location 0x2008 → third iteration of the minor loop
   g) Read byte from location 0x100C, read byte from location 0x100D, read byte from 0x100E, read byte from 0x100F
   h) Write 32 bits to location 0x200C → last iteration of the minor loop → major loop complete
6. The eDMA engine writes: TCD*n*_SADDR = 0x1000, TCD*n*_DADDR = 0x2000, TCD*n*_CITER = 1 (TCD*n*_BITER)
7. The eDMA engine writes: TCD*n*_CSR[ACTIVE] = 0, TCD*n*_CSR[DONE] = 1, INT[*n*] = 1
8. The channel retires and the eDMA goes idle or services the next channel

20.5.4.2 Multiple requests

Besides transferring 32 bytes via two hardware requests, the next example is the same as previous. The only fields that change are the major loop iteration count and the final address offsets. The eDMA is programmed for two iterations of the major loop transferring 16 bytes per iteration. After the channel's hardware requests are enabled in the ERQ register, the slave device initiates channel service requests.

```
TCDn_CITER = TCDn_BITER = 2
TCDn_SLAST = -32
TCDn_DLAST_SGA = -32
```

This would generate the following sequence of events:

1. First hardware (eDMA peripheral) request for channel service
2. The channel is selected by arbitration for servicing
3. eDMA engine writes: TCD*n*_CSR[DONE] = 0, TCD*n*_CSR[START] = 0, TCD*n*_CSR[ACTIVE] = 1
4. eDMA engine reads: channel TCD*n* data from local memory to internal register file
5. The source to destination transfers are executed as follows:
   a) Read byte from location 0x1000, read byte from location 0x1001, read byte from 0x1002, read byte from 0x1003
   b) Write 32 bits to location 0x2000 → first iteration of the minor loop
   c) Read byte from location 0x1004, read byte from location 0x1005, read byte from 0x1006, read byte from 0x1007
   d) Write 32 bits to location 0x2004 → second iteration of the minor loop
   e) Read byte from location 0x1008, read byte from location 0x1009, read byte from 0x100A, read byte from 0x100B
   f) Write 32 bits to location 0x2008 → third iteration of the minor loop
   g) Read byte from location 0x100C, read byte from location 0x100D, read byte from 0x100E, read byte from 0x100F
   h) Write 32 bits to location 0x200C → last iteration of the minor loop
6. eDMA engine writes: TCD*n*_SADDR = 0x1010, TCD*n*_DADDR = 0x2010, TCD*n*_CITER = 1
7. eDMA engine writes: TCD*n*_CSR[ACTIVE] = 0
8. The channel retires → one iteration of the major loop. The eDMA goes idle or services the next channel
9. Second hardware (eDMA peripheral) requests channel service
10. The channel is selected by arbitration for servicing
11. eDMA engine writes: TCD*n*_CSR[DONE] = 0, TCD*n*_CSR[START] = 0, TCD*n*_CSR[ACTIVE] = 1
12. eDMA engine reads: channel TCD data from local memory to internal register file
13. The source to destination transfers are executed as follows:
   a) Read byte from location 0x1010, read byte from location 0x1011, read byte from 0x1012, read byte from 0x1013
   b) Write 32 bits to location 0x2010 → first iteration of the minor loop
   c) Read byte from location 0x1014, read byte from location 0x1015, read byte from 0x1016, read byte from 0x1017
   d) Write 32 bits to location 0x2014 → second iteration of the minor loop
   e) Read byte from location 0x1018, read byte from location 0x1019, read byte from 0x101A, read byte from 0x101B
   f) Write 32 bits to location 0x2018 → third iteration of the minor loop
   g) Read byte from location 0x101C, read byte from location 0x101D, read byte from 0x101E, read byte from 0x101F
   h) Write 32 bits to location 0x201C → last iteration of the minor loop → major loop complete
14. eDMA engine writes: TCD*n*_SADDR = 0x1000, TCD*n*_DADDR = 0x2000, TCD*n*_CITER = 2(TCD*n*_BITER)
15. eDMA engine writes: TCD*n*_CSR[ACTIVE] = 0, TCD*n*_CSR[DONE] = 1, INT[*n*] = 1

16. The channel retires → major loop complete. The eDMA goes idle or services the next channel

20.5.4.3 Modulo feature

The modulo feature of the eDMA provides the ability to implement a circular data queue in which the size of the queue is a power of 2. MOD is a 5-bit field for the source and destination in the TCD, and it specifies which lower address bits increment from their original value after the address+offset calculation. All upper address bits remain the same as in the original value. A setting of 0 for this field disables the modulo feature.

Table 284 shows how the transfer addresses are specified based on the setting of the MOD field. Here a circular buffer is created where the address wraps to the original value while the 28 upper address bits (0x1234567x) retain their original value. In this example the source address is set to 0x12345670, the offset is set to 4 bytes, and the MOD field is set to 4, allowing for a 16-byte size queue.

Table 284. Modulo feature example

Transfer number	Address
1	0x12345670
2	0x12345674
3	0x12345678
4	0x1234567C
5	0x12345670
6	0x12345674

20.5.5 eDMA TCD*n* status monitoring

This section discusses how to monitor eDMA status.

20.5.5.1 Minor loop complete

There are two methods to test for minor loop completion when using software initiated service requests. The first is to read the TCD*n*_CITER field and test for a change. Another method may be extracted from the sequence shown below. The second method is to test the TCD*n*_CSR[START] bit and the TCD*n*_CSR[ACTIVE] bit. The minor-loop-complete condition is indicated by both bits reading zero after the TCD*n*_CSR[START] was set. Polling the TCD*n*_CSR[ACTIVE] bit may be inconclusive, because the active status may be missed if the channel execution is short in duration.

The TCD status bits execute a sequence for a software activated channel as shown in the following table:

Table 285. TCD status sequence – software activated channel

Step	TCD*n*_CSR bits			State
	START	ACTIVE	DONE	
1	1	0	0	Channel service request via software
2	0	1	0	Channel is executing

Table 285. TCD status sequence – software activated channel (continued)

Step	TCD*n*_CSR bits			State
	START	ACTIVE	DONE	
3a	0	0	0	Channel has completed the minor loop and is idle
3b	0	0	1	Channel has completed the major loop and is idle

The best method to test for minor-loop completion when using hardware (peripheral) initiated service requests is to read the TCD*n*_CITER field and test for a change. The hardware request and acknowledge handshake signals are not visible in the programmer's model.

The TCD status bits execute a sequence for a hardware-activated channel as shown in the following table:

Table 286. TCD status sequence – hardware activated channel

Step	TCD*n*_CSR bits			State
	START	ACTIVE	DONE	
1	0	0	0	Channel service request via hardware (peripheral request asserted)
2	0	1	0	Channel is executing
3a	0	0	0	Channel has completed the minor loop and is idle
3b	0	0	1	Channel has completed the major loop and is idle

For both activation types, the major-loop-complete status is explicitly indicated via the TCD*n*_CSR[DONE] bit. The TCD*n*_CSR[START] bit is cleared automatically when the channel begins execution regardless of how the channel activates.

20.5.5.2 Active channel TCD*n* reads

The eDMA reads back the true TCD*n*_SADDR, TCD*n*_DADDR, and TCD*n*_NBYTES values if read while a channel executes. The true values of the SADDR, DADDR, and NBYTES are the values the eDMA engine currently uses in its internal register file and not the values in the TCD local memory for that channel. The addresses (SADDR and DADDR) and NBYTES (decrements to zero as the transfer progresses) can give an indication of the progress of the transfer. All other values are read back from the TCD local memory.

20.5.5.3 Preemption status

Preemption is available only when fixed arbitration is selected for both group and channel arbitration modes. A preemptive situation is one in which a preempt-enabled channel runs and a higher priority request becomes active. When the eDMA engine is not operating in Fixed-Group, Fixed-Channel Arbitration mode, the determination of the actively running relative priority outstanding requests become undefined. Channel and/or group priorities are treated as equal (constantly rotating) when Round-Robin Arbitration mode is selected.

The TCD*n*_CSR[ACTIVE] bit for the preempted channel remains asserted throughout the preemption. The preempted channel is temporarily suspended while the preempting

channel executes one major loop iteration. If two TCD*n*_CSR[ACTIVE] bits are set simultaneously in the global TCD map, a higher priority channel is actively preempting a lower priority channel.

20.5.6 Channel linking

Channel linking (or chaining) is a mechanism where one channel sets the TCD*n*_CSR[START] bit of another channel (or itself), therefore initiating a service request for that channel. When properly enabled, the EDMA engine automatically performs this operation at the major or minor loop completion.

The minor loop channel linking occurs at the completion of the minor loop (or one iteration of the major loop). The TCD*n*_CITER[E_LINK] field determines whether a minor loop link is requested. When enabled, the channel link is made after each iteration of the major loop except for the last. When the major loop is exhausted, only the major loop channel link fields are used to determine if a channel link should be made. For example, the initial fields of:

```
TCDn_CITER[E_LINK] = 1
TCDn_CITER[LINKCH] = 0xC
TCDn_CITER[CITER] value = 0x4
TCDn_CSR[MAJOR_E_LINK] = 1
TCDn_CSR[MAJOR_LINKCH] = 0x7
```

executes as:

1. Minor loop done → set TCD12_CSR[START] bit
2. Minor loop done → set TCD12_CSR[START] bit
3. Minor loop done → set TCD12_CSR[START] bit
4. Minor loop done, major loop done → set TCD7_CSR[START] bit

When minor loop linking is enabled (TCD*n*_CITER[E_LINK] = 1), the TCD*n*_CITER[CITER] field uses a 9-bit vector to form the current iteration count. When minor loop linking is disabled (TCD*n*_CITER[E_LINK] = 0), the TCD*n*_CITER[CITER] field uses a 15-bit vector to form the current iteration count. The bits associated with the TCD*n*_CITER[LINKCH] field are concatenated onto the CITER value to increase the range of the CITER.

Note: The TCDn_CITER[E_LINK] bit and the TCDn_BITER[E_LINK] bit must equal or a configuration error is reported. The CITER and BITER vector widths must be equal to calculate the major loop, half-way done interrupt point.

Table 287 summarizes how a DMA channel can link to another DMA channel (that is, how it can use another channel's TCD) at the end of a loop.

Table 287. Channel linking parameters

Desired link behavior	TCD control field name	Description
Link at end of minor loop	CITER[E_LINK]	Enable channel-to-channel linking on minor loop completion (current iteration)
	CITER[LINKCH] Link channel number when linking at end of minor loop (current iteration)	

Table 287. Channel linking parameters (continued)

Desired link behavior	TCD control field name	Description
Link at end of major loop	CSR[MAJOR_E_LINK]	Enable channel-to-channel linking on major loop completion
	CSR[MAJOR_LINKCH] Link channel number when linking at end of major loop	

20.5.7 Dynamic programming

20.5.7.1 Dynamic priority changing

The following two options are recommended for dynamically changing channel priority levels:

1. Switch to Round-Robin Channel Arbitration mode, change the channel priorities, then switch back to Fixed Arbitration mode
2. Disable all the channels within a group, then change the channel priorities within that group only, then enable the appropriate channels

The following two options are available for dynamically changing group priority levels:

1. Switch to Round-Robin Group Arbitration mode, change the group priorities, then switch back to Fixed Arbitration mode
2. Disable all channels, change the group priorities, then enable the appropriate channels

20.5.7.2 Dynamic channel linking

Dynamic channel linking is the process of setting the TCD.major.e_link bit during channel execution. This bit is read from the TCD local memory at the end of channel execution, thus allowing the user to enable the feature during channel execution.

Because the user is allowed to change the configuration during execution, a coherency model is needed. Consider the scenario where the user attempts to execute a dynamic channel link by enabling the TCD.major.e_link bit at the same time the eDMA engine is retiring the channel. The TCD.major.e_link would be set in the programmer's model, but it would be unclear whether the actual link was made before the channel retired.

The coherency model in *Table 288* is recommended when executing a dynamic channel link request.

Table 288. Coherency model for a dynamic channel link request

Step	Action
1	Write 1b to the TCD.major.e_link bit
2	Read back the TCD.major.e_link bit
3	Test the TCD.major.e_link request status: – If TCD.major.e_link = 1b, the dynamic link attempt was successful – If TCD.major.e_link = 0b, the attempted dynamic link did not succeed (the channel was already retiring)

For this request, the TCD local memory controller forces the TCD.major.e_link bit to zero on any writes to a channel's TCD.word7 after that channel's TCD.done bit is set, indicating the major loop is complete.

Note: The user must clear the TCD.done bit before writing the TCD.major.e_link bit. The TCD.done bit is cleared automatically by the eDMA engine after a channel begins execution.

20.5.7.3 Dynamic scatter/gather

Dynamic scatter/gather is the process of setting the TCD.e_sg bit during channel execution. This bit is read from the TCD local memory at the end of channel execution, thus allowing the user to enable the feature during channel execution.

Because the user is allowed to change the configuration during execution, a coherency model is needed. Consider the scenario where the user attempts to execute a dynamic scatter/gather operation by enabling the TCD.e_sg bit at the same time the eDMA engine is retiring the channel. The TCD.e_sg would be set in the programmer's model, but it would be unclear whether the actual scatter/gather request was honored before the channel retired.

Two methods for this coherency model are shown in the following subsections. Method 1 has the advantage of reading the major.linkch field and the e_sg bit with a single read. For both dynamic channel linking and scatter/gather requests, the TCD local memory controller forces the TCD.major.e_link and TCD.e_sg bits to zero on any writes to a channel's TCD.word7 if that channel's TCD.done bit is set indicating the major loop is complete.

Note: The user must clear the TCD.done bit before writing the TCD.major.e_link or TCD.e_sg bits. The TCD.done bit is cleared automatically by the eDMA engine after a channel begins execution.

20.5.7.3.1 Method 1 (channel not using major loop channel linking)

For a channel not using major loop channel linking, the coherency model in *Table 289* may be used for a dynamic scatter/gather request.

When the TCD.major.e_link bit is zero, the TCD.major.linkch field is not used by the eDMA. In this case, the TCD.major.linkch bits may be used for other purposes. This method uses the TCD.major.linkch field as a TCD identification (ID).

Table 289. Coherency model for method 1

Step	Action
1	When the descriptors are built, write a unique TCD ID in the TCD.major.linkch field for each TCD associated with a channel using dynamic scatter/gather
2	Write 1b to theTCD.d_req bit *Note: Should a dynamic scatter/gather attempt fail, setting the d_req bit will prevent a future hardware activation of this channel. This stops the channel from executing with a destination address (daddr) that was calculated using a scatter/gather address (written in the next step) instead of a dlast final offset value.*
3	Write theTCD.dlast_sga field with the scatter/gather address
4	Write 1b to the TCD.e_sg bit

Table 289. Coherency model for method 1 (continued)

Step	Action
5	Read back the 16 bit TCD control/status field
6	Test the TCD.e_sg request status and TCD.major.linkch value: – If e_sg = 1b, the dynamic link attempt was successful – If e_sg = 0b and the major.linkch (ID) did not change, the attempted dynamic link did not succeed (the channel was already retiring) – If e_sg = 0b and the major.linkch (ID) changed, the dynamic link attempt was successful (the new TCD's e_sg value cleared the e_sg bit)

20.5.7.3.2 Method 2 (channel using major loop linking)

For a channel using major loop channel linking, the coherency model in *Table 290* may be used for a dynamic scatter/gather request. This method uses the TCD.dlast_sga field as a TCD identification (ID).

Table 290. Coherency model for method 2

Step	Action
1	Write 1b to theTCD.d_req bit **Note:** Should a dynamic scatter/gather attempt fail, setting the d_req bit will prevent a future hardware activation of this channel. This stops the channel from executing with a destination address (daddr) that was calculated using a scatter/gather address (written in the next step) instead of a dlast final offset value.
2	Write theTCD.dlast_sga field with the scatter/gather address
3	Write 1b to the TCD.e_sg bit
4	Read back the TCD.e_sg bit
5	Test the TCD.e_sg request status: – If e_sg = 1b, the dynamic link attempt was successful – If e_sg = 0b, read the 32 bit TCD dlast_sga field – If e_sg = 0b and the dlast_sga did not change, the attempted dynamic link did not succeed (the channel was already retiring) – If e_sg = 0b and the dlast_sga changed, the dynamic link attempt was successful (the new TCD's e_sg value cleared the e_sg bit)

21 DMA channel multiplexer (DMACHMUX)

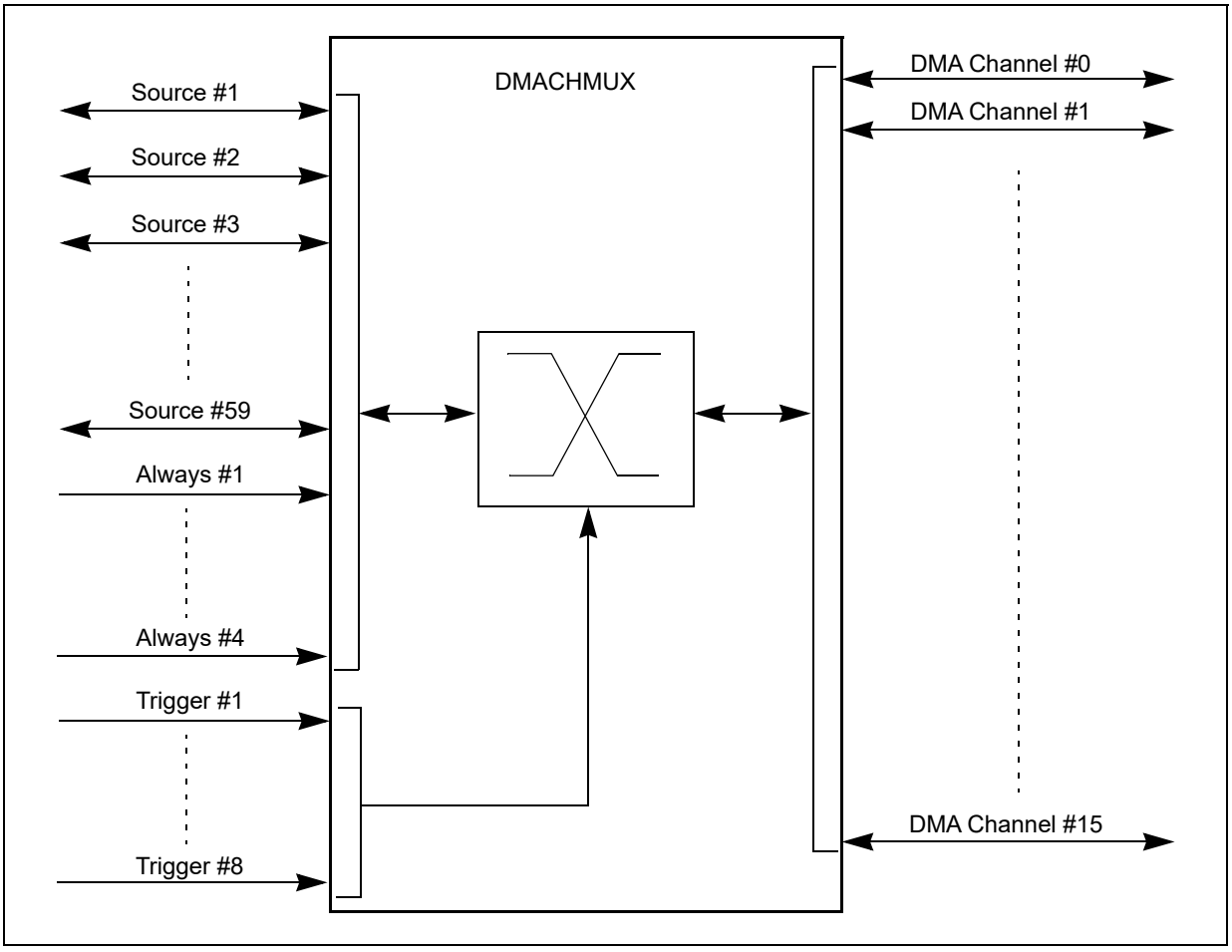
21.1 Introduction

Note: For the chip-specific implementation details of this module's instances, see the chip configuration information.

21.1.1 Overview

The DMA channel multiplexer (DMACHMUX) allows to route 59 DMA peripheral sources (called slots) to 16 DMA channels. This is illustrated in *Figure 192*.

Figure 192. DMACHMUX block diagram



21.1.2 Features

The DMACHMUX provides these features.

- 59 peripheral slots + four always-on slots can be routed to 16 channels.
- 16 independently selectable DMA channels routers.
  - The first 8 channels additionally provide a trigger functionality.
- Each channel router can be assigned to one of 59 possible peripheral DMA slots or to one of the 4 always-on slots.

21.1.3 Modes of operation

The following operation modes are available.

- Disabled Mode
  In this mode, the DMA channel is disabled. Since disabling and enabling of DMA channels is done primarily via the DMA configuration registers, this mode is used mainly as the reset state for a DMA channel in the DMACHMUX. It may also be used to temporarily suspend a DMA channel while reconfiguration of the system takes place (for example, changing the period of a DMA trigger).
- Normal Mode
  In this mode, a DMA source (such as DSPI transmit or DSPI receive for example) is routed directly to the specified DMA channel. The operation of the DMACHMUX in this mode is completely transparent to the system.
- Periodic Trigger Mode
  In this mode, a DMA source may only request a DMA transfer (such as when a transmit buffer becomes empty or a receive buffer becomes full) periodically. Configuration of the period is done in the registers of the Periodic Interrupt Timer (PIT). This mode is only available for channels 0–7 of each DMACHMUX instance.

21.2 External signal description

21.2.1 Overview

The DMACHMUX has no external pins.

21.3 Memory map and register definition

Table 291 lists the DMACHMUX memory-mapped registers. All addresses are offsets; the absolute address can be computed by adding the specified offset to the base address of the DMACHMUX.

Table 291. DMACHMUX memory map

Address offset	Register	Access	Location
eDMA channel registers per channel (*n* = 0 to 15)			
0x0000 + *n* * 0x1	Channel *n* Configuration registers (CHCFG*n*)	R/W	*Section 21.3.1.1*

All registers are accessible via 8-bit, 16-bit, or 32-bit accesses. However, 16-bit accesses must be aligned to 16-bit boundaries, and 32-bit accesses must be aligned to 32-bit boundaries. As an example, CHCFG0 through CHCFG3 are accessible by a 32-bit READ/WRITE to address ‘Base + 0x00’, but performing a 32-bit access to address ‘Base + 0x01’ is illegal.

21.3.1 Register descriptions

This section provides a detailed description of the memory-mapped registers in the DMACHMUX.

21.3.1.1 Channel *n* Configuration registers (CHCFG*n*)

Each of the DMA channels can be independently enabled/disabled and associated with one of the DMACHMUX slots (peripheral slots or always-on slots) in the system.

Address: 0x0000 + *n* * 0x1 (*n* = 0 to 15) Access: User read/write

	0	1	2	3	4	5	6	7
R W	ENBL	TRIG	SOURCE					
Reset	0	0	0	0	0	0	0	0

Figure 193. Channel Configuration registers (CHCFG*n*)

Table 292. CHCFG*n* field descriptions

Field	Description
0 ENBL	DMA Channel Enable ENBL enables the DMA Channel. 0 DMA channel is disabled. This mode is primarily used during configuration of the DMACHMUX. The DMA has separate channel enables/disables, which should be used to disable or reconfigure a DMA channel 1 DMA channel is enabled
1 TRIG	DMA Channel Trigger Enable (for triggered channels only) TRIG enables the periodic trigger capability for the DMA Channel. 0 Triggering is disabled. If triggering is disabled, and the ENBL bit is set, the DMA Channel will simply route the specified source to the DMA channel 1 Triggering is enabled
2:7 SOURCE	DMA Channel Source (DMACHMUX slot) SOURCE specifies which DMA source, if any, is routed to a particular DMA channel. Please refer to “Device configuration” chapter for further details about the peripherals and their slot numbers

Table 293. Channel and trigger enabling

ENBL	TRIG	Function	Mode
0	X	DMA Channel is disabled	Disabled Mode
1	0	DMA Channel is enabled with no triggering (transparent)	Normal Mode
1	1	DMA Channel is enabled with triggering	Periodic Trigger Mode

Note: Setting multiple CHCFG registers with the same source value will result in unpredictable behavior.

Before changing the trigger or source settings a DMA channel must be disabled via the CHCFG[n].ENBL bit.

21.4 Functional description

This section provides a functional description of the DMACHMUX. The primary purpose of the DMACHMUX is to provide flexibility in the system's use of the available DMA channels. As such, configuration of the DMACHMUX is intended to be a static procedure done during execution of the system boot code. However, if the procedure outlined in *Section 21.5.2* is followed, the configuration of the DMACHMUX may be changed during the normal operation of the system.

Functionally, the DMACHMUX channels may be divided into two classes.

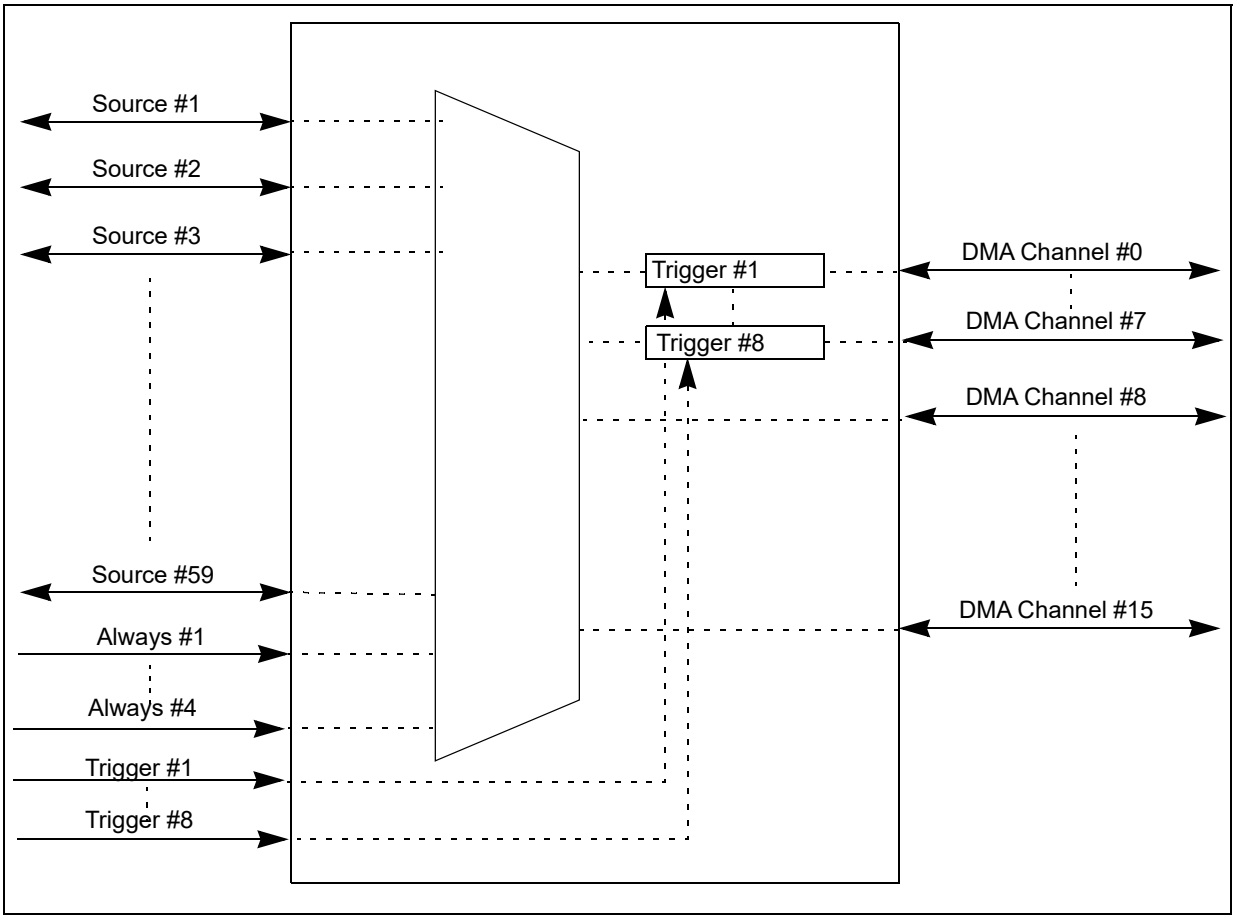
- Channels which implement the normal routing functionality plus periodic triggering capability.
- Channels which implement only the normal routing functionality.

21.4.1 DMA channels with periodic triggering capability

Besides the normal routing functionality, the first 8 channels of the DMACHMUX provide a special periodic triggering capability that can be used to provide an automatic mechanism to transmit bytes, frames or packets at fixed intervals without the need for processor intervention. The trigger is generated by the Periodic Interrupt Timer (PIT): such as the configuration of the periodic triggering interval is done via configuration registers in the PIT. Please refer to the Periodic Interrupt Timer (PIT) chapter for more information on this topic.

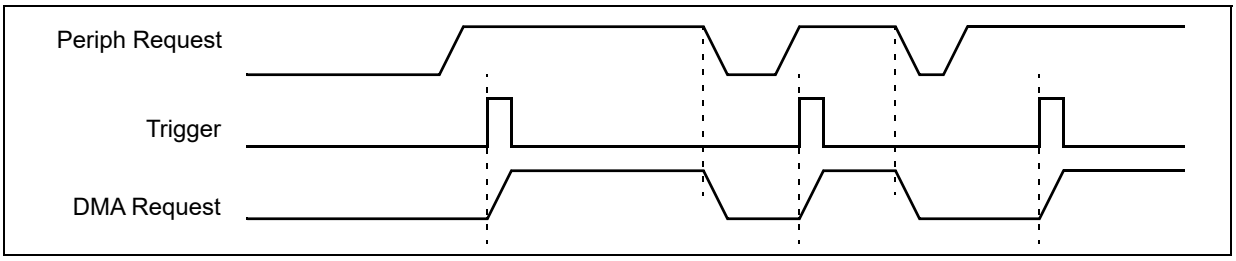
Note: Because of the dynamic nature of the system (such as DMA channel priorities, bus arbitration, interrupt service routine lengths), the number of clock cycles between a trigger and the actual DMA transfer cannot be guaranteed.

Figure 194. DMACHMUX triggered channels



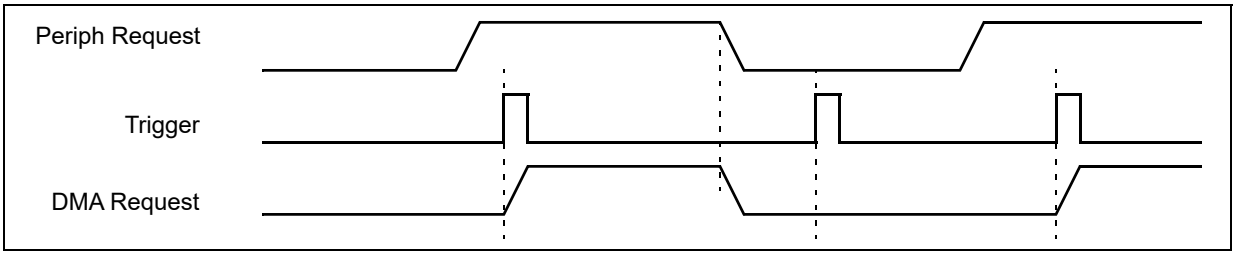
The DMA channel triggering capability allows the system to schedule regular DMA transfers, usually on the transmit side of certain peripherals, without the intervention of the processor. This trigger works by gating the request from the Peripheral to the DMA until a trigger event has been seen. This is illustrated in *Figure 195*.

Figure 195. DMACHMUX channel triggering: normal operation



Once the DMA request has been serviced, the peripheral will negate its request, effectively resetting the gating mechanism until the peripheral re-asserts its request AND the next trigger event is seen. This means that if a trigger is seen, but the peripheral is not requesting a transfer, that trigger will be ignored. This situation is illustrated in *Figure 196*.

Figure 196. DMACHMUX channel triggering: ignored trigger



This triggering capability may be used with any peripheral that supports DMA transfers, and is most useful for two types of situations.

- Periodically polling external devices on a particular bus. As an example, the transmit side of an SPI is assigned to a DMA channel with a trigger, as described above. Once setup, the SPI will request DMA transfers (presumably from memory) as long as its transmit buffer is empty. By using a trigger on this channel, the SPI transfers can be automatically performed every 5 µs (as an example). On the receive side of the SPI, the SPI and DMA can be configured to transfer receive data into memory, effectively implementing a method to periodically read data from external devices and transfer the results into memory without processor intervention.
- Using the GPIO Ports to drive or sample waveforms. By configuring the DMA to transfer data to one or more GPIO ports, it is possible to create complex waveforms using tabular data stored in on-chip memory. Conversely, using the DMA to periodically transfer data from one or more GPIO ports, it is possible to sample complex waveforms and store the results in tabular form in on-chip memory.

A more detailed description of the capability of each trigger (such as resolution, range of values) may be found in Periodic Interrupt Timer (PIT) chapter.

21.4.2 DMA channels with no triggering capability

The other channels of the DMACHMUX provide the normal routing functionality as described in *Section 21.1.3*.

21.4.3 “Always Enabled” DMA sources

In addition to the peripherals that can be used as DMA sources, there are 4 additional DMA sources that are “always enabled”. Unlike the peripheral DMA sources, where the peripheral controls the flow of data during DMA transfers, the *“always enabled”* sources provide no such “throttling” of the data transfers. These sources are most useful in the following cases:

- Doing DMA transfers to/from GPIO: Moving data from/to one or more GPIO pins, either unthrottled (that is, as fast as possible), or periodically (using the DMA triggering capability).
- Doing DMA transfers from memory to memory: Moving data from memory to memory, typically as fast as possible, sometimes with software activation.
- Doing DMA transfers from memory to the external bus (or vice-versa): Similar to memory to memory transfers, this is typically done as quickly as possible.
- Any DMA transfer that requires software activation: Any DMA transfer that should be explicitly started by software.

In cases where software should initiate the start of a DMA transfer, a *“always enabled”* DMA source can be used to provide maximum flexibility. When activating a DMA channel via software, subsequent executions of the minor loop require a new “*start*” event be sent. This

can either be a new software activation, or a transfer request from the DMACHMUX. The options for doing this are:

- Transfer all data in a single minor loop. By configuring the DMA to transfer all of the data in a single minor loop (that is major loop counter = 1), no reactivation of the channel is necessary. The disadvantage to this option is the reduced granularity in determining the load that the DMA transfer will incur on the system. For this option, the DMA channel should be disabled in the DMACHMUX.
- Use explicit software reactivation. In this option, the DMA is configured to transfer the data using both minor and major loops, but the processor is required to reactivate the channel (by writing to the DMA registers) *after every minor loop*. For this option, the DMA channel should be disabled in the DMACHMUX.
- Use a *"always enabled"* DMA source. In this option, the DMA is configured to transfer the data using both minor and major loops, and the DMACHMUX does the channel reactivation. For this option, the DMA channel should be enabled and pointing to an *"always enabled"* source. Note that the reactivation of the channel can be continuous (DMA triggering is disabled) or can use the DMA triggering capability. In this manner, it is possible to execute periodic transfers of packets of data from one source to another, without processor intervention.

21.5 Initialization/Application information

21.5.1 Reset

The reset state of each individual bit is shown within the Register Description section (See *Section 21.3.1*). In summary, after reset, all channels are disabled and must be explicitly enabled before use.

21.5.2 Enabling and configuring sources

To enable a source with periodic triggering:

1. Determine with which DMA channel the source will be associated. Note that only the first 2 DMA channels have periodic triggering capability.
2. Clear the CHCFG[ENBL] and CHCFG[TRIG] fields of the DMA channel.
3. Ensure that the DMA channel is properly configured in the DMA. The DMA channel may be enabled at this point.
4. Configure the corresponding timer.
5. Select the source to be routed to the DMA channel. Write to the corresponding CHCFG register, ensuring that the CHCFG[ENBL] and CHCFG[TRIG] fields are set.

Note: The following is an example. See the chip configuration details for the number of this device's DMA channels that have triggering capability.

Example 1 Configure source #5 Transmit for use with DMA Channel 2, with periodic triggering capability:

1. Write 0x00 to CHCFG2 (Base Address + 0x02).
2. Configure Channel 2 in the DMA, including enabling the channel.
3. Configure a timer for the desired trigger interval.
4. Write 0xC5 to CHCFG2 (Base Address + 0x02).

The following code example illustrates steps #1 and #4 above:

In file **registers.h:**

```
#define DMAMUX_BASE_ADDR     0xFC084000/* Example only ! */
/* Following example assumes char is 8-bits */
volatile unsigned char *CHCFG0 = (volatile unsigned char *)
(DMAMUX_BASE_ADDR+0x0000);
volatile unsigned char *CHCFG1 = (volatile unsigned char *)
(DMAMUX_BASE_ADDR+0x0001);
volatile unsigned char *CHCFG2 = (volatile unsigned char *)
(DMAMUX_BASE_ADDR+0x0002);
volatile unsigned char *CHCFG3 = (volatile unsigned char *)
(DMAMUX_BASE_ADDR+0x0003);
volatile unsigned char *CHCFG4 = (volatile unsigned char *)
(DMAMUX_BASE_ADDR+0x0004);
volatile unsigned char *CHCFG5 = (volatile unsigned char *)
(DMAMUX_BASE_ADDR+0x0005);
volatile unsigned char *CHCFG6 = (volatile unsigned char *)
(DMAMUX_BASE_ADDR+0x0006);
volatile unsigned char *CHCFG7 = (volatile unsigned char *)
(DMAMUX_BASE_ADDR+0x0007);
volatile unsigned char *CHCFG8 = (volatile unsigned char *)
(DMAMUX_BASE_ADDR+0x0008);
volatile unsigned char *CHCFG9 = (volatile unsigned char *)
(DMAMUX_BASE_ADDR+0x0009);
volatile unsigned char *CHCFG10= (volatile unsigned char *)
(DMAMUX_BASE_ADDR+0x000A);
volatile unsigned char *CHCFG11= (volatile unsigned char *)
(DMAMUX_BASE_ADDR+0x000B);
volatile unsigned char *CHCFG12= (volatile unsigned char *)
(DMAMUX_BASE_ADDR+0x000C);
volatile unsigned char *CHCFG13= (volatile unsigned char *)
(DMAMUX_BASE_ADDR+0x000D);
volatile unsigned char *CHCFG14= (volatile unsigned char *)
(DMAMUX_BASE_ADDR+0x000E);
volatile unsigned char *CHCFG15= (volatile unsigned char *)
(DMAMUX_BASE_ADDR+0x000F);
```

In File **main.c:**

```
#include "registers.h"
   :
   :
*CHCFG2 = 0x00;
*CHCFG2 = 0xC5;
```

To enable a source without periodic triggering:

1. Determine with which DMA channel the source will be associated. Note that only the first 8 DMA channels have periodic triggering capability.
2. Clear the CHCFG[ENBL] and CHCFG[TRIG] fields of the DMA channel.
3. Ensure that the DMA channel is properly configured in the DMA. The DMA channel may be enabled at this point.
4. Select the source to be routed to the DMA channel. Write to the corresponding CHCFG register, ensuring that CHCFG[ENBL] is set while CHCFG[TRIG] is cleared.

Note: The following is an example. See the chip configuration details for the number of this device's DMA channels that have triggering capability.

Example 2 Configure source #5 Transmit for use with DMA Channel 2, with no periodic triggering capability:

1. Write 0x00 to CHCFG2 (Base Address + 0x02).
2. Configure Channel 2 in the DMA, including enabling the channel.
3. Write 0x85 to CHCFG2 (Base Address + 0x02).

The following code example illustrates steps #1 and #3 above:

In File **registers.h:**

```
  #define DMAMUX_BASE_ADDR     0xFC084000/* Example only ! */
  /* Following example assumes char is 8-bits */
  volatile unsigned char *CHCFG0 = (volatile unsigned char *)
(DMAMUX_BASE_ADDR+0x0000);
  volatile unsigned char *CHCFG1 = (volatile unsigned char *)
(DMAMUX_BASE_ADDR+0x0001);
  volatile unsigned char *CHCFG2 = (volatile unsigned char *)
(DMAMUX_BASE_ADDR+0x0002);
  volatile unsigned char *CHCFG3 = (volatile unsigned char *)
(DMAMUX_BASE_ADDR+0x0003);
  volatile unsigned char *CHCFG4 = (volatile unsigned char *)
(DMAMUX_BASE_ADDR+0x0004);
  volatile unsigned char *CHCFG5 = (volatile unsigned char *)
(DMAMUX_BASE_ADDR+0x0005);
  volatile unsigned char *CHCFG6 = (volatile unsigned char *)
(DMAMUX_BASE_ADDR+0x0006);
  volatile unsigned char *CHCFG7 = (volatile unsigned char *)
(DMAMUX_BASE_ADDR+0x0007);
  volatile unsigned char *CHCFG8 = (volatile unsigned char *)
(DMAMUX_BASE_ADDR+0x0008);
  volatile unsigned char *CHCFG9 = (volatile unsigned char *)
(DMAMUX_BASE_ADDR+0x0009);
  volatile unsigned char *CHCFG10= (volatile unsigned char *)
(DMAMUX_BASE_ADDR+0x000A);
  volatile unsigned char *CHCFG11= (volatile unsigned char *)
(DMAMUX_BASE_ADDR+0x000B);
  volatile unsigned char *CHCFG12= (volatile unsigned char *)
(DMAMUX_BASE_ADDR+0x000C);
```

```
  volatile unsigned char *CHCFG13= (volatile unsigned char *)
(DMAMUX_BASE_ADDR+0x000D);
  volatile unsigned char *CHCFG14= (volatile unsigned char *)
(DMAMUX_BASE_ADDR+0x000E);
  volatile unsigned char *CHCFG15= (volatile unsigned char *)
(DMAMUX_BASE_ADDR+0x000F);
```

In File **main.c:**

```
  #include "registers.h"
     :
     :
  *CHCFG2 = 0x00;
  *CHCFG2 = 0x85;
```

Disabling a source:

A particular DMA source may be disabled by not writing the corresponding source value into any of the CHCFG registers. Additionally, some module specific configuration may be necessary. Please refer to the appropriate section for more details.

Switching the source of a DMA Channel:

1. Disable the DMA channel in the DMA and reconfigure the channel for the new source.
2. Clear the ENBL and TRIG bits of the DMA channel.
3. Select the source to be routed to the DMA channel. Write to the corresponding CHCFG register, ensuring that the ENBL and TRIG bits are set.

Example 3 Switch DMA Channel 8 from source #5 transmit to source #7 transmit:

1. In the DMA configuration registers, disable DMA channel 8 and reconfigure it to handle the transfers to peripheral slot 7. This example assumes channel 8 does not have triggering capability.
2. Write 0x00 to CHCFG8 (Base Address + 0x08).
3. Write 0x87 to CHCFG8 (Base Address + 0x08). (In this example, setting the TRIG bit would have no effect, due to the assumption that channels 8 does not support the periodic triggering functionality).

The following code example illustrates steps #2 and #3 above.

In File **registers.h:**

```
  #define DMAMUX_BASE_ADDR     0xFC084000/* Example only ! */
  /* Following example assumes char is 8-bits */
  volatile unsigned char *CHCFG0 = (volatile unsigned char *)
(DMAMUX_BASE_ADDR+0x0000);
  volatile unsigned char *CHCFG1 = (volatile unsigned char *)
(DMAMUX_BASE_ADDR+0x0001);
  volatile unsigned char *CHCFG2 = (volatile unsigned char *)
(DMAMUX_BASE_ADDR+0x0002);
  volatile unsigned char *CHCFG3 = (volatile unsigned char *)
(DMAMUX_BASE_ADDR+0x0003);
  volatile unsigned char *CHCFG4 = (volatile unsigned char *)
(DMAMUX_BASE_ADDR+0x0004);
```

```
  volatile unsigned char *CHCFG5 = (volatile unsigned char *)
(DMAMUX_BASE_ADDR+0x0005);
  volatile unsigned char *CHCFG6 = (volatile unsigned char *)
(DMAMUX_BASE_ADDR+0x0006);
  volatile unsigned char *CHCFG7 = (volatile unsigned char *)
(DMAMUX_BASE_ADDR+0x0007);
  volatile unsigned char *CHCFG8 = (volatile unsigned char *)
(DMAMUX_BASE_ADDR+0x0008);
  volatile unsigned char *CHCFG9 = (volatile unsigned char *)
(DMAMUX_BASE_ADDR+0x0009);
  volatile unsigned char *CHCFG10= (volatile unsigned char *)
(DMAMUX_BASE_ADDR+0x000A);
  volatile unsigned char *CHCFG11= (volatile unsigned char *)
(DMAMUX_BASE_ADDR+0x000B);
  volatile unsigned char *CHCFG12= (volatile unsigned char *)
(DMAMUX_BASE_ADDR+0x000C);
  volatile unsigned char *CHCFG13= (volatile unsigned char *)
(DMAMUX_BASE_ADDR+0x000D);
  volatile unsigned char *CHCFG14= (volatile unsigned char *)
(DMAMUX_BASE_ADDR+0x000E);
  volatile unsigned char *CHCFG15= (volatile unsigned char *)
(DMAMUX_BASE_ADDR+0x000F);
```

In File **main.c:**

```
  #include "registers.h"
     :
     :
  *CHCFG8 = 0x00;
  *CHCFG8 = 0x87;
```

22 Wakeup unit (WKPU)

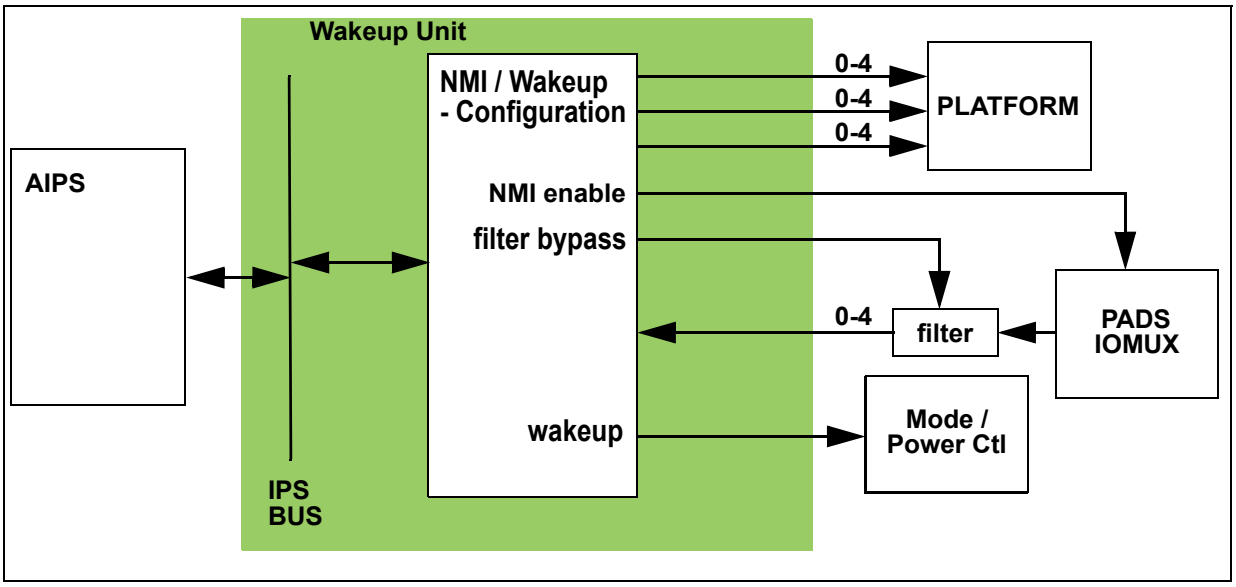
22.1 Introduction

22.1.1 Overview

The Wakeup unit supports up to 3 external sources that can cause non-maskable interrupt requests or wakeup events. In addition, it combines its wakeup events to supply a single wakeup to the system. This NMI external source can also generate a functional reset event to the MC_RGM. *Figure 197* is a block diagram of the Wakeup unit and its interfaces to other system components.

Note: For the chip-specific implementation details of this module see the "Device Configuration" chapter.

Figure 197. Wakeup unit Block Diagram



22.1.2 Features

The Wakeup unit supports these distinctive features:

Non-maskable Interrupt support with:

- Up to four NMI sources
- Up to four analog glitch filters
- Independent CPU exceptions destination: non-maskable interrupt, critical interrupt, or machine check request
- Edge detection
- Configurable system wakeup triggering from all NMI sources

22.2 Memory Map and Register Description

This section provides a detailed description of all registers accessible in the WKPU module.

22.2.1 Memory Map

Table 294 gives an overview on the WKPU registers implemented.

Table 294. WKPU Memory Map

Address Offset	Registers	Supported Access Sizes	Location
0x0000	NMI Status Flag Register (NSR)	32/16/8	*Section 22.2.2.1*
0x0004–0x0007	Reserved		
0x0008	NMI Configuration Register (NCR)	32/16/8	*Section 22.2.2.2*
0x000C–0x3FFF	Reserved		

Note: Reserved registers will read as 0, writes will have no effect. If supported and enabled by the SoC, a transfer error will be issued when trying to access completely reserved register space.

22.2.2 Register Descriptions

This section describes in address order all the Wakeup unit registers.

22.2.2.1 NMI Status Flag Register (NSR)

This register holds the non-maskable interrupt status flags.

Offset: 0x0000 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	NIF0	NOVF0	0	0	0	0	0	0	NIF1	NOVF1	0	0	0	0	0	0	NIF2	NOVF2	0	0	0	0	0	0	RIF	ROVF	0	0	0	0	0	0
W	w1c	w1c							w1c	w1c							w1c	w1c							w1c	w1c						
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 198. NMI Status Flag Register (NSR)

Table 295. NSR Field Descriptions

Field	Description
0 NIF0	NMI Status Flag 0 This flag can be cleared only by writing a 1. Writing a 0 has no effect. If enabled (NREE0 or NFEE0 set), NIF0 causes an exception to the associated CPU, as configured within the NDSS bit of the NCR register. In addition this event can also be configured to trigger a system wakeup. 1 An event as defined by NREE0 and NFEE0 has occurred. 0 No event has occurred on the pad.
1 NOVF0	NMI Overrun Status Flag 0 This flag can be cleared only by writing a 1. Writing a 0 has no effect. It will be a copy of the current NIF0 value whenever a NMI event occurs, thereby indicating to the software that a NMI occurred while the last one was not yet serviced. If enabled (NREE0 or NFEE0 set), NOVF0 causes an interrupt request. 1 An overrun has occurred on NMI input 0. 0 No overrun has occurred on NMI input 0.
8 NIF1	NMI Status Flag 1 This flag can be cleared only by writing a 1. Writing a 0 has no effect. If enabled (NREE1 or NFEE1 set), NIF1 causes an exception to the associated CPU, as configured within the NDSS bit of the NCR register. In addition this event can also be configured to trigger a system wakeup. 1 An event as defined by NREE1 and NFEE1 has occurred. 0 No event has occurred on the pad.
9 NOVF1	NMI Overrun Status Flag 1 This flag can be cleared only by writing a 1. Writing a 0 has no effect. It will be a copy of the current NIF1 value whenever a NMI event occurs, thereby indicating to the software that a NMI occurred while the last one was not yet serviced. If enabled (NREE1 or NFEE1 set), NOVF1 causes an interrupt request. 1 An overrun has occurred on NMI input 1. 0 No overrun has occurred on NMI input 1.
24 RIF	Reset status Flag This flag can be cleared only by writing a 1. Writing a 0 has no effect. If enabled (that is if RREE or RFEE is set), RIF status bit is set and will cause a functional reset request to RGM if RDSS[1:0] of NCR is ‘00’. This bit is automatically cleared if the F_WAKEUP reset request is configured in the MC_RGM to generate a long or short “functional” reset. 1 An event as defined by RREE and RFEE has occurred. 0 No event has occurred on the pad.
25 ROVF	Reset overrun status flag This flag can be cleared only by writing a 1. Writing a 0 has no effect. It will be a copy of the current RIF value whenever a NMI event occurs, thereby indicating to the software that an NMI occurred while the last one was not yet serviced. If enabled (RREE or RFEE set), ROVF can cause a reset request to MC_RGM if RDSS[1:0] of NCR is ‘00’. This bit is automatically cleared if the F_WAKEUP reset request is configured in the MC_RGM to generate a long or short “functional” reset. 1 An overrun has occurred on NMI input. 0 No overrun has occurred on NMI input.

Note: Refer to the Chapter 7: Device configuration to see how the Figure 198: NMI Status Flag Register (NSR) is used in the device.

22.2.2.2 NMI Configuration Register (NCR)

This register holds the configuration bits for the non-maskable interrupt settings.

Offset: 0x0008 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	NLOCK0	NDSS0		NWRE0	0	NREE0	NFEE0	NFE0	NLOCK1	NDSS1		NWRE1	0	NREE1	NFEE1	NFE1	NLOCK2	NDSS2		NWRE2	0	NREE2	NFEE2	NFE2	RLOCK	RDSS		RWRE	0	RREE	RFEE	RFE(1)
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0

Figure 199. NMI Configuration Register (NCR)

1. Writing this bit has no functional impact.

Table 296. NCR Field Descriptions

Field	Description
0 NLOCK0	NMI Configuration Lock Register 0 Writing a 1 to this bit locks the configuration for the NMI until it is unlocked by a system reset. Writing a 0 has no effect.
1:2 NDSS0	NMI Destination Source Select 0 This value is reset on system reset. 00 non-maskable interrupt 01 critical interrupt 10 machine check request 11 reset request
3 NWRE0	NMI Wakeup Request Enable 0 1 NMI0 wakeup is enabled. A set NIF0 bit or set NOVF0 bit causes a system wakeup request. 0 NMI0 wakeup is disabled. System wakeup requests from the corresponding NIF0 bit are disabled.
5 NREE0	NMI Rising-edge Events Enable 0 1 Rising-edge event is enabled. 0 Rising-edge event is disabled.
6 NFEE0	NMI Falling-edge Events Enable 0 1 Falling-edge event is enabled. 0 Falling-edge event is disabled.
7 NFE0	NMI Filter Enable 0 Enable analog glitch filter on the NMI pad input. 1 Filter is enabled. 0 Filter is disabled.
8 NLOCK1	NMI Configuration Lock Register 1 Writing a 1 to this bit locks the configuration for the NMI until it is unlocked by a system reset. Writing a 0 has no effect.

Table 296. NCR Field Descriptions (continued)

Field	Description
9:10 NDSS1	NMI Destination Source Select 1 This value is reset on system reset. 00 non-maskable interrupt 01 critical interrupt 10 machine check request 11 reset request
11 NWRE1	NMI Wakeup Request Enable 1 1 NMI1 wakeup is enabled. A set NIF1 bit or set NOVF1 bit causes a system wakeup request 0 NMI1 wakeup is disabled. System wakeup requests from the corresponding NIF1 bit are disabled
13 NREE1	NMI Rising-edge Events Enable 1 1 Rising-edge event is enabled. 0 Rising-edge event is disabled.
14 NFEE1	NMI Falling-edge Events Enable 1 1 Falling-edge event is enabled. 0 Falling-edge event is disabled.
15 NFE1	NMI Filter Enable 1 Enable analog glitch filter on the NMI pad input. 1 Filter is enabled. 0 Filter is disabled.
16 NLOCK2	NMI Configuration Lock Register 2 Writing a 1 to this bit locks the configuration for the NMI until it is unlocked by a system reset. Writing a 0 has no effect.
17:18 NDSS2	NMI Destination Source Select 2 This value is reset on system reset. 00 non-maskable interrupt 01 critical interrupt 10 machine check request 11 reset request
19 NWRE2	NMI Wakeup Request Enable 2 1 NMI2 wakeup is enabled. A set NIF2 bit or set NOVF2 bit causes a system wakeup request. 0 NMI2 wakeup is disabled. System wakeup requests from the corresponding NIF2 bit are disabled.
21 NREE2	NMI Rising-edge Events Enable 2 1 Rising-edge event is enabled. 0 Rising-edge event is disabled.
22 NFEE2	NMI Falling-edge Events Enable 2 1 Falling-edge event is enabled. 0 Falling-edge event is disabled.
23 NFE2	NMI Filter Enable 2 Enable analog glitch filter on the NMI pad input. 1 Filter is enabled. 0 Filter is disabled.

Table 296. NCR Field Descriptions (continued)

Field	Description
24 RLOCK	Reset Configuration Lock Register Writing a 1 to this bit locks the configuration for the reset until it is unlocked by a system reset. Writing a 0 has no effect.
25:26 RDSS	Reset Destination Source Select This value is reset on system reset. 00 Reset request to RGM 01 No reaction 10 No reaction 11 No reaction
27 RWRE	Reset Wakeup Request Enable 1 A set RIF bit or set ROVF bit causes a system wakeup request. 0 System wakeup requests from the corresponding RIF bit are disabled.
29 RREE	Reset Rising-edge Events Enable 1 Rising-edge event is enabled. 0 Rising-edge event is disabled.
30 RFEE	Reset Falling-edge Events Enable 1 Falling-edge event is enabled. 0 Falling-edge event is disabled.
31 RFE	Reset Filter Enable Enable analog glitch filter on the NMI pad input. 1 Filter is enabled. 0 Filter is disabled.

Note: Refer to the Chapter 7: Device configurationto see how the Figure 199: NMI Configuration Register (NCR) is used in the device.

Note: Writing a '0' to both NREEx and NFEEx disables the NMI functionality completely (no system wakeup or interrupt will be generated on any pad activity).

22.3 Functional Description

22.3.1 General

This section provides a complete functional description of the Wakeup unit.

22.3.2 Non-Maskable Interrupts

The Wakeup unit supports up to 4 non-maskable interrupts which can be allocated to any pad necessary at the SoC level. This allocation is fixed per SoC.

The Wakeup unit supports the generation of 3 types of interrupts per NMI input to the SoC. The Wakeup unit supports the capturing of a second event per NMI input before the interrupt is cleared, thus reducing the chance of losing an NMI event.

Each NMI passes through a bypassable analog glitch filter.

Note: Glitch filter control and pad configuration should be done while the NMI is disabled in order to avoid erroneous triggering by glitches caused by the configuration process itself.

22.3.2.1 NMI Management

Each NMI can be enabled or disabled independently. This can be performed using the single NCR register laid out to contain all configuration bits for a given NMI in a single byte (see *Figure 199*). A pad defined as an NMI can be configured by the user to recognize interrupts with an active rising edge, an active falling edge or both edges being active. A setting of having both edge events disabled results in no interrupt being detected and should not be configured.

The active NMI edge is controlled by the user through the configuration of the NREE and NFEE bits.

Note: After reset, NREE and NFEE are set to '0', therefore the NMI functionality is disabled after reset and must be enabled explicitly by software.

Once a pad's NMI functionality has been enabled, the pad cannot be reconfigured in the IOMUX to override or disable the NMI.

The NMI destination interrupt is controlled by the user through the configuration of the NDSS bits. See *Table 296* for details.

Each NMI supports a status flag and an overrun flag which are located in the NSR register (see *Figure 198*). This register is a clear-by-write-1 register type, preventing inadvertent overwriting of other flags in the same register. The status flag is set whenever an NMI event is detected. The overrun flag is set whenever an NMI event is detected and the status flag is set (it has not yet been cleared).

Note: The overrun flag is cleared by writing a '1' to the appropriate overrun bit in the NSR register. If the status bit is cleared and the overrun bit is still set, the pending interrupt will not be cleared.

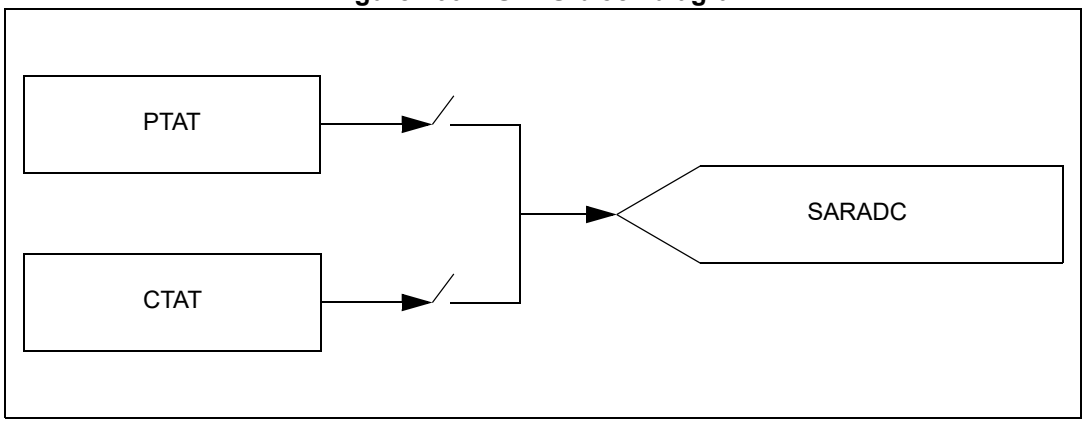
23 Temperature sensor (TSENS)

23.1 Introduction

The TSENS module provides an analog voltage that is proportional to the internal temperature of the SPC574Sx. This voltage can be sampled by the ADC and converted to an actual temperature by using factory-programmed calibration constants.

Figure 200 shows a block diagram of the TSENS module. The module operates on the principles of a basic silicon bandgap sensor that measures the difference in base-emitter voltages of multiple transistors, which can be related to the absolute temperature of the device.

Figure 200. TSENS block diagram



23.2 Features

The TSENS has the following features:

- Temperature monitoring range: –40 to +150 °C
- Sensitivity: approximately 5.14 mV/°C

23.3 Signals

The TSENS does not use any input signals external to the device.

23.4 TSENS outputs

TSENS produces two outputs from separate sources:

- Proportional to absolute temperature (PTAT). PTAT channel number is TCH120 (test channel).
- Complementary to absolute temperature (CTAT). CTAT channel number is TCH121 (test channel).

23.5 Obtaining device temperature using TSENS

In order to obtain the device temperature using TSENS, you must do the following:

- Extract the necessary TSENS calibration constants from the SPC574Sx test flash memory (see *Section 23.5.1*).
- Measure PTAT and CTAT voltages through the ADC.
- Calculate the device temperature using the equations in *Section 23.5.2*.

The need for the measurements in two different modes is driven by the fact that although the TSENS output is linear in either mode, the intercept of the V_{TSENS}-T plot varies significantly based on the ADC reference voltage. Performing the two measurements allows the resulting equations to be independent of this reference voltage.

23.5.1 TSENS calibration constants

The equations needed to convert V_{TSENS} into device temperature information depend on four calibration constants as described in *Table 297*. These constants are determined during factory testing and stored in the test flash memory.

Table 297. TSENS calibration constants (1)

Constant	Description
P_1	Code from the ADC converting PTAT voltage at 150 °C
P_2	Code from the ADC converting PTAT voltage at –40 °C
C_1	Code from the ADC converting CTAT voltage at 150 °C
C_2	Code from the ADC converting CTAT voltage at –40 °C

1. Flash locations for P2, C2, P1, and C1 are 0x400000, 0x400002, 0x400004, and 0x400006 respectively for Temperature Sensor 0. Flash locations for P2, C2, P1, and C1 are 0x4000F0, 0x4000F2, 0x4000F4, and 0x4000F6 respectively for Temperature Sensor 1.

23.5.2 Equations for converting TSENS voltage to device temperature

In the equations below:

- P_n and C_n are the calibration constants described in *Section 23.5.1*
- T is the device temperature in °C
- P_x is the code from the ADC converting the PTAT voltage output at a generic temperature T
- C_x is the code from the ADC converting the CTAT voltage output at a generic temperature T
- $T_2 = -40$
- $T_1 = +150$

Given that:

Equation 5

$$A = P_x C_2 - P_2 C_x$$

Equation 6

$$B = C_x P_1 - P_x C_1$$

The temperature is calculated as:

Equation 7

$$T = T_2 + \frac{(T_1 - T_2) \cdot A}{A + B}$$

23.6 Equations for converting CTAT and PTAT voltages into constant reference code

By converting the two outputs of CTAT and PTAT through ADC, one can find out a constant reference code which does not change with temperature.

- Pn and Cn are the calibration constants described in *Table 297*
- ref_code is the value of the constant reference code
- Px is the code from the ADC converting PTAT voltage output at any temperature T with any ADC reference voltage Vref
- Cx is the code from the ADC converting CTAT voltage output at the same temperature T with the same ADC reference voltage Vref

Equation 8

$$\text{ref_code} = P_x + \left(\frac{(P_2 - P_1)}{(C_1 - C_2)}\right) . Cx$$

Note: It is mandatory to measure Px and Cx at the same ADC reference voltage and at the same temperature.

24 Clocking

24.1 Introduction

This chapter describes the architecture for the system level clocks and includes the following information:

- System clock specifics
- Clock architecture
- Clock sources
- Clock monitoring
- Programmable clock dividers
- Clock control registers

The system clock (SYSCLK) for the platform (which includes the two CPUs (Main Core_0 and Checker Core_0s), memories, and debug logic is independent from the peripheral clocks. This allows SYSCLK to be a frequency modulated clock to reduce electromagnetic emissions, while having a precise clock for the peripheral timer and communication functions. It also allows the system clock to be reduced during low computational activity periods to help reduce power consumption, while providing a fixed frequency to the communication interfaces and timers.

This device boots from the internal 16 MHz RC oscillator (IRCOSC), and uses this as a backup clock in the event of a PLL or oscillator failure (if backup clock is enabled). The backup clock can be enabled in the Mode Entry Module (MC_ME). See *Chapter 52: Mode Entry Module (MC_ME)* for details.

There are two possible ways to provide the source clock:

- External oscillator/External crystal
- Internal 16 MHz RC oscillator

From one of these input sources, the internal clocks are generated from both PLLs, PLL0 and PLL1, using PLL0_PHI and PLL1_PHI outputs, respectively. These two clocks, along with the XOSC and IRCOSC, can be selected to drive system peripherals depending on the configuration of the Auxiliary Clock Selectors in the Clock Generation Module (MC_CGM). The PHI1 output of PLL0 can also be used as the clock source for PLL1. Six Auxiliary Clock Selectors are utilized, allowing developers to select an independent clock source for each of the system peripherals.

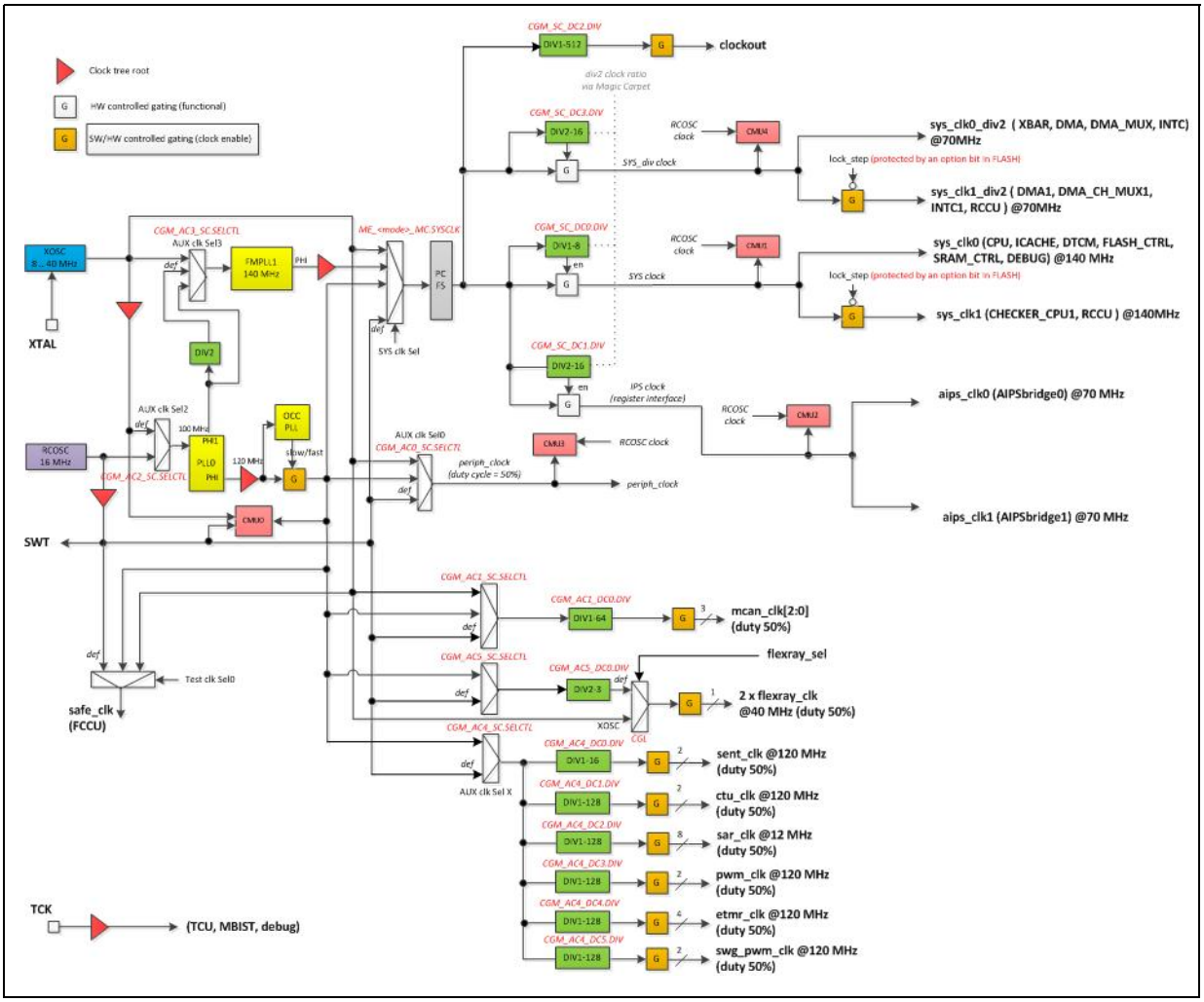
Each of the outputs of the module clock selectors have individual dividers that allow for more clock frequency granularity with the ability to divide the clock selector outputs by up to 512 for a given peripheral. Clock Monitor Units (CMUs) are connected to outputs of clock dividers as shown in *Figure 201: Clock generation architecture* to verify that their clock frequencies stay within necessary operating limits. The action taken when a CMU detects an issue with the monitored clock depends on the CMU configuration. When an issue with a clock is found, the CMUs can signal an interrupt or initiate a system reset.

24.2 Clock generation

The top-level clock generation architecture can be seen in *Figure 201*. All clock selectors and dividers are located and programmed in the MC_CGM. All dividers are integer dividers (1, 2, 3, …, *n*) with the ranges shown in the divider blocks in *Figure 201*.

Note: CMU_1, CMU_2 ,CMU_3 and CMU_4 in Figure 201 use IRCOSC for their reference clock, whereas the user can select either IRCOSC or XOSC for the reference in CMU_0.

Figure 201. Clock generation architecture



24.2.1 MC_CGM registers

Table 298 shows the relationship between the clocks in *Figure 201* and the MC_CGM registers.

Table 298. MC_CGM relationship to clocks(1)

MC_CGM register(s)	Register description	Output clock	CMU(2)
CGM_SC_DC0	System(3) Clock Divider Configuration 0	Core/XBAR (XBAR_CLK)	CMU_1
CGM_SC_DC1	System Clock Divider Configuration 1	PBRIDGE (PBRIDGE_CLK)	CMU_2
CGM_SC_DC2	System Clock Divider Configuration 2	Clock out divisor	—
CGM_SC_DC3	System Clock Divider Configuration 3	SYS_CLK/2 (INTC, DMA)	CMU_4

Table 298. MC_CGM relationship to clocks[1] (continued)

MC_CGM register(s)	Register description	Output clock	CMU[2]
CGM_AC0_DC0 CGM_AC0_SC	Aux Clock 0 Divider Configuration 0 Aux Clock 0 Select Control	PIT clock divisor	—
CGM_AC0_DC1 CGM_AC0_SC	Aux Clock 0 Divider Configuration 1 Aux Clock 0 Select Control	DSPI	—
CGM_AC0_DC2 CGM_AC0_SC	Aux Clock 0 Divider Configuration 2 Aux Clock 0 Select Control	LinFlexD	—
CGM_AC1_DC0 CGM_AC1_SC	Aux Clock 1 Divider Configuration 0 Aux Clock 1 Select Control	MCAN clock divisor	—
CGM_AC2_SC	Aux Clock 2 Select Control	PLL0	—
CGM_AC3_SC	Aux Clock 3 Select Control	PLL1	—
CGM_AC4_DC0	Aux Clock 4 Divider Configuration 0	SENT_CLK	—
CGM_AC4_DC1	Aux Clock 4 Divider Configuration 1	CTU_CLK	—
CGM_AC4_DC2	Aux Clock 4 Divider Configuration 2	SAR_CLK	—
CGM_AC4_DC3	Aux Clock 4 Divider Configuration 3	PWM_CLK	—
CGM_AC4_DC4	Aux Clock 4 Divider Configuration 4	ETMR_CLK	—
CGM_AC4_DC5	Aux Clock 4 Divider Configuration 5	SWG_PWM_CLK	—
CGM_AC5_DC0	Aux Clock 5 Divider Configuration 0	FLEXRAY_CLK	—

1. See the *Chapter 27: Clock Generation Module (MC_CGM)* for specifics on register implementation.
2. CMU_0 not shown in table since it does not monitor an output clock from the MC_CGM.
3. This clock is used for the following modules: CPU, XBAR, PFlash controller, PRAM controller, Debug.

24.3 System clock frequency limitations

The maximum frequency of operation for the device system level clocks is given in *Table 299*.

Table 299. Maximum system level clock frequencies

System clock	Max frequency (MHz)[1]
Core	140
XBAR, system RAM	140
PBRIDGE_0	70
DMA	70
Interrupt controller	70
DSPI_CLK, LIN_CLK	120
Peripheral clock	120
PLL0_PHI	120
PLL0_PHI1	100

Table 299. Maximum system level clock frequencies (continued)

System clock	Max frequency (MHz)[1]
PLL1_PHI	140
ADCs	12
FlexRay	40
Max protocol clock	40
Max host clock	70
SENT	120
CTU	120
SAR	12
FlexPWM	120
eTimer	120
SWG_PWM	120
CCCU	120
TCK (JTAGM)	70
TCK (PAD)	35

1. The user must ensure that these values are not exceeded.

In order to maintain synchronization between the different system clock branches (for example, SYS_CLK output from the System Clock Selector), there are limitations on the frequencies of those clocks. The system clocks must be binary multiples of each other with the frequency relationships described in the electrical specifications.

Note: CORE_CLK and PBRIDGE0_CLK are not required to be at the same frequency.

CORE_CLK and PBRIDGE0_CLK could be at the same frequency in the range from 0 Hz to the PBRIDGE_0 maximum frequency (70 MHz).

The ratio between CORE_CLK and PBRIDGE0_CLK shall not be greater than 2.

In order to maintain synchronization between divider clocks on auxiliary clock 4, there are limitations on the clock frequencies of its dividers. The ratio between each divider should always be an integer value.

24.3.1 JTAG frequencies

Table 300 shows the maximum frequencies of the JTAG interface.

Table 300. JTAG frequencies

Configuration	e200z420n3 max. internal JTAG frequency	Max. external JTAG frequency	Notes
Unconfigured clock	—	See electrical characteristics for maximum usable external JTAG frequency	Assumes 16 MHz clock (IRCOSC) with ± 1.5% tolerance, and JTAG usable at 1/2 CPU frequency
Clock configured to 140 MHz	70 MHz		JTAG usable at 1/2 CPU frequency

24.4 Default clock configuration

At power up, the IRCOSC is the default clock for the entire system. All system clock dividers are set to ‘1’ at power on reset.

24.5 Clock sources

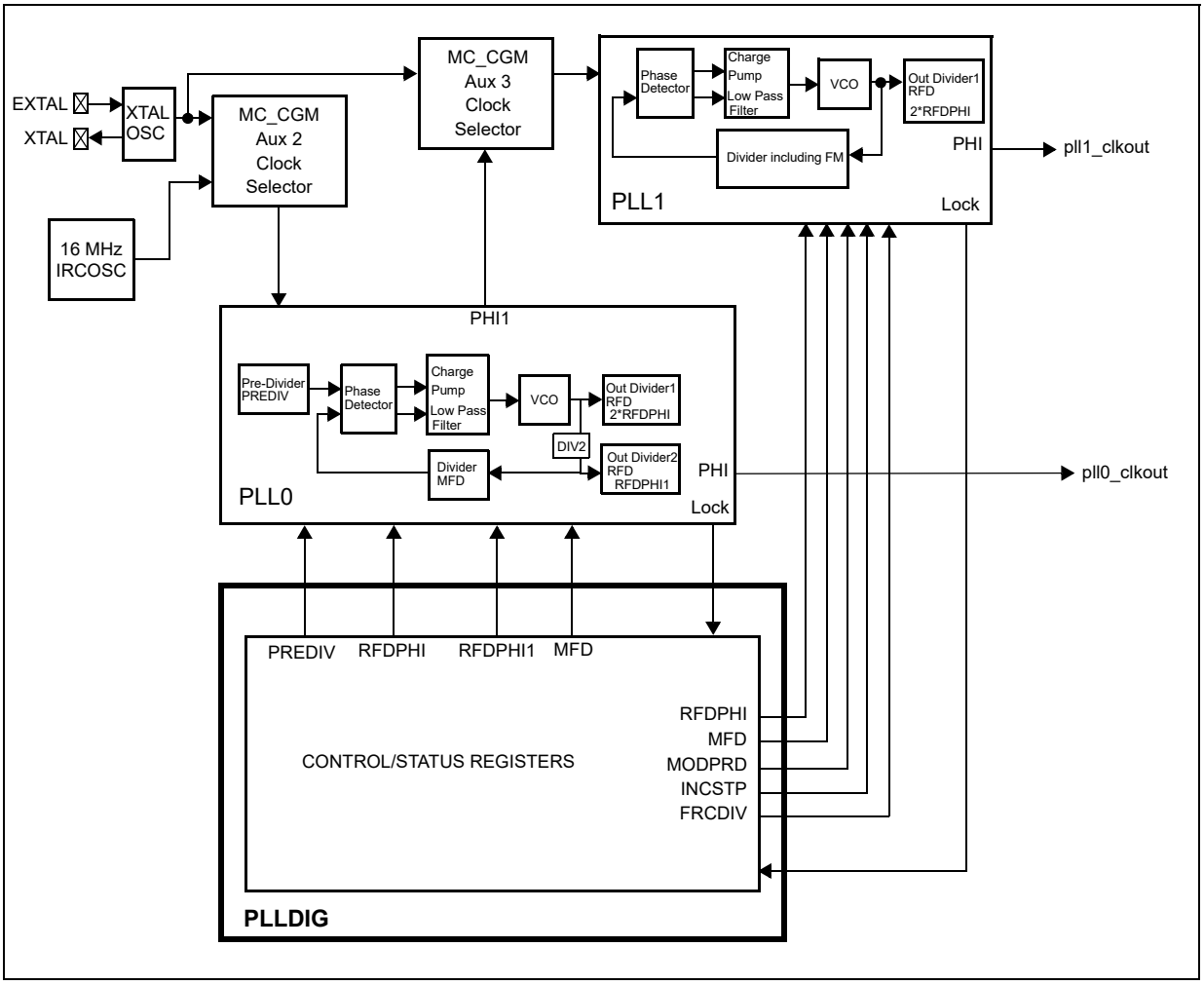
The following clock sources are described in this section:

- PLL0_PHI
- PLL1_PHI (FMPLL)
- External oscillator (XOSC)
- External clock (EXTAL bypass)
- 16 MHz internal RC oscillator (IRCOSC)

24.5.1 PLL

The PLL is a dual-PLL providing separate system and peripheral clocks as shown in *Figure 201*. The PLLs are disabled after power is cycled and must be enabled by software. The block diagram for the Dual PLL is shown in *Figure 202*.

Figure 202. SPC574Sx dual PLL digital interface block diagram



24.5.1.1 PLL0 — base PLL (non-FM)

PLL0 is the primary PLL. This PLL0 provides a non-Frequency Modulated clock to the peripheral IPs and the reference clock to PLL1.

24.5.1.1.1 PLL0 input clocks

The input clock sources for PLL0 are the XOSC, IRCOSC, and external reference clock. The external reference clock is input on the EXTAL pin.

24.5.1.1.2 PLL0 output clocks

The output clocks from PLL0 are PHI and PHI1. The PHI output clock drives the peripheral clock and the system clock when PLL1 is not locked or active. The PHI1 output clock provides the input reference clock for PLL1.

24.5.1.2 PLL1 — FMPLL

PLL1 is a Frequency Modulated PLL (FMPLL) that is used to drive the system clock.

24.5.1.2.1 PLL1 input clocks

The input clocks to PLL1 are the PHI1 output clock from PLL0, PHI1 output clock from PLL0 divided by 2 and the external reference clock.

24.5.1.2.2 PLL1 output clocks

The output clock from PLL1 is the PHI clock, which drives the system clock. The PHI output clock contains a fractional divider that can be applied to the loop divide of the PLL to achieve good granularity in the PLL1 PHI output clock frequency.

24.5.1.2.3 PLL register interface

This device has a single digital interface for the dual PLLs. The digital interface provides register control of all features of the PLLs. Details are provided in the Dual PLL Digital Interface (PLLDIG) chapter.

Note: Certain PLL registers can be write-protected. Refer to Section 7.11.4.4: PLL Digital Interface (PLL_DIG) protected registers of the Chapter 7: Device configuration for details.

24.5.1.3 PLL register reset values

All reset values for the PLL_DIG registers that are device-specific are shown in *Table 301*. All other register reset values are shown in the *Chapter 25: Dual PLL Digital Interface (PLLDIG)*.

Table 301. Dual PLL Digital Interface register reset values

Offset	Register	Reset value
0008h	PLL0DV — PLL0 Divider Register	0000_0000h
0028h	PLL1DV — PLL1 Divider Register	0000_0000h

24.5.1.4 Loss of clock detection

Loss of clock detection should be enabled at all times when the PLLs are enabled. There is a software option to generate a reset or interrupt in the event of a loss of clock condition. An optional backup clock is implemented within the PLL that software may enable or disable.

24.5.2 External oscillator (XOSC)

The external oscillator (XOSC) is provided as a reference for the on-chip PLLs. It can also be used as a clock source to the ADCs, timer, serial interfaces, and as a system clock source. It is available for observation on either of the CLKOUT pins, and is used as a reference to calibrate the IRCOSC frequency.

The external oscillator allows a crystal or external clock to be used as the reference clock for the microcontroller. The XOSC has the following features:

- Internal load capacitors for 20 MHz/40 MHz crystal oscillators
- Automatic Loop Control (ALC) to remove need for external series resistor
- Gain selection for <= 20 MHz and 40 MHz crystals to allow for optimal startup margin without overdrive issues
- Reference clock to PLL0 and to PLL1
- Reference clock to CMU_0 (IRCOSC trimming)
- Option to drive the CAN protocol clock directly from the XOSC

The XOSC provides support for 8 – 40 MHz crystal inputs, has integrated load capacitors and ALC to remove the need for external series resistor.

Note: The user must take care when using 16 MHz crystals or lower, because the S_XOSC flag in the ME_GS register will not be set if the CMU_PLL (CMU 0) default configuration is used. Indeed CMU_PLL (CMU 0) is programmed by default to use crystal frequencies greater than IRC (16 MHz) + 0.5 MHz and this condition is no longer valid with XOSC <= 16 MHz. In this case, to use an XOSC frequency less than or equal to 16 MHz, an RCDIV value (CMU_0.CSR register) that respects the previous formula (CMU_CSR.B.RCDIV = 3) must be used. This setting has to be done before enabling XOSC in the user code.

24.5.2.1 XOSC startup

Enabling of the oscillator at startup is determined by a bit in the UTEST row of flash memory (UTEST DCF Miscellaneous row, bit 24). The default value has the XOSC disabled (UTEST Misc bit 24 = 0 (XOSC EN)).

If enabled, the oscillator is started in phase 3 of the reset sequence, after negation of the internal POR circuits, but while the external PORST input is asserted. This is done to allow fast application start-up time while accounting for settling time of the power regulators.

The selection of internal or external load capacitors on the XTAL/EXTAL pins is also determined by a bit in the UTEST row of flash memory. The default value is external load capacitors (UTEST Misc bit 25 = 1 (XOSC EXT LOAD)). Load capacitor values are determined by the crystal manufacturer data sheet requirements, while accounting for stray PCB and on-chip capacitance. See the device data sheet for on-chip capacitance values.

When using the internal load capacitors for the oscillator, the startup value of the capacitors is stored in the UTEST row of the flash memory. During phase 3 of the device reset sequence, the oscillator UTEST values are read and driven to the oscillator. The enable/disable state of the oscillator is captured in the MC_ME module at this time. After reset, the oscillator can be enabled/disabled by software in the MC_ME module. Internal load capacitor selection is maintained by the oscillator if disabled after reset. The default reset value for the XOSC trimming capacitors does not trim the internal capacitor values (UTEST Misc bit 26:30 = 00000b (XOSC LOAD CAP SEL)).

The internal load capacitor values stored in UTEST flash memory row have triple-voting flip-flop (TVF) implementation to prevent an incorrect value, and thus an undetectable error in the system.

The XOSC has the ability to be started with either 8 MHz to 20 MHz, or 40 MHz to 44 MHz crystal. The XOSC EN_40 MHz bit in the UTEST Misc row selects which crystal source will be used. The default value of this field is for an 8 MHz to 20 MHz crystal (UTEST Misc bit 21 = 0).

See the flash memory chapter for more information on the UTEST row programming.

Note: Certain XOSC registers can be write-protected. Refer to Section 7.11.4.5: Oscillator Digital Interface (XOSC) protected registers of the Chapter 7: Device configuration for details.

24.5.2.2 XOSC reset value

Table 302 shows the default reset value for the XOSC registers.

Table 302. XOSC register reset values

Offset	Register	Reset value
00h	XOSC_CTL—XOSC Control Register	0030_0000h

24.5.3 16 MHz internal RC oscillator (IRCOSC)

The microcontroller has a 16 MHz internal RC oscillator that is always enabled and can be used as the clock source for the PLLs. The IRCOSC is the default clock after reset. The register interface is used for user trimming of the oscillator and dividing the output from 1 to 32.

24.5.3.1 IRCOSC register interface

This device has a dedicated digital interface for the IRCOSC. The digital interface provides a register for fine tuning the IRCOSC frequency by the user. The IRCOSC is always enabled when the device is powered.

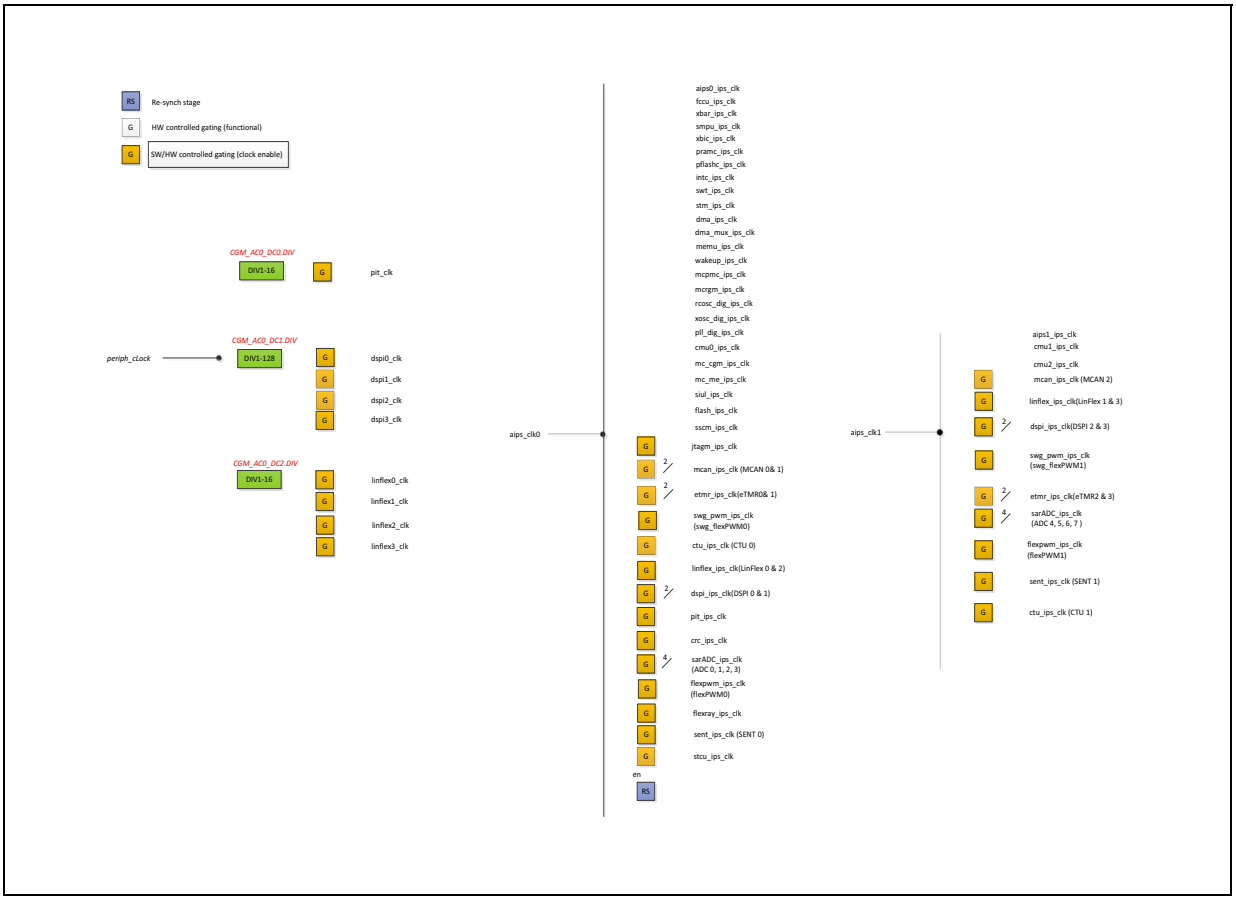
24.5.3.2 IRCOSC register write-protection

The IRCOSC registers do not utilize write-protection.

24.6 Peripheral clocks

Figure 203 shows the clock distribution to the cores and peripheral modules.

Figure 203. Clock distribution



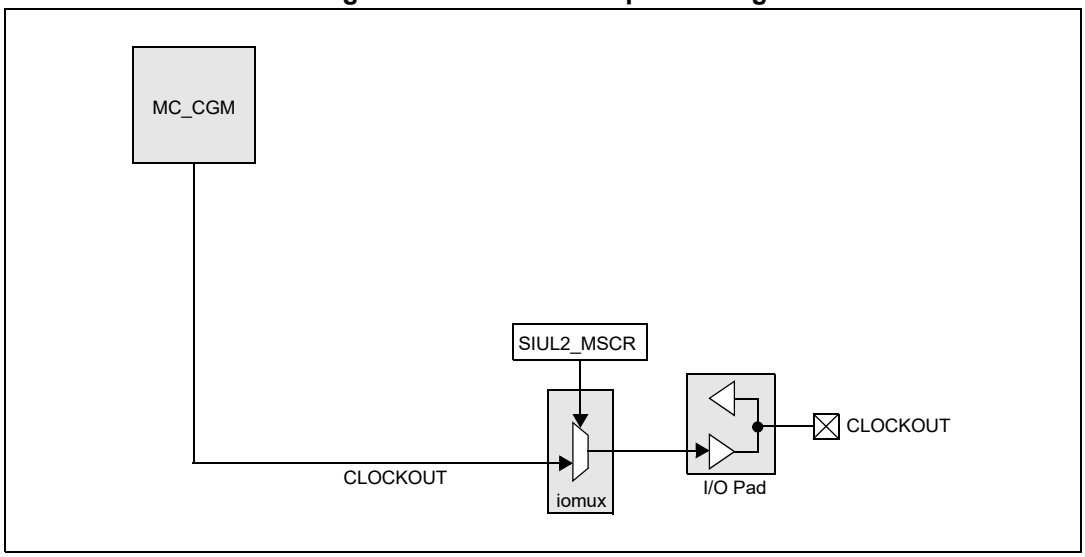
24.6.1 MCAN clocking

The CAN_CLK is generated from either the XOSC or PLL0:PHI clocks, depending on the configuration of CGM_AC1_SC in the MC_CGM. The CAN_CLK can also be reduced by configuring the dividing factor in the CGM_AC1_DC0 (divide by 1 to 64).

24.6.1.1 System clock

There is one system clock output pin on the device: CLOCKOUT. This is a divided version of device's system clock and can be observed outside the device by programming CGM_SC_DC2 register. See *Figure 201*.

Figure 204. CLOCKOUT pin muxing



24.7 Clock monitoring

For all safety relevant clocks the microcontroller uses clock monitoring units (CMUs) to detect a missing clock or incorrect frequency. Each CMU is programmed independently (see *Figure 201* for CMU distribution). The IRCOSC and XOSC are used as the clock monitor references. Detailed information on the CMUs can be found in the Clock Monitor Unit chapter.

24.7.1 CMU configuration

This section explains the CMU configuration.

Figure 205 shows the block diagram for CMU_0 on the SPC574Sx.

Figure 205. CMU_0 block diagram

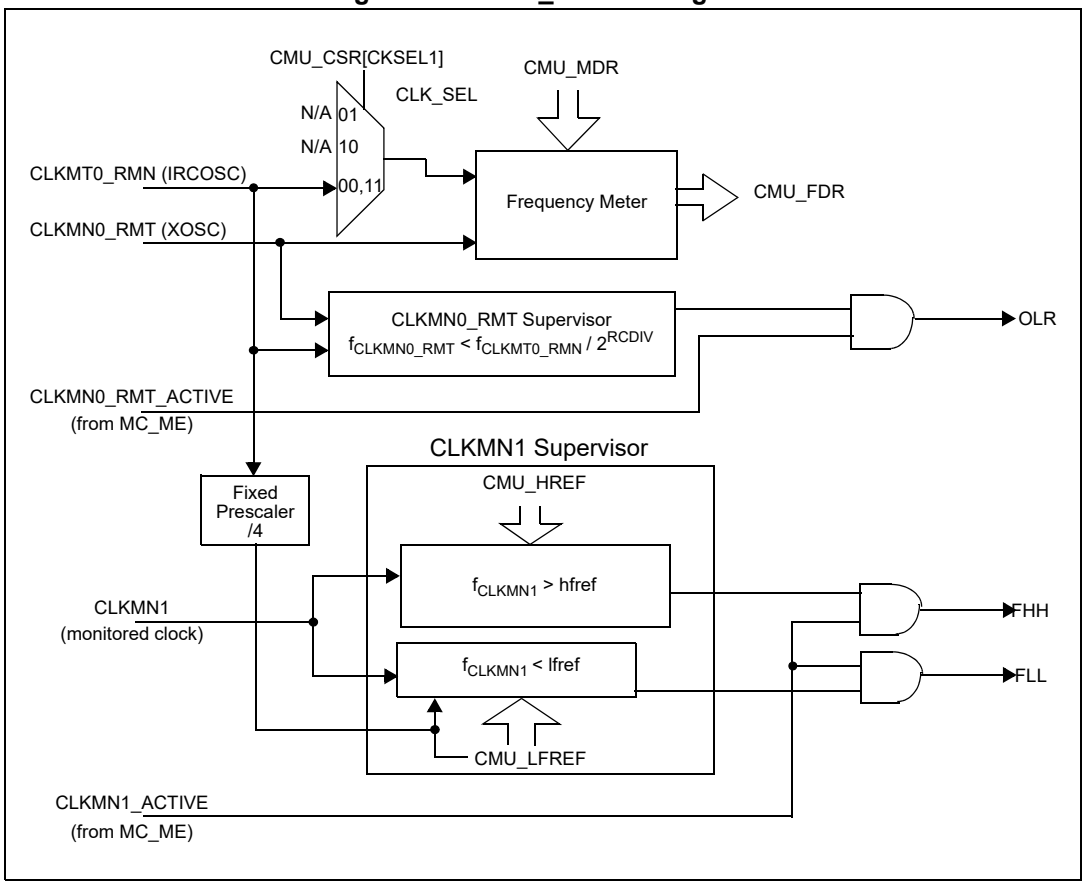
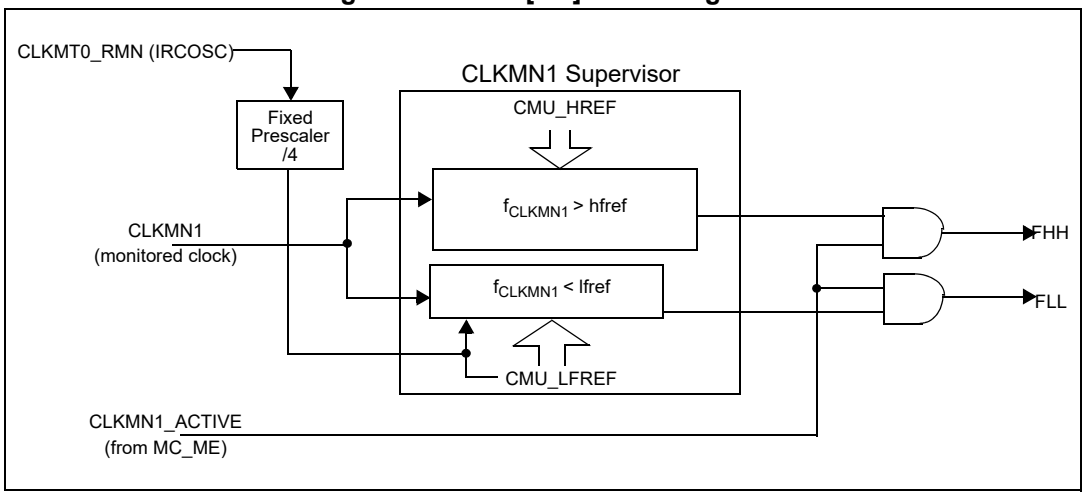


Figure 206 shows the block diagram for CMU[1:4] on the SPC574Sx.

Figure 206. CMU[1:4] block diagram



24.7.1.1 Clock input sources

Table 303 shows the clocks that are monitored by each CMU. These signals are connected internally on the chip, but are not accessible via pins on the boundary of the device. IRCOSC is the reference clock for all clock monitors. Only CMU_0 implements the XOSC

monitor. CMU_0 uses the IRCOSC clock to measure if the XOSC is too low. CMU_0 can also be used to calibrate the IRCOSC frequency using the XOSC. All other CMUs are configured identically.

Table 303. Clock input sources

Clock module	Monitored clock
CMU_0	PLL0:PHI, XOSC, IRCOSC
CMU_1	Main Core_0, Safety Core (Main Core_0, Checker Core_0s), XBAR_CLK
CMU_2	PBRIDGE_CLK
CMU_3	PERIPH_CLOCK
CMU_4	DMA, INTC

24.7.1.2 CMU registers and field availability

Table 304 specifies which registers and fields are available on a given CMU for this device.

Table 304. CMU registers availability

Address offset	Register	CMU	Notes
0x0000	CMU_CSR	All	Fields CMU_CSR[SFM] and CMU_CSR[CKSEL1] are present in CMU_0 only
0x0004	CMU_FDR	CMU_0	—
0x0008	CMU_HFREFR	All	—
0x000C	CMU_LFREFR	All	—
0x0010	CMU_ISR	All	Fields CMU_ISR[FLCI] and CMU_ISR[OLRI] are present in CMU_0 only.
0x0014	Reserved	—	—
0x0018	CMU_MDR	All	—

Note: Certain CMU registers can be write-protected. Refer to Section 7.11.4.2: Clock Monitor Unit (CMU) protected registers of the Chapter 7: Device configuration for details.

24.7.2 PLL0 monitor

Software can program an upper and lower limit on the expected PLL0 PHI output clock frequency. If the monitor is enabled and the measured frequency is above or below the limits, a flag bit is set and an interrupt, if enabled, is generated. The default condition of the clock monitor is disabled.

24.7.3 External oscillator (XOSC) monitor

The XOSC frequency is compared to a minimum value limit. If the measured XOSC frequency is below the limit, a flag is set and an interrupt, if enabled, is generated.

24.7.4 Internal RC oscillator (IRCOSC) monitor

The period of the IRCOSC can be measured in CMU_0, using the XOSC as a reference. This allows for application trimming of the IRCOSC frequency.

24.7.5 System clock monitors

A CMU is assigned to monitor the frequency of the core, XBAR, and both peripheral bridges (refer to *Figure 201*).

24.8 Loss of system clock behavior

This device has built-in mechanisms for detecting loss of the oscillator or PLL clocks, and provides several options for reaction to a loss of clock in the application. *Figure 207* gives a high-level view of the loss of clock logic.

24.8.1 Loss of PLL/XOSC clock

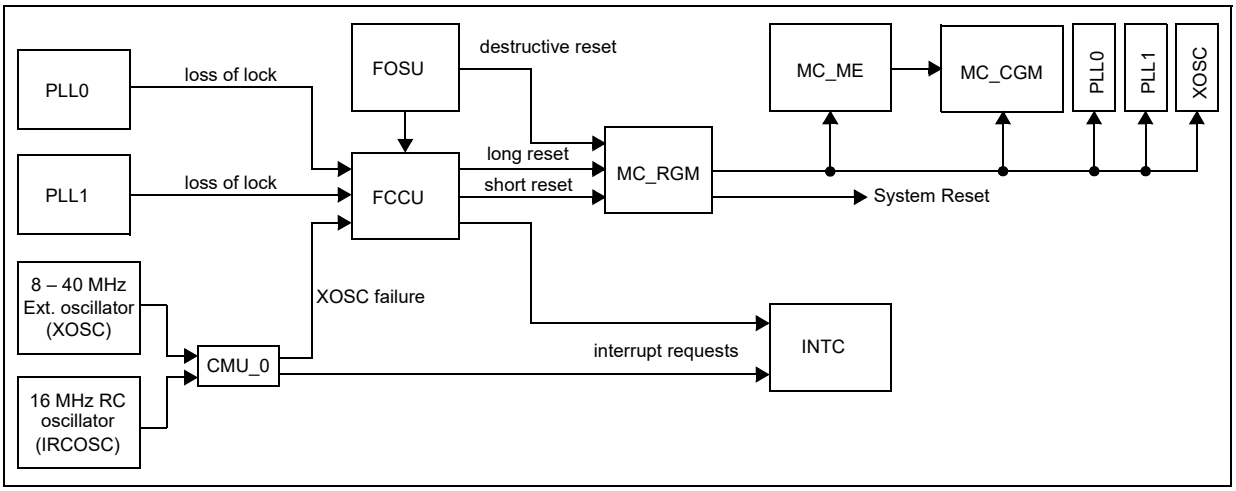
As shown in *Figure 207*, each loss of lock signal from each PLL and the XOSC failure signals are monitored by the FCCU. The FCCU can be programmed to generate a short or long reset sequence when a clock failure occurs, or simply to generate an interrupt. In the case of the interrupt, only the backup clock in the PLL can provide clocks to the system in order to perform a system clock switch. There is no automatic system clock switch, and the user is required to program the switch through the Mode Entry module (MC_ME).

If a long reset is selected the PLL, XOSC, MC_ME, and MC_CGM are reset to their default states, and the system clock is switched to the IRCOSC. If a short reset is selected, the clock configuration is maintained, but again, the PLL backup clock is the only available clock to clock the system through an MC_ME mode switch.

24.8.2 Loss of IRCOSC clock

The frequency of the IRCOSC clock is monitored by the frequency meter in CMU_0. There is no automated trigger of an FCCU error condition if the IRCOSC fails. Since the IRCOSC is the boot and backup clock, a failure (a loss of IRCOSC clock) is a catastrophic failure.

Figure 207. Loss of clock logic



25 Dual PLL Digital Interface (PLLDIG)

25.1 Introduction

The MCU provides a user interface and control over the dual PLL system composed of PLL0 and PLL1 blocks (PLLDIG). The two analog PLL blocks are cascaded, with the PHI1 output of PLL0 feeding the clock input of PLL1 as shown in *Chapter 24: Clocking*. The key feature of the dual PLL architecture is the ability to drive peripherals from the PLL0 PHI output, which is non-modulated and independent of the core clock frequency. The core and platform clocks are driven by PLL1 PHI frequency modulated output.

25.2 Block Diagram

The block diagram of the PLLDIG block is found in the *Chapter 24: Clocking* of the *SPC574Sx Reference Manual*.

25.3 Features

The Dual PLLDIG has the following features:

- Supports dual PLL in cascaded mode with PLL0 clock out as reference clock to PLL1.
- Reference clock pre-divider for finer frequency synthesis resolution.
- Reduced frequency divider for reducing the PLL0/PLL1 output clock frequency without causing the PLLs to lose lock.
- Programmable frequency modulation on PLL1.
- The frequency range after the prescaler is required to be 8-20MhzLock detect circuitry reports when the PLLs have achieved frequency lock, and continuously monitors lock status to report loss of lock conditions.
  - User-selectable ability to generate an interrupt request upon loss of lock.
  - The loss of lock indication is sent to the FCCU via the system glue logic.

Note: Please refer to the Device Data sheet for information on the input and output clock frequency range.

25.4 Modes of operation

The operating modes of the PLLs is determined by the value of PLL*n*CR[CLKCFG]. The various modes of operation for the PLL are:

- Normal mode with reference, and either PLL0 or both PLLs enabled

Note: Mode changes are controlled by configuring the MC_ME Mode Configuration registers (MC_ME_<mode>_MC).

Normal mode is defined as the mode where the clocks are driven by the PLL. When using a crystal for the clock reference, reset should remain asserted until the oscillator has stabilized.

When PLL0 is powered down and working in Bypass mode, PLL1 should be bypassed and powered down. PLL1 should be configured to work in normal mode only after PLL0 has

achieved lock (for example, bypass with PLL0 running or PLL0 in normal mode, PLL*n*SR[LOCK] = 1).

25.4.1 Normal mode with reference, PLL0 or both PLLs enabled

In the normal mode, PLL0 receives an input clock from the reference and the pre-divider. PLL0 multiplies the frequency to create the PLL0 output clock. The user must supply a crystal that is within the appropriate frequency range, the crystal manufacturer recommended external support circuitry and short signal route from the MCU to the crystal.

PLL0 generates a non-modulated clock which drives PLL0_PHI1. PLL1 can generate a frequency modulated clock or a non-modulated clock (for example, locked on a single frequency). The modulation rate, modulation depth, output divider (Reduced Frequency Divider) and whether the PLL1 is modulated or not can be programmed by writing to the PLL1FM register, *Section 25.5.2.7: PLL1 Frequency Modulation Register (PLL1FM)*.

Note: While powering down the PLLs, user must ensure that PLL1 is powered down first followed by powering down PLL0 for proper shut off.

Note: Only when the PLL0 achieves lock should PLL1 be configured to work in either mode by programming the appropriate MC_ME_<mode>_MC register.

25.5 Memory map and register definition

This section provides the memory map and detailed descriptions of all registers for configuring the PLLs.

25.5.1 Memory map

Table 305 shows the memory map. Addresses are given as offsets from the module base address. All registers can be accessed using 8-bit, 16-bit or 32-bit addressing.

Table 305. Dual PLL Digital Interface memory map

Offset (hex)	Register	Section
0x0000	PLL0 Control Register (PLL0CR)	*Section 25.5.2.1*
0x0004	PLL0 Status Register (PLL0SR)	*Section 25.5.2.2*
0x0008	PLL0 Divider Register (PLL0DV)	*Section 25.5.2.3*
0x000C–0x001F	Reserved	
0x0020	PLL1 Control Register (PLL1CR)	*Section 25.5.2.4*
0x0024	PLL1 Status Register (PLL1SR)	*Section 25.5.2.5*
0x0028	PLL1 Divider Register (PLL1DV)	*Section 25.5.2.6*
0x002C	PLL1 Frequency Modulation Register (PLL1FM)	*Section 25.5.2.7*
0x0030	PLL1 Fractional Divide Register (PLL1FD)	*Section 25.5.2.8*
0x0034–0x003F	Reserved	

25.5.2 Register descriptions

Some of the register reset values are specifically configured for each unique device by external configuration signals or parameters.

25.5.2.1 PLL0 Control Register (PLL0CR)

Offset: 0x0000 Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	CLKCFG(3)		EXPDIE	0(1)	0(1)	0(1)	LOLIE(2)	0(1)	0(1)	0(1)
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. This field can be written at any time, but writes are ignored. Reads returns previously written value.
2. See the “Clocking” chapter for details on configuring this field.
3. CLKCFG can be written, but writes have no effect. Mode changes are implemented by writing to the appropriate MC_ME_<*mode*>_MC register.

Figure 208. PLL0 Control Register (PLL0CR)

Table 306. PLL0CR field descriptions

Field	Description
22:23 CLKCFG	Clock Configuration This field is indicates the operational state of PLL. 00 PLL off. 01 Reserved 10 Reserved. 11 Normal mode with PLL running When PLLs are externally powered down the CLKCFG field changes to 00b. When external power down is removed, these bits will read 11b. **Note:** In cascaded PLL configuration (PLL0 driving the reference clock for PLL1) the external power down signal for PLL1 must be removed only when PLL0 has locked, PLL0SR[LOCK] = 1.
24 EXPDIE	External Power Down Cycle Complete indication interrupt enable This bit enables generation of an interrupt when the external power down of the PLL is complete (for example, PLLs are powered down and powered back up). 0 Ignore interrupt. Interrupt not requested 1 Enable interrupt request on power down cycle completion
28 LOLIE	Loss of lock interrupt enable The LOLIE bit enables a loss of lock interrupt request when PLL0SR[LOLF] = 1. If PLL0SR[LOLF] = 0 or PLL0CR[LOLIE] = 0, the interrupt request is ignored. The interrupt request only occurs in normal mode. 0 Ignore loss of lock. Interrupt not requested 1 Enable interrupt request upon loss of lock

25.5.2.2 PLL0 Status Register (PLL0SR)

Offset: 0x0004 Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	EXTPDF	0	0	0	LOLF	LOCK	0	0
W									w1c		*(1)		w1c		*(1)	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. This field is reserved, read only.

Figure 209. PLL0 Status Register (PLL0SR)

Table 307. PLL0SR field descriptions

Field	Description
24 EXTPDF	External Power Down Cycle Complete indication interrupt flag This bit indicates that the external power down of the PLL is complete (for example, PLLs are powered down and then powered up). User must write 1 to clear this bit. 0 PLLs not power cycled. Interrupt not requested 1 PLLs power down cycle is complete. Interrupt is requested
28 LOLF	Loss of lock flag This bit provides the interrupt flag for the loss of lock condition. Software must write 1 to LOLF to clear it, writing 0 has no effect. The flag is set when the PLL loses lock or when the divider/modulation registers are changed on the fly. This flag bit is sticky in the sense that if lock is reacquired, the bit will remain set until cleared by either writing 1 or asserting reset. 0 No loss of lock detected. Interrupt service not requested 1 Loss of lock detected. Interrupt service requested
29 LOCK	Lock status bit Indicates whether PLL has acquired lock. 0 PLL is unlocked 1 PLL is locked

25.5.2.3 PLL0 Divider Register (PLL0DV)

The PLL0DV register provides the PHI/PHI1 output clock reduced frequency dividers, pre-divider, and loop divider. The values of PREDIV and MFD should not be changed when the PLL is on and locked (working in Normal mode). If these fields are changed without powering down the PLL, the PLL will lose lock and generate either a reset or an interrupt, based on which is enabled. The reduced frequency divider fields (RFDPHI, RFDPHI1) can

be modified at anytime, but the changes only become effective after PLL0 is disabled, then reenabled.

Offset: 0x0008 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	RFDPHI1				0	0	0	0	0	RFDPHI					
W																
Reset	0	_(1)	_(1)	_(1)	_(1)	0	0	0	0	0	_(1)	_(1)	_(1)	_(1)	_(1)	_(1)

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	PREDIV			0	0	0	0	0	MFD						
W																
Reset	0	_(1)	_(1)	_(1)	0	0	0	0	0	_(1)	_(1)	_(1)	_(1)	_(1)	_(1)	_(1)

1. See the Clocking chapter for default reset value information.

Figure 210. PLL0 Divider Register (PLL0DV)

Table 308. PLL0DV field descriptions

Field	Description
1:4 RFDPHI1	PHI1 reduced frequency divider This 4-bit field determines the VCO/2 clock post divider for driving the PHI1 output clock. 00xx Reserved 0100 Divide by 4 0101 Divide by 5 ... 1110 Divide by 14 1111 Divide by 15
10:15 RFDPHI	PHI reduced frequency divider This 6-bit field determines the VCO clock post divider for driving the PHI output clock. 00 Reserved 01 Divide by 1 02 Divide by 2 03 Divide by 3 ... 3E Divide by 62 3F Divide by 63

Table 308. PLL0DV field descriptions (continued)

Field	Description
17:19 PREDIV	Input clock pre-divider This 3-bit field controls the value of the divider on the input clock. The output of the pre-divider circuit generates the reference clock to the PLL analog loop. The PREDIV value 000b causes the input clock to be inhibited. 000 Clock inhibit 001 Divide by 1 010 Divide by 2 ... 110 Divide by 6 111 Divide by 7
25:31 MFD	Loop multiplication factor divider This 7-bit field controls the value of the divider in the feedback loop. The value specified by the MFD bits establishes the multiplication factor applied to the reference frequency. 00 Reserved ... 07 Reserved 08 Divide by 8 ... 7E Divide by 126 7F Divide by 127

25.5.2.4 PLL1 Control Register (PLL1CR)

Offset: 0x0020 Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	CLKCFG(3)		EXPDIE	0(1)	0(1)	0(1)	LOLIE(2)	0(1)	0(1)	0(1)
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. This field can be written at any time, but writes are ignored. Reads returns previously written value.
2. See the “Clocking” chapter for details on configuring this field.
3. CLKCFG can be written, but writes have no effect. Mode changes are implemented by writing to the appropriate MC_ME_<*mode*>_MC register.

Figure 211. PLL1 Control Register (PLL1CR)

Table 309. PLL1CR field descriptions

Field	Description
22:23 CLKCFG	Clock Configuration This field is indicates the operational state of PLL. 00 PLL off 01 Reserved 10 Reserved 11 Normal mode with PLL running When PLLs are externally powered down the CLKCFG field changes to 00b. When external power down is removed, these bits will read 11b. **Note:** In cascaded PLL configuration (PLL0 driving the reference clock for PLL1) the external power down signal for PLL1 must be removed only when PLL0 has locked, PLL0SR[LOCK] = 1.
24 EXPDIE	External Power Down Cycle Complete indication interrupt enable This bit enables generation of an interrupt when the external power down of the PLL is complete (for example, PLLs are powered down and powered back up). 0 Ignore interrupt. Interrupt not requested 1 Enable interrupt request on power down cycle completion
28 LOLIE	Loss of lock interrupt enable The LOLIE bit enables a loss of lock interrupt request when PLL1SR[LOLF] = 1. If PLL1SR[LOLF] = 0 or PLL1CR[LOLIE] = 0, the interrupt request is ignored. The interrupt request only occurs in normal mode. 0 Ignore loss of lock. Interrupt not requested 1 Enable interrupt request upon loss of lock

25.5.2.5 PLL1 Status Register (PLL1SR)

Offset: 0x0024 Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	EXTPDF	0	0	0	LOLF	LOCK	0	0
W									w1c		*(1)		w1c		*(1)	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. This field is reserved, read only.

Figure 212. PLL1 Status Register (PLL1SR)

Table 310. PLL1SR field descriptions

Field	Description
24 EXTPDF	External Power Down Cycle Complete indication interrupt flag This bit indicates that the external power down of the PLL is complete (for example, PLLs are powered down and then powered up). User must write 1 to clear this bit. 0 PLLs not power cycled. Interrupt not requested 1 PLLs power down cycle is complete. Interrupt is requested
28 LOLF	Loss of lock flag This bit provides the interrupt flag for the loss of lock condition. Software must write 1 to LOLF to clear it, writing 0 has no effect. The flag is set when the PLL loses lock or when the divider/modulation registers are changed on the fly. This flag bit is sticky in the sense that if lock is reacquired, the bit will remain set until cleared by either writing 1 or asserting reset. 0 No loss of lock detected. Interrupt service not requested 1 Loss of lock detected. Interrupt service requested
29 LOCK	Lock status bit Indicates whether PLL has acquired lock. 0 PLL is unlocked 1 PLL is locked

25.5.2.6 PLL1 Divider Register (PLL1DV)

The PLL1DV register provides the output clock reduced frequency dividers and loop divider values. The value of MFD should not be changed when the PLL is on and locked (working in Normal mode). If the field is changed without powering down the PLL, the PLL will lose lock and generate either a reset or an interrupt, based on which is enabled. The reduced frequency divider field (RFDPHI) can be modified at anytime, but the changes only become effective after PLL1 is disabled, then reenabled .

Offset: 0x0028 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	RFDPHI[1]					
W																
Reset	0	0	0	0	0	0	0	0	0	0	_[1]	_[1]	_[1]	_[1]	_[1]	_[1]

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	MFD[1]						
W																
Reset	0	0	0	0	0	0	0	0	0	_[1]	_[1]	_[1]	_[1]	_[1]	_[1]	_[1]

1. See the Clocking chapter for default reset value information.

Figure 213. PLL1 Divider Register (PLL1DV)

Table 311. PLL1DV field descriptions

Field	Description
10:15 RFDPHI	PHI reduced frequency divider This 6-bit field determines the VCO clock post divider for driving the PHI output clock. 00 Reserved 01 Divide by 1 02 Divide by 2 ... 3E Divide by 62 3F Divide by 63
25:31 MFD	Loop multiplication factor divider This 7-bit field controls the value of the divider in the feedback loop. The value specified by the MFD bits establishes the multiplication factor applied to the reference frequency. 10 Divide by 16 11 Divide by 17 .... 21 Divide by 33 22 Divide by 34 All other settings are Reserved

25.5.2.7 PLL1 Frequency Modulation Register (PLL1FM)

The PLL1FM register enables frequency modulation on PLL1 and provides controls for the FM spread spectrum, modulation depth and modulation rate. This register should be written when PLL1CR[CLKCFG] = 00b (PLL1 powered down). Changing the values of PLL1FM[MODEN] and PLL1FM[MODSEL] fields once the PLL is running, and has locked, can result in the PLL losing its lock and the device being reset (if PLL1CR[LOLRE] = 1). The PLL would then require power cycling to regain normal functionality. The modulation period (PLL1FM[MODPRD]) and increment step fields (PLL1FM[INCSTP]) can be changed without losing lock, however, the output clock from PLL may have an incorrect frequency for a few cycles after either of these updates (as defined in the Data Sheet).

Offset: 0x002C Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	MODEN	MODSEL	MODPRD												
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	INCSTP														
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 214. PLL1 Frequency Modulation Register (PLL1FM)

Table 312. PLL1FM field descriptions

Field	Description
1 MODEN	Modulation enable This bit enables the frequency modulation. 0 Frequency modulation disabled 1 Frequency modulation enabled
2 MODSEL	Modulation selection This bit selects whether modulation is centered around the nominal frequency or spread below the nominal frequency. 0 Modulation centered around nominal frequency 1 Modulation spread below nominal frequency
3:15 MODPRD	Modulation period This 13-bit field is the binary equivalent of the modulation period variable derived from the formula: $MODPRD = round(\frac{f_{ref}}{4 \times f_{mod}})$ where f_{ref} represents the frequency of the feedback divider and f_{mod} represents the modulation frequency (see equation in *Section 25.6.4*).
17:31 INCSTP	Increment step This 15-bit field is the binary equivalent of the INCSTP variable derived from the formula: $INCSTEP = round(\frac{(2^{15} - 1) \times MD \times PLL1CR[MFD]}{100 \times 5 \times MODPRD})$ where *md* represents the peak modulation depth in percentage (+/- *MD* for centered modulation, -2**MD* for modulation below nominal frequency), and MFD represents the nominal value of the feedback loop divider (see equation in *Section 25.6.4*).

25.5.2.8 PLL1 Fractional Divider Register (PLL1FD)

The PLL1FD register provides the enable and fractional divide factor for the loop divider. This register should be written when PLL1CR[CLKCFG] = 00b (PLL1 powered down). Changing the values of FDEN once the PLL is running, and has locked, can result in the PLL losing its lock and the device being reset (if PLL1CR[LOLRE] = 1). The PLL would then require power cycling to regain normal functionality. The dither disable and fractional divider fields can be changed without losing lock, however, the output clock from PLL can have an incorrect frequency for a few cycles after either of these updates .

Offset: 0x0030 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	FDEN	0	0	0	0	0	0	0	0	0	0	0	0	DTHDIS	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	FRCDIV											
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 215. PLL1 Fractional Divider Register (PLL1FD)

Table 313. PLL1FD field descriptions

Field	Description
1 FDEN	Fractional Divide Enable This bit enables the fractional divider in the loop divider for PLL1. 0 Fractional divide disabled 1 Fractional divide enabled
14:15 DTHDIS	Dither Disable Bit 14 and bit 15 control noise of SSCG and Fractional Controller. 00 Increase PLL multiplication factor by 1/2^13 and has maximum noise control of SSCG and Fractional Controller. 01 Increase PLL multiplication factor by 1/2^13 and has some noise control of SSCG and Fractional Controller. 10 No influence on PLL multiplication factor and has some noise control of SSCG and Fractional Controller. 11 No influence on PLL multiplication factor and no noise control of SSCG and Fractional Controller.
20:31 FRCDIV	Fractional divide input When the fractional divide is disabled, the VCO clock frequency is the product of the input clock and the loop divide factor (MFD). When fractional divide is enabled, the mean VCO clock frequency is given by: $\text{VCO Frequency} = \text{InputClock} \times (\text{PLL1DV[MFD]} + (\text{FRCDIV} \div 2^{12}))$ where FRCDIV is the decimal equivalent of the FRCDIV bits (see *Section 25.6.2*). **Note:** Depending of DTHDIS value the multiplication factor can change as described in DTHDIS field description.

25.5.3 Register classification for safety requirements

This module is classified as a safety Vital Module (ViMo). Hence, the registers of this module have been classified as shown in the SPC574Sx Reference Manual “Functional Safety” chapter.

25.6 Functional description

This section explains the operation and configuration of the Dual PLLDIG module.

25.6.1 Input clock frequency

PLL0 and PLL1 are designed to operate over an input clock frequency range. The operating ranges for each PLL are discussed in detail in the *SPC574Sx Data Sheet*.

25.6.2 Clock configuration

The relationship between input and output frequency is determined by programming the PLL0DV, PLL1DV and PLL1FD registers, and calculated according to the following equations:

Equation 9

$$f_{pll0_phi} = f_{pll0_ref} \times \frac{PLL0DV[MFD]}{PLL0DV[PREDIV] \times PLL0DV[RFDPHI]}$$

Equation 10

$$f_{pll0_phi1} = f_{pll0_ref} \times \frac{PLL0DV[MFD]}{PLL0DV[PREDIV] \times PLL0DV[RFDPHI1]}$$

Equation 11

$$f_{pll1_phi} = f_{pll1_ref} \times \left(\frac{PLL1DV[MFD] + \frac{PLL1FD[FRCDIV]}{2^{12}}}{2 \times PLL1DV[RFDPHI]} \right)$$

Note: Depending of DTHDIS value the multiplication factor of *Equation 11* can change as described in PLL1DV[DTHDIS] field description (see *Table 313*).

The relationship between the VCO frequency f_{VCO} and the output frequency of the PLLs is determined by the configuration of the PLL0DV, PLL1DV and PLL1FD registers, according to the following equations:

Equation 12

$$f_{pll0_VCO} = \frac{f_{pll0_ref} \times PLL0DV[MFD] \times 2}{PLL0DV[PREDIV]}$$

Equation 13

$$f_{pll1_VCO} = f_{pll1_ref} \times \left(PLL1DV[MFD] + \frac{PLL1FD[FRCDIV]}{2^{12}} \right)$$

Note: Depending of DTHDIS value the multiplication factor of *Equation 13* can change as described in PLL1DV[DTHDIS] field description (see *Table 313*).

Note: *f_{pll0_phi1} is the reference clock generated by PLL0 for PLL1.*

When programming the PLLs, user software must not violate the maximum system clock frequency or max/min VCO frequency specification for PLL0 or PLL1 (refer to the device datasheet). Furthermore, the PLL0DV[PREDIV] value must not be set to any value that causes the input frequency to the phase detector of analog PLL blocks to go below the prescribed ranges.

The lock signal and PLL*n*SR[LOCK] flag are immediately negated if the fields of PLL*n*DV are changed without powering down the analog PLLs.

When any of these events occur, an internal timer is initialized to count 64 cycles of the PLL input clock. During this period (64 cycles and a few extra clock cycles for synchronization, for example, 64 to 72 cycles), the PLL*n*SR[LOCK] flag is held negated. After the timer expires, the PLL*n*SR[LOCK] flag reflects the value coming from the PLL lock detection circuitry. To prevent an immediate reset, the PLL*n*CR[LOLRE] bit of the respective PLLs must be cleared before doing any of the above operations.

Note: *The PLL must be powered down and powered up by configuring MC_ME_<mode>_MC before PLL0CR[PREDIV], PLL0CR[MFD], PLL1CR[MFD] or the input reference clock values are modified.*

The recommended procedure to program the PLLs and engage normal mode is shown in *Section 25.7*.

25.6.3 Loss of lock

The PLL digital interface registers provide the flexibility to select whether to generate an interrupt assert system reset or do nothing in the event that the PLL loses lock according to the available PLL*n*CR options.

Loss of lock reset and interrupt are only generated when the PLL is operating in normal mode. The PLLDIG_PLL*n*SR[LOLF] bits are not asserted by a loss of lock condition detected during Bypass mode, although going to Bypass mode from normal mode does not automatically clear the flag if it was previously asserted while the PLL was in normal mode.

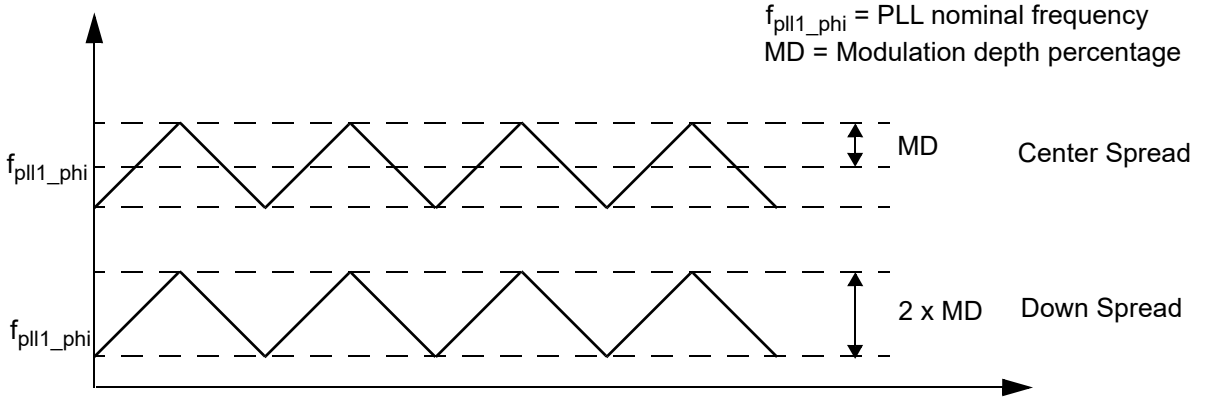
The lock indication from the analog PLL is synchronized and stored in the status register. When the analog PLL loses lock or regains lock, it will be immediately indicated in the status register (after synchronization time).

A pair of counters monitor the reference and feedback clocks to determine when the system has acquired frequency lock. Once the PLL has locked, the counters continue to monitor the reference and feedback clocks and report whether the PLL has lost lock. The lock status is indicated in the status register.

25.6.4 Frequency modulation

Frequency modulation uses a triangular profile as shown in *Figure 216*. The modulation frequency and depth are controlled using PLL1FM[MODPRD] and PLL1FM[INCSTP].

Figure 216. Triangular frequency modulation



Note: The device maximum operating frequency includes the frequency modulation. If center modulation is selected, the fsys must be below fmax by MD percentage such that fmax = fsys(1 + MD%). Please refer to the Analog PLL Chapter for more details.

The following equations define how to calculate PLL1FM[MODPRD] (*Equation 14*) and PLL1FM[INCSTP] (*Equation 15*) based on the frequency of the feedback divider (f_{pll1_ref}), the modulation frequency (f_{mod}) and the modulation depth percentage (MD):

Equation 14

$$PLL1FM[MODPRD] = round\left(\frac{f_{pll1_ref}}{4 \times f_{mod}}\right)$$

Equation 15

$$PLL1FM[INCSTP] = round\left(\frac{(2^{15} - 1) \times MD \times PLL1DV[MFD]}{100 \times 5 \times PLL1FM[MODPRD]}\right)$$

PLL1FM[MODPRD] and PLL1FM[INCSTP] are subject to the following restriction:

Equation 16

$$(PLL1FM[MODPRD] \times PLL1FM[INCSTP]) < 2^{15}$$

Because of the above rounding operations, the effective modulation depth applied to the FMPLL is given by the following formula:

Equation 17

$$\text{ModulationDepth} = \text{round}\left(\frac{\text{PLL1FM[MODPRD]} \times \text{PLL1FM[INCSTP]} \times 100 \times 5}{(2^{15} - 1) \times \text{PLL1DV[MFD]}}\right)$$

As an example, suppose the following configuration:

- Input frequency: 40 MHz
- Loop divider (PLL1DV[MFD]): 20
- Input divider: 1
- Output divider RFD: 5
- VCO frequency: 40 MHz × 20 = 800 MHz
- PLL output frequency: 800 MHz / 2*RFD = 800 MHz / 10 = 80 MHz
- Center spread (PLL1FM[MODSEL] = 0)
- Modulation frequency: 200 kHz
- Modulation depth: ± 1.9% (3.8% peak-to-peak)
- PLL1FM[MODPRD] = Round[(40 × 10^6) / (4 × 200 × 10^3)] = Round[50] = 50
- PLL1FM[INCSTP] = Round[((2^{15} – 1) × 1.9× 20) / (100 × 5 × 50)] = Round[49.8] = 50
- PLL1FM[MODPRD] × PLL1FM[INCSTP] = 50 × 50 = 2500 (which is less than 2^{15})
- MD (quantized) = ((50 × 50 × 100 × 5) / ((2^{15} – 1) × 20) = 1.90740%

In this example, the modulation error is 0.00740%.

The FM parameters do not get reset when PLL0 looses its lock. These parameters get reset on destructive reset. Whenever PLL0 gets relocked FM modulation starts with the previous values. The sequence for reprogramming the FM is:

1. Power down PLL1[i]
2. Program the PLL1FM while PLL1 is powered down
3. Power up the PLL1[j]

25.7 Initialization information

Coming out of reset the PLLs are disabled per the DRUN mode configuration register, MC_ME_DRUN_MC. The recommended procedure to program the PLLs and enter normal mode is:

i. By writing MC_ME_<*mode*>_MC[PLL1ON] = 0b.

j. By writing MC_ME_<*mode*>_MC[PLL1ON] = 1b.

1. Configure PLL0 and related modules.

   Program PLL0DV[PREDIV], PLL0DV[RFDPHI1], PLL0DV[MFD] and PLL0DV[RFDPHI].

   a) With PLL0 disabled, program PLL0 clock source in MC_CGM_AC3_SC[SELCTL]. Default source is 16 MHz IRCOSC. Write MC_CGM_AC3_SC[SELCTL] = 1 to select XOSC as source.

   b) Program PLL0DV[PREDIV], PLL0DV[RFDPHI1], PLL0DV[MFD] and PLL0DV[RFDPHI].

   c) If desired, modify the XOSC_CTL[EOCV] to adjust the external oscillator stabilization count, used when the external oscillator is turned on (by the MC_ME).

   d) If necessary, modify the CMU frequency meter values (CMU_MDR and CMU_FDR) used to monitor the XOSC frequency. XOSC frequency is compared to the IRCOSC source when XOSC is turned on.

2. Turn on XOSC and PLL0.

   a) Configure a mode configuration for turning on PLL0 and XOSC. In a mode configuration register (for example, MC_ME_DRUN_MC) set MC_ME_<mode>_MC[XOSCON] = 1 and MC_ME_<*mode*>_MC[PLL0ON] = 1. If desired, also set MC_ME_<*mode*>_MC[SYSCLK] = 2 for this new mode configuration to use PLL0 (PLL0 PHI output) as the sysclk.

   b) Enter that mode by two writes to MC_ME_CTRL register to enter that mode. This is required even if re-entering the same mode.

3. Wait for the mode transition to complete.

   a) Wait for the mode transition to complete by polling MC_ME_GS[S_MTRANS] or enabling an interrupt for flag MC_ME_IS[I_MTC]. A timer, even if it is the watchdog, should be used to make sure the mode transition completes. Mode transition will NOT complete until:

   - XOSC counter expires (if the new mode configuration changes MC_ME_<*mode*>_MC[XOSCON] to 1), and

   - PLL is locked (if the new mode configuration changes MC_ME_<*mode*>_MC[PLL0ON] = 1).

   b) Confirm the desired target mode was entered by checking the status of MC_ME_GS[S_CURRENT_MODE].

4. Configure PLL1 and related modules

   a) With PLL1 disabled, program PLL1 clock source in MC_CGM_AC4_SC[SELCTL]. Default is MC_CGM_AC4_SC[SELCTL] = which selects XOSC.

   b) Program PLL1DV[MFD] and PLL1DV[RFDPHI].

   c) If required, program the PLL1FM register with the desired frequency modulation parameters and enable FM modulation, PLL1FM[MODEN] = 1. The PLL1FM register must not be written after PLL1 is enabled and operating in normal mode.

5. Turn on PLL1

   a) Configure a mode configuration for turning on PLL1, and keeping XOSC and PLL0 on. In a mode configuration register (for example, MC_ME_DRUN_MC), initialize MC_ME_<*mode*>_MC[XOSCON] = 1, MC_ME_<*mode*>_MC[PLL0ON] = 1 and MC_ME_<*mode*>_MC[PLL1ON] = 1. If desired, also set

MC_ME_<*mode*>_MC[SYSCLK] = 4 for this new mode configuration to use PLL1 as the sysclk.

b) Enter that mode by two writes to ME_CTRL register to enter that mode. This is required even if re-entering the same mode.

6. Wait for the mode transition to complete.

a) Wait for the mode transition to complete by polling MC_ME_GS[S_TRANS] or enabling an interrupt for flag MC_ME_IS[I_MTC]. A timer, even if it is the watchdog, should be used to make sure the mode transition completes. Mode transition will NOT complete until:

XOSC counter expires (if the new mode configuration changes MC_ME_<*mode*>_MC[XOSCON] = 1), and PLL1 is locked (if the new mode configuration changes MC_ME_<*mode*>_MC[PLL1ON] = 1).

b) Confirm the desired target mode was entered by checking the status of MC_ME_GS[S_CURRENT_MODE].

Note: See the Chapter 52: Mode Entry Module (MC_ME) for more details.

26 Clock Monitor Unit (CMU)

26.1 Introduction

The Clock Monitor Unit (CMU), also referred to as Clock Quality Checker or Clock Fault Detector, serves three purposes:

- Measures the frequency of clock sources CLKMT0_RMN with CLKMN0_RMT as the reference clock
- Monitors CLKMN0_RMT frequency with CLKMT0_RMN as reference clock
- Monitors CLKMN1 frequency with CLKMT0_RMN as reference clock and also detects if the monitored clock frequency leaves an upper or lower frequency boundary

One of the tasks is to supervise the integrity of the various clock sources on the chip, for example CLKMN0_RMT or CLKMN1. If the monitored clock frequency is less than the reference clock, or it violates an upper or lower frequency boundary, the CMU detects and reports this event. These events signal the FCCU, which can take necessary corrective actions as its configuration dictates.

The CMU can monitor CLKMN0_RMT, which must have a frequency higher than that of CLKMT0_RMN divided by the division factor shown in CMU_CSR[RCDIV], and reports this event. The CMU can also monitor CLKMN1 and generate events when the CLKMN1 frequency is less than the CLKMT0_RMN frequency divided by four or if CLKMN1 leaves an upper or lower frequency boundary. The upper and lower frequency boundaries are defined by the CMU High Frequency Reference Register (CMU_HFREFR) and CMU Low Frequency Reference Register (CMU_LFREFR).

The second task of the CMU is to provide a frequency meter, which allows measuring the frequency of one clock source against a reference clock. This is useful to allow the calibration of the metered clocks (such as CLKMT0_RMN), as well as to be able to correct/calculate the time deviation of a counter that is clocked by the metered clocks.

Note: See the Chapter 24: Clocking for the number of instances of the CMU in this device.

26.1.1 Main features

- CLKMT0_RMN frequency measurement with CLKMN0_RMT as the reference clock.
- Event generation for various failures detected inside monitoring unit.

26.2 Block diagram

The block diagram of the CMU module(s) is shown in the *Chapter 24: Clocking*.

26.3 Signals

Table 314 describes the signals on the boundary of the CMU (in alphabetical order).

Table 314. Signal description

Signal	I/O	Description
CLKMN0_RMT	I	Monitored Clock Signal 0/Metered Clock Signal Reference: Receives a clock signal that the CMU compares to a specified low-limit frequency to determine whether the frequency of the clock signal is greater than the specified limit. Also provides a reference clock signal for all metered clock signals.
CLKMN1	I	Monitored Clock Signal 1: Receives a clock signal that the CMU compares to specified low-limit and high-limit frequencies to determine whether the frequency of the clock signal is between the specified limits.
CLKMT0_RMN	I	Metered Clock Signal 0/Monitored Clock Signal Reference: Receives a clock signal that the CMU measures against a reference clock frequency. Also provides a reference clock signal for all monitored clock signals.

Note: See the "Chapter 24: Clocking for device specific clock sources of each CMU.

26.4 Memory map and register definition

This section describes in address order all the CMU registers. Each description includes a standard register diagram with an associated figure number. The CMU memory map is listed in *Table 315*.

Table 315. CMU memory map

Address offset	Register	Location
0x0000	CMU Control Status Register (CMU_CSR)	*Section 26.4.1.1*
0x0004	CMU Frequency Display Register (CMU_FDR)	*Section 26.4.1.2*
0x0008	CMU High Frequency Reference Register CLKMN1 (CMU_HFREFR)	*Section 26.4.1.3*
0x000C	CMU Low Frequency Reference Register CLKMN1 (CMU_LFREFR)	*Section 26.4.1.4*
0x0010	CMU Interrupt Status Register (CMU_ISR)	*Section 26.4.1.5*
0x0014–0x0017	Reserved	
0x0018	CMU Measurement Duration Register (CMU_MDR)	*Section 26.4.1.6*

Note: See Chapter 24: Clocking for register and field availability details.

26.4.1 Register descriptions

26.4.1.1 CMU Control Status Register (CMU_CSR)

Offset: 0x0000 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	SFM(1)	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	CKSEL1(1)		0	0	0	0	0	RCDIV(1)		CME
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. Not all CMU blocks will utilize this feature. See Clocking chapter for device specific CMU implementation details.

Figure 217. CMU Control Status Register (CMU_CSR)

Table 316. CMU_CSR field descriptions

Field	Description
8 SFM	Start Frequency Measure The software can only set this bit to start a clock frequency measure. It is reset by hardware when the measure is ready in the CMU_FDR register. MDR[MD] must be written before enabling frequency measurement (CSR[SFM] = 1). Do not write MDR[MD] while CSR[SFM] = 1. Software should only read the value of CMU_FDR[FD] bits once the hardware has cleared the CMU_CSR[SFM] bit. 0 Frequency measurement is completed or not yet started. 1 Frequency measurement is not completed.
22:23 CKSEL1	Frequency measure clock selection bit CKSEL1 selects the clock to be measured by the frequency meter. This only effects CMU instances that utilizes clock metering. Not all CMU blocks will utilize this feature. See the “Clocking” chapter for device specific CMU implementation details. 00 CLKMT0_RMN is selected. 01 Reserved 10 Reserved. 11 CLKMT0_RMN is selected.

Table 316. CMU_CSR field descriptions (continued)

Field	Description
29:30 RCDIV	CLKMT0_RMN division factor These bits specify the CLKMT0_RMN division factor. The output clock frequency is f_{CLKMT0_RMN} divided by the factor 2^{RCDIV}. This output clock is used as reference clock to compare with CLKMN0_RMT for crystal clock monitor feature. Ensure clock source for CLKMN0_RMT is off before writing CMU_CSR[RCDIV]. The clock division coding is as follows: 00 CLKMT0_RMN divided by 1 (No division). 01 CLKMT0_RMN divided by 2. 10 CLKMT0_RMN divided by 4. 11 CLKMT0_RMN divided by 8.
31 CME	CLKMN1 monitor enable CMU_HFREF and CMU_LFREF registers must be written before enabling the clock monitoring (CSR[CME] =1) 0 CLKMN1 monitor is disabled. 1 CLKMN1 monitor is enabled.

26.4.1.2 CMU Frequency Display Register (CMU_FDR)

The FDR is used to determine the measured frequency being monitored by the CMU.

Offset: 0x0004 Access: User read-only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	FD			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	FD															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 218. CMU Frequency Display Register (CMU_FDR)

Table 317. CMU_FDR field descriptions

Field	Description
12:31 FD	Measured frequency bits This register displays the measured frequency (f_{sel}) with respect to the reference clock (f_{CLKMN0_RMT}). The measured value is given by the following formula: $f_{sel} = (f_{CLKMN0_RMT} \times CMU_MDR[MD]) / CMU_FDR[FD]$

26.4.1.3 CMU High Frequency Reference Register CLKMN1 (CMU_HFREFR)

The HFREFR is configured for the high frequency reference that the CMU will use for comparing against the monitored clock. *Figure 219* and *Table 318* show the CMU_HFREFR register.

Offset: 0x0008 Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	HFREF											
W																
Reset	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1

Figure 219. CMU High Frequency Reference Register CLKMN1 (CMU_HFREFR)

Table 318. CMU_HFREFR field descriptions

Field	Description
20:31 HFREF	High Frequency reference value These bits determine the high reference value for the CLKMN1 frequency. The reference value is given by: (HFREF/16) × (f_{CLKMT0_RMN}/4).

26.4.1.4 CMU Low Frequency Reference Register CLKMN1 (CMU_LFREFR)

The LFREFR is configured for the low frequency reference that the CMU will use for comparing against the monitored clock. *Figure 220* and *Table 319* show the CMU_LFREFR register.

Offset 0x000C Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	LFREF											
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 220. CMU Low Frequency Reference Register CLKMN1 (CMU_LFREFR)

Table 319. CMU_LFREFR_0 field descriptions

Field	Description
20:31 LFREF	Low Frequency reference value These bits determine the low reference value for the CLKMN1 frequency. The reference value is given by: (LFREF/16) × (f_{CLKMT0_RMN}/4).

26.4.1.5 CMU Interrupt Status Register (CMU_ISR)

Offset: 0x0010 Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	FHHI	FLLI	OLRI(1)
W													—(2)	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. Not all CMU blocks will utilize this feature. See Clocking chapter for device specific CMU implementation details.
2. Field is read only.

Figure 221. CMU Interrupt Status Register (CMU_ISR)

Table 320. CMU_ISR field descriptions

Field	Description
29 FHHI	CLKMN1 frequency higher than high reference event status This bit is set by hardware when CLKMN1 frequency becomes higher than HFREF value and CLKMN1 is 'ON' as signaled by the MC_ME. It can be cleared by software by writing '1'. 0 No FHH event 1 FHH event occurred

Table 320. CMU_ISR field descriptions (continued)

Field	Description
30 FLLI	CLKMN1 frequency less than low reference event status This bit is set by hardware when CLKMN1 frequency becomes lower than LFREF value and CLKMN1 is 'ON' as signaled by the MC_ME. It can be cleared by software by writing '1'. 0 No FLL event 1 FLL event occurred
31 OLRI	Oscillator frequency less than f_{CLKMT0_RMN} / 2^{RCDIV} event status This bit is set by hardware when the f_{CLKMN0_RMT} is less than f_{CLKMT0_RMN} / 2^{RCDIV} frequency and CLKMN0_RMT is 'ON' as signaled by the MC_ME. It can be cleared by software by writing '1'. 0 No OLR event 1 OLR event occurred

Note: All the flags in CMU_ISR are set asynchronously. This register must be read only after an interrupt (safe mode entry) is triggered by the CMU as confirmed by MC_RGM. Otherwise, the read access on this register may fetch an incorrect value.

Before entering low-power stop modes, all clock frequency measurements in CMU should be disabled. Failing to do so may result in spurious interrupts.

26.4.1.6 CMU Measurement Duration Register (CMU_MDR)

Offset: 0x0018 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	MD			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	MD															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 222. CMU Measurement Duration Register (CMU_MDR)

Table 321. CMU_MDR field descriptions

Field	Description
12:31 MD	Measurement duration bits This field displays the measurement duration in terms of selected clock (CLKMT0_RMN) cycles. This value is loaded in the frequency meter down-counter. The down-counter starts counting when CMU_CSR [SFM] = 1.

26.5 Functional description

This section describes the functionality of the CMU.

26.5.1 Frequency meter

The purpose of frequency meter is to evaluate the deviation from the nominal metered source (such as, CLKMT0_RMN) frequencies. This in turn allows either recalibration of these clocks or other timing corrections. Programming of the CMU_CSR[CKSEL1] field is used to select one of metered clocks from a multiplexer that drives a simple Frequency Meter (see the CMU Block Diagram in the Clocking chapter). The reference clock for the Frequency Meter is the CLKMN0_RMT signal. The measurement starts when CMU_CSR[SFM] = 1. The measurement duration is given by the contents of CMU_MDR[MD] in terms of number of clock cycles of the selected metered clock. The CMU_CSR[SFM] bit is cleared by hardware once the frequency measurement is complete and the count is loaded in CMU_FDR[FD]. The frequency of the selected clock (f_{sel}) can be derived from the value loaded in the CMU_FDR as shown in the following equation:

Equation 18 $f_{sel} = (f_{CLKMN0_RMT} \times MDR[MD]) / FDR[FD]$

26.5.2 CLKMN0_RMT supervisor

- The CMU writes 1 to CMU_ISR[OLRI].
- The CMU asserts the OLR signal.

26.5.3 CLKMN1 supervisor

The frequency of CLKMN1(f_{CLKMN1}) can be monitored by programming CMU_CSR[CME] = 1. CLKMN1 monitoring starts as soon as CMU_CSR[CME] = 1. This monitor can be disabled at any time by programming CMU_CSR[CME] = 0.

- The CMU writes 1 to CMU_ISR[FHHI].
- The CMU asserts the FHH signal.

Note: An example of determining the HFREFActual value is as follows. Assume a fCLKMT0_RMN = 16 MHz with an accuracy of +/-5%. In order to monitor fCLKMN1 = 200 MHz, the ideal HFREFIdeal = 800. The actual HFREF value will be 842 when the accuracy is taken into consideration (HFREFActual = (HFREFIdeal + 2) × 1.05).

- The CMU writes 1 to CMU_ISR[FLLI].
- The CMU asserts the FLL signal.

Note: The low reference value must be programmed considering the following factors:

- *2 cycles of f_{CLKMN1} which is the built-in tolerance of the monitor implementation*
- *Frequency variation % of f_{CLKMT0_RMN} across PVT*

For example, assuming f_{CLKMT0_RMN} is 16 MHz with +/-5% variation, in order to monitor the f_{CLKMN1} (200 MHz), the ideal $LFREF_{Ideal}$ is 800. The adjusted value considering the tolerances will be 758 ($LFREF_{Actual}$ = ($LFREF_{Ideal}$ - 2) x 0.95).

The minimum value that can be programmed in CMU_LFREFR[LFREF] is 3. The minimum frequency that can be monitored using the CLKMN1 supervisor must be >= f_{CLKMT0_RMN} / 16.

27 Clock Generation Module (MC_CGM)

27.1 Introduction

This chapter describes the Clock Generation Module (MC_CGM) which includes, but is not limited to, the funtionality, pin descriptions, and registers of the MC_CGM module.

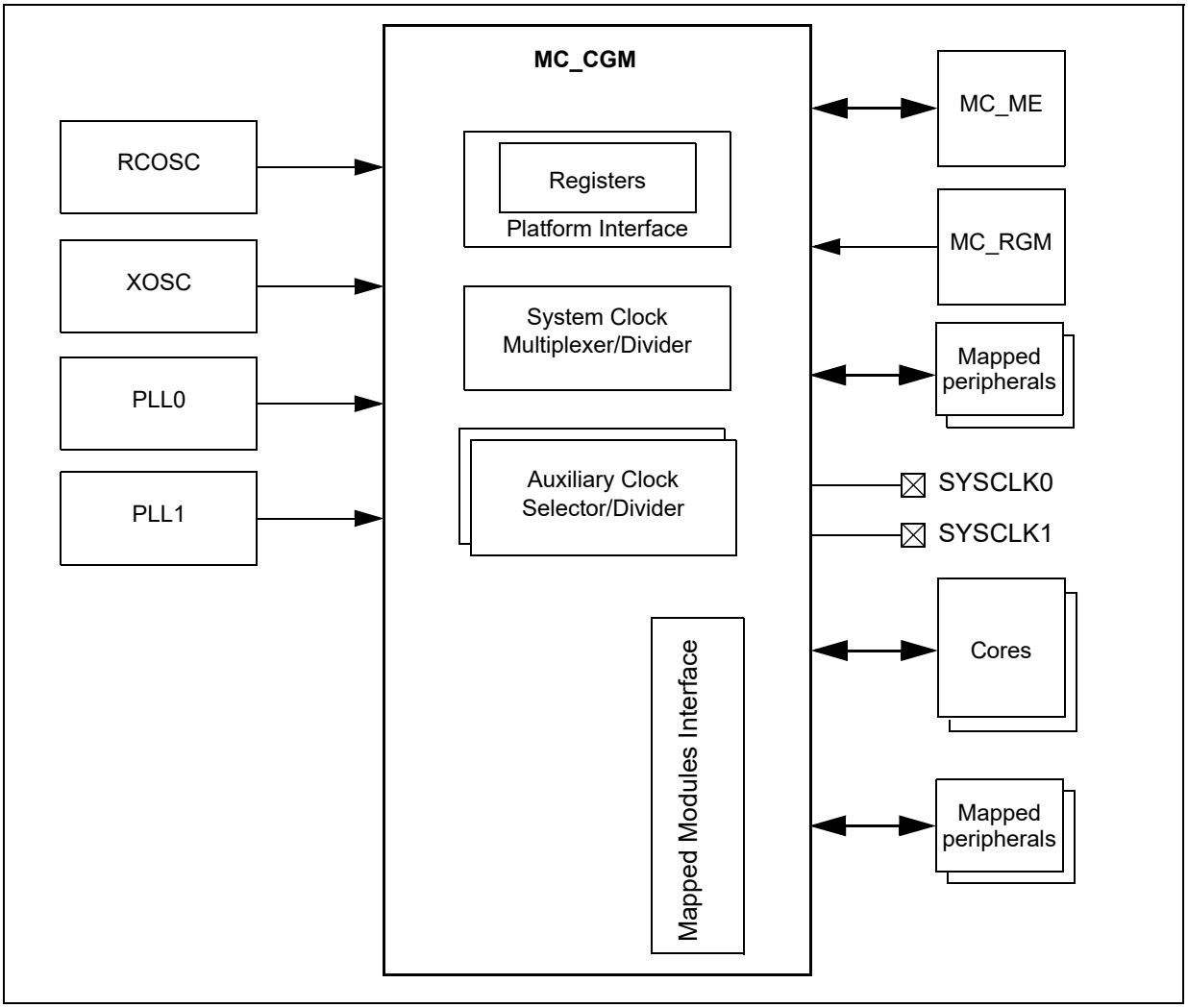
27.1.1 Overview

The Clock Generation Module (MC_CGM) generates reference clocks for all the chip blocks and enables the system clock from one of the available system clock sources. System clock selection is handled in the Mode Entry control module, see the MC_ME chapter for more details.

The addressable memory spaces of system and peripheral clock are accessed through the MC_CGM memory space.

Figure 223 shows the MC_CGM block diagram.

Figure 223. MC_CGM block diagram



27.1.2 Features

The MC_CGM:

- Generates system and peripheral clocks
- Selects and enables/disables the system clock supply from system clock sources according to MC_ME control
- Performs progressive system clock frequency change depending on MC_ME mode configuration
- Contains a set of registers to control clock dividers for divided clock generation
- Supports multiple clock sources and maps their address spaces to its memory map
- Guarantees glitch-free clock transitions when changing the system clock selection
- Supports 8, 16, and 32-bit wide read/write accesses

27.2 External signal description

The MC_CGM delivers output clocks to the SYSCLK0 and SYSCLK1 pins for off-chip use and/or observation.

27.3 Memory map and register definition

Table 322. MC_CGM register description

Offset Address	Description	Location
0x0000	PCS Switch Duration (CGM_PCS_SDUR)	*Section 27.3.1.1*
0x0004	PCS Divider Change for XOSC (CGM_PCS_DIVC1)	*Section 27.3.1.2*
0x0008	PCS Divider End for XOSC (CGM_PCS_DIVE1)	*Section 27.3.1.3*
0x000C	PCS Divider Start for XOSC (CGM_PCS_DIVS1)	*Section 27.3.1.5*
0x0010	PCS Divider Change for PLL0 PHI (CGM_PCS_DIVC2)	*Section 27.3.1.5*
0x0014	PCS Divider End for PLL0 PHI (CGM_PCS_DIVE2)	*Section 27.3.1.6*
0x0018	PCS Divider Start for PLL0 PHI (CGM_PCS_DIVS2)	*Section 27.3.1.6*
0x0028	PCS Divider Change for PLL1 PHI (CGM_PCS_DIVC4)	*Section 27.3.1.7*
0x002C	PCS Divider End for PLL1 PHI (CGM_PCS_DIVE4)	*Section 27.3.1.9*
0x0030	PCS Divider Start for PLL1 PHI (CGM_PCS_DIVS4)	*Section 27.3.1.9*
0x00D0	System Clock Divider Ratio Change Register (CGM_SC_DIV_RC)	*Section 27.3.1.11*
0x00D4	Divider Update Type Register (CGM_DIV_UPD_TYPE)	*Section 27.3.1.12*
0x00D8	Divider Update Trigger Register (CGM_DIV_UPD_TRIG)	*Section 27.3.1.13*
0x00DC	Divider Update Status Register (CGM_DIV_UPD_STAT)	*Section 27.3.1.14*
0x00E4	System Clock Select Status (CGM_SC_SS)	*Section 27.3.1.15*
0x00E8	System Clock Divider Configuration 0 (CGM_SC_DC0)	*Section 27.3.1.16*
0x00EC	System Clock Divider Configuration 1 (CGM_SC_DC1)	*Section 27.3.1.17*
0x00F0	System Clock Divider Configuration 2 (CGM_SC_DC2)	*Section 27.3.1.18*
0x00F4	System Clock Divider Configuration 3 (CGM_SC_DC3)	*Section 27.3.1.19*
0x0100	Aux Clock 0 Select Control (CGM_AC0_SC)	*Section 27.3.1.20*
0x0104	Aux Clock 0 Select Status (CGM_AC0_SS)	*Section 27.3.1.21*
0x0108	Aux Clock 0 Divider Configuration 0 (CGM_AC0_DC0)	*Section 27.3.1.22*
0x010C	Aux Clock 0 Divider Configuration 1 (CGM_AC0_DC1)	*Section 27.3.1.23*
0x0110	Aux Clock 0 Divider Configuration 2 (CGM_AC0_DC2)	*Section 27.3.1.24*
0x0120	Aux Clock 1 Select Control (CGM_AC1_SC)	*Section 27.3.1.25*
0x0124	Aux Clock 1 Select Status (CGM_AC1_SS)	*Section 27.3.1.26*
0x0128	Aux Clock 1 Divider Configuration 0 (CGM_AC1_DC0)	*Section 27.3.1.27*
0x0140	Aux Clock 2 Select Control (CGM_AC2_SC)	*Section 27.3.1.28*

Table 322. MC_CGM register description (continued)

Offset Address	Description	Location
0x0144	Aux Clock 2 Select Status (CGM_AC2_SS)	*Section 27.3.1.29*
0x0160	Aux Clock 3 Select Control (CGM_AC3_SC)	*Section 27.3.1.30*
0x0164	Aux Clock 3 Select Status (CGM_AC3_SS)	*Section 27.3.1.31*
0x0180	Aux Clock 4 Select Control (CGM_AC4_SC)	*Section 27.3.1.32*
0x0184	Aux Clock 4 Select Status (CGM_AC4_SS)	*Section 27.3.1.33*
0x0188	Aux Clock 4 Divider Configuration 0 (CGM_AC4_DC0)	*Section 27.3.1.34*
0x018C	Aux Clock 4 Divider Configuration 1 (CGM_AC4_DC1)	*Section 27.3.1.35*
0x0190	Aux Clock 4 Divider Configuration 2 (CGM_AC4_DC2)	*Section 27.3.1.36*
0x0194	Aux Clock 4 Divider Configuration 3 (CGM_AC4_DC3)	*Section 27.3.1.37*
0x0198	Aux Clock 4 Divider Configuration 4 (CGM_AC4_DC4)	*Section 27.3.1.38*
0x019C	Aux Clock 4 Divider Configuration 5 (CGM_AC4_DC5)	*Section 27.3.1.39*
0x01A0	Aux Clock 5 Select Control (CGM_AC5_SC)	*Section 27.3.1.40*
0x01A4	Aux Clock 5 Select Status (CGM_AC5_SS)	*Section 27.3.1.41*
0x01A8	Aux Clock 5 Divider Configuration 0 (CGM_AC5_DC0)	*Section 27.3.1.42*

Note: Any access to unused registers as well as write accesses to read-only registers:

- *Does not change register content*
- *Causes a transfer error*

27.3.1 Register descriptions

Unless otherwise noted, all registers may be accessed as 32-bit words, 16-bit half-words, or 8-bit bytes.

The bytes are ordered according to big endian.

For example, the CGM_PCS_DIVC1 register RATE bits may be accessed as a word at address offset 0x0004, as a half-word at address offset 0x0006, or as a byte at address offset 0x0007.

27.3.1.1 PCS Switch Duration Register (CGM_PCS_SDUR)

This register contains the progressive system clock switching duration for each step. See *Section 27.4.1.2: Progressive System Clock Switching* for details on how to set this value.

Address: 0x0000 Access: User read/write, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7
R W	SDUR							
Reset	0	0	0	0	0	0	0	0

Figure 224. PCS Switch Duration Register (CGM_PCS_SDUR)

Table 323. CGM_PCS_SDUR field descriptions

Field	Description
0:7 SDUR	Switch Duration This value defines the duration of one PCS clock switch step in terms of 16 MHz Internal RC oscillator (RCOSC) cycles.

27.3.1.2 PCS Divider Change 1 Register (CGM_PCS_DIVC1)

This register relates to the external crystal oscillator (XOSC).

When the system clock source switches to this clock or from this clock on ramp-up or ramp-down, the register defines the rate of frequency change and the initial change value for progressive system clock switching.

See *Section 27.4.1.2: Progressive System Clock Switching* for details on how to set these values.

Note: Byte write accesses are not allowed for the INIT field of this register. Such attempts do not result in an exception, however the new value is not loaded.

Address: 0x0004 Access: User read/write, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	INIT															
Reset	0	0	0	0	0	0	1	1	1	1	1	0	0	1	1	1

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	RATE							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 225. PCS Divider Change 1 Register (CGM_PCS_DIVC1)

Table 324. CGM_PCS_DIVC1 field descriptions

Field	Description
0:15 INIT	Divider Change Initial Value The initial change value of the clock divider for the clock ramp-up phase when switching to this clock.
24:31 RATE	Divider Change Rate Controls the change value of the clock divider for the clock ramp-up and ramp-down phase when switching to or from this clock

27.3.1.3 PCS Divider End 1 Register (CGM_PCS_DIVE1)

This register relates to the external crystal oscillator (XOSC).

When switching the system clock source away from this clock on ramp-down, the register defines the final division value for progressive system clock switching.

See *Section 27.4.1.2* for details on how to set this value.

Note: Byte and half-word write accesses are not allowed for this register. Such attempts do not result in an exception, however the new value is not loaded.

Address: 0x0008 Access: User read/write, Supervisor read/write, Test read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	DIVE			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	DIVE															
W																
Reset	0	0	0	0	0	0	1	1	1	1	1	0	0	1	1	1

Figure 226. PCS Divider End 1 Register (CGM_PCS_DIVE1)

Table 325. CGM_PCS_DIVE1 field descriptions

Field	Description
12:31 DIVE	Divider End Value The clock divider end value for the clock ramp-down phase when switching away from this clock.

27.3.1.4 PCS Divider Start 1 Register (CGM_PCS_DIVS1)

This register relates to the external crystal oscillator (XOSC).

When switching the system clock source to this clock on ramp-up, the register defines the initial division value for progressive system clock switching.

See *Section 27.4.1.2: Progressive System Clock Switching* for details on how to set these values.

Note: Byte and half-word write accesses are not allowed for this register. Such attempts do not result in an exception, however the new value is not loaded.

Address: 0x000C Access: User read/write, Supervisor read/write, Test read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	DIVS			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	DIVS															
W																
Reset	0	0	0	0	0	0	1	1	1	1	1	0	0	1	1	1

Figure 227. PCS Divider Start 1 Register (CGM_PCS_DIVS1)

Table 326. CGM_PCS_DIVS1 field descriptions

Field	Description
12:31 DIVS	Divider Start Value The start value of the clock divider for the clock ramp-up phase when switching to this clock.

27.3.1.5 PCS Divider Change 2 (CGM_PCS_DIVC2)

This register relates to PLL0_PHI.

When the system clock source switches to this clock or from this clock on ramp-up or ramp-down, the register defines the rate of frequency change and the initial change value for progressive system clock switching.

See *Section 27.4.1.2: Progressive System Clock Switching* for details on how to set these values.

Note: Byte write accesses are not allowed for the INIT field of this register. Such attempts do not result in an exception, however the new value is not loaded.

Address: 0x0010 Access: User read/write, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	INIT															
Reset	0	0	0	0	0	0	1	1	1	1	1	0	0	1	1	1

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	RATE							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 228. PCS Divider Change 2 (CGM_PCS_DIVC2)

Table 327. CGM_PCS_DIVC2 field descriptions

Field	Description
0:15 INIT	Divider Change Initial Value The initial change value of the clock divider for the clock ramp-up phase when switching to this clock.
24:31 RATE	Divider Change Rate Controls the change value of the clock divider for the clock ramp-up and ramp-down phase when switching to or from this clock.

27.3.1.6 PCS Divider End 2 Register (CGM_PCS_DIVE2)

This register relates to PLL0_PHI.

When switching the system clock source away from this clock on ramp-down, the register defines the final division value for progressive system clock switching.

See *Section 27.4.1.2: Progressive System Clock Switching* for details on how to set this value.

Note: Byte and half-word write accesses are not allowed for this register. Such attempts do not result in an exception, however the new value is not loaded.

Address: 0x0014 Access: User read/write, Supervisor read/write, Test read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	DIVE			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	DIVE															
W																
Reset	0	0	0	0	0	0	1	1	1	1	1	0	0	1	1	1

Figure 229. PCS Divider End 2 Register (CGM_PCS_DIVE2)

Table 328. CGM_PCS_DIVE2 field descriptions

Field	Description
12:31 DIVE	Divider End Value The clock divider end value for the clock ramp-down phase when switching away from this clock.

27.3.1.7 PCS Divider Start 2 Register (CGM_PCS_DIVS2)

This register relates to PLL0_PHI.

When switching the system clock source to this clock on ramp-up, the register defines the initial division value for progressive system clock switching.

See *Section 27.4.1.2: Progressive System Clock Switching* for details on how to set these values.

Note: Byte and half-word write accesses are not allowed for this register. Such attempts do not result in an exception, however the new value is not loaded.

Address: 0x0018 Access: User read/write, Supervisor read/write, Test read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	DIVS			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	DIVS															
W																
Reset	0	0	0	0	0	0	1	1	1	1	1	0	0	1	1	1

Figure 230. PCS Divider Start 2 Register (CGM_PCS_DIVS2)

Table 329. CGM_PCS_DIVS2 field descriptions

Field	Description
12:31 DIVS	Divider Start Value The start value of the clock divider for the clock ramp-up phase when switching to this clock.

27.3.1.8 PCS Divider Change 4 Register (CGM_PCS_DIVC4)

This register relates to PLL1_PHI.

When the system clock source switches to this clock or from this clock on ramp-up or ramp-down, the register defines the rate of frequency change and the initial change value for progressive system clock switching.

See *Section 27.4.1.2: Progressive System Clock Switching* for details on how to set these values.

Note: Byte write accesses are not allowed for the INIT field of this register. Such attempts do not result in an exception, however the new value is not loaded.

Address: 0x0028 Access: User read/write, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	INIT															
Reset	0	0	0	0	0	0	1	1	1	1	1	0	0	1	1	1

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	RATE							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 231. PCS Divider Change 4 Register (CGM_PCS_DIVC4)

Table 330. CGM_PCS_DIVC4 field descriptions

Field	Description
0:15 INIT	Divider Change Initial Value The initial change value of the clock divider for the clock ramp-up phase when switching to this clock.
24:31 RATE	Divider Change Rate Controls the change value of the clock divider for the clock ramp-up and ramp-down phase when switching to or from this clock.

27.3.1.9 PCS Divider End 4 Register (CGM_PCS_DIVE4)

This register relates to PLL1_PHI.

When switching the system clock source away from this clock on ramp-down, the register defines the final division value for progressive system clock switching.

See *Section 27.4.1.2: Progressive System Clock Switching* for details on how to set this value.

Note: Byte and half-word write accesses are not allowed for this register. Such attempts do not result in an exception, however the new value is not loaded.

Address: 0x002C Access: User read/write, Supervisor read/write, Test read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	DIVE			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	DIVE															
W																
Reset	0	0	0	0	0	0	1	1	1	1	1	0	0	1	1	1

Figure 232. PCS Divider End 4 Register (CGM_PCS_DIVE4)

Table 331. CGM_PCS_DIVE4 field descriptions

Field	Description
12:31 DIVE	Divider End Value The clock divider end value for the clock ramp-down phase when switching from this clock.

27.3.1.10 PCS Divider Start 4 Register (CGM_PCS_DIVS4)

This register relates to PLL1_PHI.

When switching the system clock source to this clock on ramp-up, the register defines the initial division value for progressive system clock switching.

See *Section 27.4.1.2: Progressive System Clock Switching* for details on how to set these values.

Note: Byte and half-word write accesses are not allowed for this register. Such attempts do not result in an exception, however the new value is not loaded.

Address: 0x0030 Access: User read/write, Supervisor read/write, Test read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	DIVS			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	DIVS															
W																
Reset	0	0	0	0	0	0	1	1	1	1	1	0	0	1	1	1

Figure 233. PCS Divider Start 4 Register (CGM_PCS_DIVS4)

Table 332. CGM_PCS_DIVS4 field descriptions

Field	Description
12:31 DIVS	Divider Start Value The start value of the clock divider for the clock ramp-up phase when switching to this clock.

27.3.1.11 System Clock Divider Ratio Change Register (CGM_SC_DIV_RC)

This register selects whether the system clock divider ratios will change from the current configuration to the configuration in the next update. If the value of SYS_DIV_RATIO_CHNG is ‘1’ at the time of a write to the CGM_DIV_UPD_TRIG register and a pre-loaded system clock divider update is pending, the crossbar is halted while the system clock divider update takes place. If the value of SYS_DIV_RATIO_CHNG is ‘0’, the crossbar switch is not halted. This register has no effect if the system clock divider update is configured in the CGM_DIV_UPD_TYPE register to be immediate.

An example of a divider ratio not changing is divider 0 changes from divide by 1 to divide by 2, and divider 1 changes from divide by 2 to divide by 4. The ratio is 1:2 in both configurations. An example of a divider ratio changing is divider 0 changes from divide by 1 to divide by 2, and divider 1 changes from divide by 2 to divide by 6. The ratio changes from 1:2 to 1:3. The value of SYS_DIV_RATIO_CHNG is returned to ‘1’ automatically on each write to the CGM_DIV_UPD_TRIG register.

Caution: Software must never write a ‘0’ to the SYS_DIV_RATIO_CHNG bit if the system clock divider ratios change from the current configuration to the next configuration. This can potentially cause errors in the crossbar which lead to lost data.

Address: 0x00D0 Access: User read/write, Supervisor read/write, Test read

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	SYS_DIV_RATIO_CHN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Figure 234. CGM_SC_DIV_RC

Table 333. CGM_SC_DIV_RC field descriptions

Field	Description
31 SYS_DIV_RATIO_CHNG	System Clock Divider Ratio Change 0 The system clock divider ratios will not change with the next system clock divider configuration update 1 The system clock divider ratios will change with the next system clock divider configuration update

27.3.1.12 Divider Update Type Register (CGM_DIV_UPD_TYPE)

This register selects whether the dividers associated with a clock are updated immediately on writing to the corresponding divider configuration register or only on writing to the divider update trigger register CGM_DIV_UPD_TRIG, after which the pre-loaded configuration in the divider configuration registers will take effect.

Address: 0x00D4 Access: User read/write, Supervisor read/write, Test read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	SYS_UPD_TYPE	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	AUX5_UPD_TYPE	AUX4_UPD_TYPE	0	0	AUX1_UPD_TYPE	AUX0_UPD_TYPE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 235. CGM_DIV_UPD_TYPE

Table 334. CGM_DIV_UPD_TYPE field descriptions

Field	Description
0 SYS_UPD_TYPE	System Clock Divider Update Type 0 The configuration for each system clock divider is updated immediately on writing to the corresponding CGM_SC_DCi register 1 The configuration for each system clock divider is pre-loaded on writing to the corresponding CGM_SC_DCi register, and the pre-loaded configurations of all the system clock dividers are updated on writing to the CGM_DIV_UPD_TRIG register
26 AUX5_UPD_TYPE	Auxiliary Clock 5 Divider Update Type 0 The configuration for each auxiliary clock 5 divider is updated immediately on writing to the corresponding CGM_AC5_DCi register 1 The configuration for each auxiliary clock 5 divider is pre-loaded on writing to the corresponding CGM_AC5_DCi register, and the pre-loaded configurations of all the auxiliary clock 1 dividers are updated on writing to the CGM_DIV_UPD_TRIG register
27 AUX4_UPD_TYPE	Auxiliary Clock 4 Divider Update Type 0 The configuration for each auxiliary clock 4 divider is updated immediately on writing to the corresponding CGM_AC4_DCi register 1 The configuration for each auxiliary clock 4 divider is pre-loaded on writing to the corresponding CGM_AC4_DCi register, and the pre-loaded configurations of all the auxiliary clock 1 dividers are updated on writing to the CGM_DIV_UPD_TRIG register

Table 334. CGM_DIV_UPD_TYPE field descriptions (continued)

Field	Description
30 AUX1_UPD_TYPE	Auxiliary Clock 1 Divider Update Type 0 The configuration for each auxiliary clock 1 divider is updated immediately on writing to the corresponding CGM_AC1_DCi register 1 The configuration for each auxiliary clock 1 divider is pre-loaded on writing to the corresponding CGM_AC1_DCi register, and the pre-loaded configurations of all the auxiliary clock 1 dividers are updated on writing to the CGM_DIV_UPD_TRIG register
31 AUX0_UPD_TYPE	Auxiliary Clock 0 Divider Update Type 0 The configuration for each auxiliary clock 0 divider is updated immediately on writing to the corresponding CGM_AC0_DCi register 1 The configuration for each auxiliary clock 0 divider is pre-loaded on writing to the corresponding CGM_AC0_DCi register, and the pre-loaded configurations of all the auxiliary clock 0 dividers are updated on writing to the CGM_DIV_UPD_TRIG register

27.3.1.13 Divider Update Trigger Register (CGM_DIV_UPD_TRIG)

Writing any value to this register will cause all dividers that have corresponding bits set to '1' in the CGM_DIV_UPD_TYPE register and have pre-loaded configurations to be updated immediately. Reading this register will always return all zeroes.

Address: 0x00D8 Access: User write, Supervisor write, Test read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W	DIV_UPD_TRIGGER[0:15]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0															
W	DIV_UPD_TRIGGER[16:31]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 236. CGM_DIV_UPD_TRIG

Table 335. CGM_DIV_UPD_TRIG field descriptions

Field	Description
0:31 DIV_UPD_TRIGGER	Writing any value to this register will cause all dividers that have corresponding bits set to '1' in the CGM_DIV_UPD_TYPE register and have pre-loaded configurations to be updated immediately. Reading this register will always return all zeroes.

27.3.1.14 Divider Update Status Register (CGM_DIV_UPD_STAT)

This register indicates whether a divider for each clock is in the process of being updated. If the CGM_DIV_UPD_TYPE[i] is '0', CGM_DIV_UPD_STAT[i] is set to '1' immediately on writing to one of the corresponding divider configuration registers. Otherwise,

CGM_DIV_UPD_STAT[i] is set to ‘1’ immediately on writing to the CGM_DIV_UPD_TRIG register if one of the corresponding divider configurations has been pre-loaded. CGM_DIV_UPD_STAT[i] is cleared to ‘0’ as soon as all the corresponding dividers have been updated and the dividers generate the desired frequency.

Address: 0x00DC Access: User read/write, Supervisor read/write, Test read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	SYS_UPD_STAT	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	AUX5_UPD_STAT	AUX4_UPD_STAT	0	0	AUX1_UPD_STAT	AUX0_UPD_STAT
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 237. CGM_DIV_UPD_STAT

Table 336. CGM_DIV_UPD_STAT field descriptions

Field	Description
0 SYS_UPD_STAT	System Clock Divider Update Status. 0 The configuration for at least one system clock divider is not being updated 1 The configuration for at least one system clock divider is being updated
26 AUX5_UPD_STAT	Auxiliary Clock 5 Divider Update Status 0 The configuration for at least one auxiliary clock 5 divider is not being updated 1 The configuration for at least one auxiliary clock 5 divider is being updated
27 AUX4_UPD_STAT	Auxiliary Clock 4 Divider Update Status 0 The configuration for at least one auxiliary clock 4 divider is not being updated 1 The configuration for at least one auxiliary clock 4 divider is being updated
30 AUX1_UPD_STAT	Auxiliary Clock 1 Divider Update Status 0 The configuration for at least one auxiliary clock 1 divider is not being updated 1 The configuration for at least one auxiliary clock 1 divider is being updated
31 AUX0_UPD_STAT	Auxiliary Clock 0 Divider Update Status 0 The configuration for at least one auxiliary clock 0 divider is not being updated 1 The configuration for at least one auxiliary clock 0 divider is being updated

27.3.1.15 System Clock Select Status Register (CGM_SC_SS)

This register provides the current system clock source selection.

Address: 0x00E4 Access: User read, Supervisor read, Test read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	SELSTAT				0	0	0	0	SWTRG			SWIP
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0

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

Figure 238. System Clock Select Status Register (CGM_SC_SS)

Table 337. CGM_SC_SS field descriptions

Field	Description
4:7 SELSTAT	System Clock Source Selection Status This value indicates the current source for the system clock. Unused values are reserved. 0000 RCOSC 0001 XOSC 0010 PLL0 PHI 0100 PLL1 PHI 1111 system clock is disabled
12:14 SWTRG	Switch Trigger cause Indicates the cause for the latest clock source switch. Unused values are reserved. 001 switch after request from MC_ME succeeded 010 switch after request from MC_ME failed due inactive target clock 011 switch after request from MC_ME failed due inactive current clock 100 switch to RCOSC due to SAFE mode request or reset succeeded 101 switch to RCOSC due to SAFE mode request or reset succeeded, but current clock source was inactive
15 SWIP	Switch In Progress 0 clock source switching completed 1 clock source switching in progress

27.3.1.16 System Clock Divider 0 Configuration Register (CGM_SC_DC0)

This register relates to Core and XBAR.

See *Section 27.4.1.4.1: System Clock Divider Synchronization* for details on division value limitations.

Note: Byte and half-word write accesses are not allowed for this register. Such attempts do not result in an exception, however the new value is not loaded.

Address: 0x00E8 Access: User read/write, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DE	0	0	0	0	0	0	0	0	0	0	0	0	DIV		
W																
Reset	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

Figure 239. System Clock Divider 0 Configuration Register (CGM_SC_DC0)

Table 338. CGM_SC_DC0 field descriptions

Field	Description
0 DE	Divider 0 Enable This divider is always enabled. This bit is read-only and always returns a value of ‘1’.
13:15 DIV	Divider 0 Division Value The resultant clock has a period ’DIV + 1’ times that of the system clock.

27.3.1.17 System Clock Divider 1 Configuration Register (CGM_SC_DC1)

This register relates to PBRIDGE.

See *Section 27.4.1.4.1: System Clock Divider Synchronization* for details on division value limitations.

Note: Byte and half-word write accesses are not allowed for this register. Such attempts do not result in an exception, however the new value is not loaded.

Address: 0x00EC Access: User read/write, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DE	0	0	0	0	0	0	0	0	0	0	0	DIV			
W																
Reset	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

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

Figure 240. System Clock Divider 1 Configuration Register (CGM_SC_DC1)

Table 339. CGM_SC_DC1 field descriptions

Field	Description
0 DE	Divider 1 Enable This divider is always enabled. This bit is read-only and always returns a value of ‘1’.
12:15 DIV	Divider 1 Division Value The resultant clock has a period ’DIV + 1’ times that of the system clock.

27.3.1.18 System Clock Divider 2 Configuration Register (CGM_SC_DC2)

This register relates to CLKOUT clock.

See *Section 27.4.1.4.1: System Clock Divider Synchronization* for details on division value limitations.

Note: Byte and half-word write accesses are not allowed for this register. Such attempts do not result in an exception, however the new value is not loaded.

Address: 0x00F0 Access: User read/write, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DE	0	0	0	0	0	0	DIV								
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

Figure 241. System Clock Divider 2 Configuration Register (CGM_SC_DC2)

Table 340. CGM_SC_DC2 field descriptions

Field	Description
0 DE	Divider 2 Enable This divider is disabled by default. This bit is writable.
7:15 DIV	Divider 2 Division Value The resultant clock will have a period ’DIV + 1’ times that of the system clock.

27.3.1.19 System Clock Divider 3 Configuration Register (CGM_SC_DC3)

This register relates to DMA/INTC clock.

See *Section 27.4.1.4.1: System Clock Divider Synchronization* for details on division value limitations.

Note: Byte and half-word write accesses are not allowed for this register. Such attempts do not result in an exception, however the new value is not loaded.

Address: 0x00F4 Access: User read/write, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DE	0	0	0	0	0	0	0	0	0	0	0	DIV			
W																
Reset	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

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

Figure 242. System Clock Divider 3 Configuration Register (CGM_SC_DC3)

Table 341. CGM_SC_DC3 field descriptions

Field	Description
0 DE	Divider 3 Enable This divider is always enabled. This bit is read-only and always returns a value of ‘1’.
12:15 DIV	Divider 3 Division Value The resultant clock will have a period ’DIV + 1’ times that of the system clock.

27.3.1.20 Auxiliary Clock 0 Select Control Register (CGM_AC0_SC)

This register selects the clock source for the following:

- Auxiliary Clock 0 Divider 0: PIT clock.
- Auxiliary Clock 0 Divider 1: DSPI clock.
- Auxiliary Clock 0 Divider 2: Linflex clock.

See *Figure 243* for details.

Address: 0x0100 Access: User read/write, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	SELCTL				0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

Figure 243. Auxiliary Clock 0 Select Control Register (CGM_AC0_SC)

Table 342. CGM_AC0_SC field descriptions

Field	Description
4:7 SELCTL	Auxiliary Clock Source Selection Control Selects the clock source for this auxiliary clock. Unused values are reserved. 0000 RCOSC 0001 XOSC 0010 PLL0 PHI

27.3.1.21 Auxiliary Clock 0 Select Status Register (CGM_AC0_SS)

This register reports the current source clock for Auxiliary Clock 0.

Address: 0x0104 Access: User read, Supervisor read, Test read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	SELSTAT				0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

Figure 244. Auxiliary Clock 0 Select Status Register (CGM_AC0_SS)

Table 343. CGM_AC0_SS field descriptions

Field	Description
4:7 SELSTAT	Auxiliary Clock Source Selection Status Indicates the current clock source for this auxiliary clock. All unused values are reserved. 0000 RCOSC 0001 XOSC 0010 PLL0 PHI

27.3.1.22 Auxiliary Clock 0 Divider 0 Configuration Register (CGM_AC0_DC0)

This register controls the PIT clock

Note: Byte and half-word write accesses are not allowed for this register. Such attempts do not result in an exception, however the new value is not loaded.

Address 0x0108 Access: User read/write, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DE	0	0	0	0	0	0	0	0	0	0	0	DIV			
W																
Reset	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

Figure 245. Auxiliary Clock 0 Divider 0 Configuration Register (CGM_AC0_DC0)

Table 344. CGM_AC0_DC0 field descriptions

Field	Description
0 DE	Divider (and resultant clock) Enable 0 Disable 1 Enable
12:15 DIV	Divider Division Value The resultant clock has a period 'DIV + 1' times the selected auxiliary clock source. If the divider is disabled (DE = '0'), writes to the DIV field are ignored.

27.3.1.23 Auxiliary Clock 0 Divider 1 Configuration Register (CGM_AC0_DC1)

This register controls the DSPI clock.

Note: Byte and half-word write accesses are not allowed for this register. Such attempts do not result in an exception, however the new value is not loaded.

Address: 0x010C Access: User read/write, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DE	0	0	0	0	0	0	0	0	DIV						
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

Figure 246. Auxiliary Clock 0 Divider 1 Configuration Register (CGM_AC0_DC1)

Table 345. CGM_AC0_DC1 field descriptions

Field	Description
0 DE	Divider (and resultant clock) Enable 0 Disable. 1 Enable.
9:15 DIV	Divider Division Value The resultant clock has a period 'DIV + 1' times the selected auxiliary clock source. If the divider is disabled (DE = '0'), writes to the DIV field are ignored.

27.3.1.24 Auxiliary Clock 0 Divider 2 Configuration Register (CGM_AC0_DC2)

This register controls the Linflex clock.

Note: Byte and half-word write accesses are not allowed for this register. Such attempts do not result in an exception, however the new value is not loaded.

Address: 0x0110 Access: User read/write, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DE	0	0	0	0	0	0	0	0	0	0	0	DIV			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

Figure 247. Auxiliary Clock 0 Divider 2 Configuration Register (CGM_AC0_DC2)

Table 346. CGM_AC0_DC2 field descriptions

Field	Description
0 DE	Divider (and resultant clock) Enable 0 Disable. 1 Enable.
12:15 DIV	Divider Division Value The resultant clock has a period 'DIV + 1' times the selected auxiliary clock source. If the divider is disabled (DE = '0'), writes to the DIV field are ignored.

27.3.1.25 Auxiliary Clock 1 Select Control Register (CGM_AC1_SC)

This register is used to select the current clock source for the following clocks:

- Auxiliary Clock 1 Divider 0: CAN Subsystem clock

See *Figure 248* for details.

Address: 0x0120 Access: User read/write, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	SELCTL				0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

Figure 248. Auxiliary Clock 1 Select Control Register (CGM_AC1_SC)

Table 347. CGM_AC1_SC field descriptions

Field	Description
4:7 SELCTL	Auxiliary Clock Source Selection Control Selects the clock source for this auxiliary clock. Unused values are reserved. 0000 RCOSC 0001 XOSC 0010 PLL0 PHI

27.3.1.26 Auxiliary Clock 1 Select Status Register (CGM_AC1_SS)

This register reports the current source clock for Auxiliary Clock 1.

Address: 0x0124 Access: User read, Supervisor read, Test read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	SELSTAT				0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

Figure 249. Auxiliary Clock 1 Select Status Register (CGM_AC1_SS)

Table 348. CGM_AC1_SS field descriptions

Field	Description
4:7 SELSTAT	Auxiliary Clock Source Selection Status Indicates the current clock source for this auxiliary clock. All unused values are reserved. 0000 RCOSC 0001 XOSC 0010 PLL0 PHI

27.3.1.27 Auxiliary Clock 1 Divider 0 Configuration Register (CGM_AC1_DC0)

This register controls the CAN Subsystem clock.

Note: Byte and half-word write accesses are not allowed for this register. Such attempts do not result in an exception, however the new value is not loaded.

Address: 0x0128 Access: User read/write, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DE	0	0	0	0	0	0	0	0	0	DIV					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

Figure 250. Auxiliary Clock 1 Divider 0 Configuration Register (CGM_AC1_DC0)

Table 349. CGM_AC1_DC0 field descriptions

Field	Description
0 DE	Divider (and resultant clock) Enable 0 Disable 1 Enable
10:15 DIV	Divider Division Value The resultant clock has a period 'DIV + 1' times the selected auxiliary clock source. If the divider is disabled (DE = '0'), writes to the DIV field are ignored.

27.3.1.28 Auxiliary Clock 2 Select Control Register (CGM_AC2_SC)

This register is used to select the current clock source for the PLL0 reference clock.

Address: 0x0140 Access: User read/write, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	SELCTL				0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

Figure 251. Auxiliary Clock 2 Select Control Register (CGM_AC2_SC)

Table 350. CGM_AC2_SC field descriptions

Field	Description
4:7 SELCTL	Auxiliary Clock Source Selection Control. Selects the clock source for this auxiliary clock. Unused values are reserved. 0000 RCOSC 0001 XOSC

27.3.1.29 Auxiliary Clock 2 Select Status Register (CGM_AC2_SS)

This register reports the current source clock for Auxiliary Clock 2.

Address: 0x0144 Access: User read/write, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	SELSTAT				0	0	0	0	0	0		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

Figure 252. Auxiliary Clock 2 Select Status Register (CGM_AC2_SS)

Table 351. CGM_AC2_SS field descriptions

Field	Description
4:7 SELSTAT	Auxiliary Clock Source Selection Status Indicates the current clock source for this auxiliary clock. All unused values are reserved. 0000 RCOSC 0001 XOSC

27.3.1.30 Auxiliary Clock 3 Select Control Register (CGM_AC3_SC)

This register is used to select the current clock source for the PLL1 reference clock.

See *Figure 253* for details.

Address : 0x0160 Access: User read/write, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	SELCTL				0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

Figure 253. Auxiliary Clock 3 Select Control Register (CGM_AC3_SC)

Table 352. CGM_AC3_SC field descriptions

Field	Description
4:7 SELCTL	Auxiliary Clock Source Selection Control Selects the clock source for this auxiliary clock. Unused values are reserved. 0001 XOSC 0011 PLL PHI1 0101 PLL0 PHI1 DIV2

27.3.1.31 Auxiliary Clock 3 Select Status Register (CGM_AC3_SS)

This register reports the current source clock for Auxiliary Clock 3.

Address: 0x0164 Access: User read, Supervisor read, Test read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	SELSTAT				0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

Figure 254. Auxiliary Clock 3 Select Status Register (CGM_AC3_SS)

Table 353. CGM_AC3_SS field descriptions

Field	Description
4:7 SELSTAT	Auxiliary Clock Source Selection Status Indicates the current clock source for this auxiliary clock. All unused values are reserved. 0001 XOSC 0011 PLL PHI1 0101 PLL0 PHI1 DIV2

27.3.1.32 Auxiliary Clock 4 Select Control Register (CGM_AC4_SC)

This register is used to select the current clock source for the Motor Control Subsystem.

See *Figure 255* for details.

Address: 0x0180 Access: User read/write, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	SELCTL				0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

Figure 255. Auxiliary Clock 4 Select Control Register (CGM_AC4_SC)

Table 354. CGM_AC4_SC field descriptions

Field	Description
4:7 SELCTL	Auxiliary Clock Source Selection Control Selects the clock source for this auxiliary clock. Unused values are reserved. 0000 RCOSC 0010 PLL0 PHI

27.3.1.33 Auxiliary Clock 4 Select Status Register (CGM_AC4_SS)

This register reports the current source clock for Auxiliary Clock 4.

Address: 0x0184 Access: User read, Supervisor read, Test read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	SELSTAT				0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

Figure 256. Auxiliary Clock 4 Select Status Register (CGM_AC4_SS)

Table 355. CGM_AC4_SS field descriptions

Field	Description
4:7 SELSTAT	Auxiliary Clock Source Selection Status Indicates the current clock source for this auxiliary clock. All unused values are reserved. 0000 RCOSC 0010 PLL0 PHI

27.3.1.34 Auxiliary Clock 4 Divider 0 Configuration Register (CGM_AC4_DC0)

This register controls the sent clock.

Note: Byte and half-word write accesses are not allowed for this register. Such attempts do not result in an exception, however the new value is not loaded.

Address: 0x0188 Access: User read/write, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DE	0	0	0	0	0	0	0	0	0	0	0	DIV			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

Figure 257. Auxiliary Clock 4 Divider 0 Configuration Register (CGM_AC4_DC0)

Table 356. CGM_AC4_DC0 field descriptions

Field	Description
0 DE	Divider (and resultant clock) Enable 0 Disable. 1 Enable.
12:15 DIV	Divider Division Value The resultant clock has a period 'DIV + 1' times the selected auxiliary clock source. If the divider is disabled (DE = '0'), writes to the DIV field are ignored.

27.3.1.35 Auxiliary Clock 4 Divider 1 Configuration Register (CGM_AC4_DC1)

This register controls the CTU clock.

Note: Byte and half-word write accesses are not allowed for this register. Such attempts do not result in an exception, however the new value is not loaded.

Address: 0x018C Access: User read/write, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DE	0	0	0	0	0	0	0	0	DIV						
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

Figure 258. Auxiliary Clock 4 Divider 1 Configuration Register (CGM_AC4_DC1)

Table 357. CGM_AC4_DC1 field descriptions

Field	Description
0 DE	Divider (and resultant clock) Enable 0 Disable 1 Enable
9:15 DIV	Divider Division Value The resultant clock has a period 'DIV + 1' times the selected auxiliary clock source. If the divider is disabled (DE = '0'), writes to the DIV field are ignored.

27.3.1.36 Auxiliary Clock 4 Divider 2 Configuration Register (CGM_AC4_DC2)

This register controls the SAR clock.

Note: Byte and half-word write accesses are not allowed for this register. Such attempts do not result in an exception, however the new value is not loaded.

Address: 0x0190 Access: User read/write, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DE	0	0	0	0	0	0	0	0	DIV						
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

Figure 259. Auxiliary Clock 4 Divider 2 Configuration Register (CGM_AC4_DC2)

Table 358. CGM_AC4_DC2 field descriptions

Field	Description
0 DE	Divider (and resultant clock) Enable 0 Disable 1 Enable
9:15 DIV	Divider Division Value The resultant clock has a period 'DIV + 1' times the selected auxiliary clock source. If the divider is disabled (DE = '0'), writes to the DIV field are ignored.

27.3.1.37 Auxiliary Clock 4 Divider 3 Configuration Register (CGM_AC4_DC3)

This register controls the PWM clock.

Note: Byte and half-word write accesses are not allowed for this register. Such attempts do not result in an exception, however the new value is not loaded.

Address: 0x0194 Access: User read/write, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DE	0	0	0	0	0	0	0	0	DIV						
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

Figure 260. Auxiliary Clock 4 Divider 3 Configuration Register (CGM_AC4_DC3)

Table 359. CGM_AC4_DC3 field descriptions

Field	Description
0 DE	Divider (and resultant clock) Enable 0 Disable 1 Enable
9:15 DIV	Divider Division Value The resultant clock has a period 'DIV + 1' times the selected auxiliary clock source. If the divider is disabled (DE = '0'), writes to the DIV field are ignored.

27.3.1.38 Auxiliary Clock 4 Divider 4 Configuration Register (CGM_AC4_DC4)

This register controls the ETMR clock.

Note: Byte and half-word write accesses are not allowed for this register. Such attempts do not result in an exception, however the new value is not loaded.

Address: 0x0198 Access: User read/write, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DE	0	0	0	0	0	0	0	0	DIV						
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

Figure 261. Auxiliary Clock 4 Divider 4 Configuration Register (CGM_AC4_DC4)

Table 360. CGM_AC4_DC4 field descriptions

Field	Description
0 DE	Divider (and resultant clock) Enable 0 Disable 1 Enable
9:15 DIV	Divider Division Value The resultant clock has a period 'DIV + 1' times the selected auxiliary clock source. If the divider is disabled (DE = '0'), writes to the DIV field are ignored.

27.3.1.39 Auxiliary Clock 4 Divider 5 Configuration Register (CGM_AC4_DC5)

This register controls the SWG_PWM clock.

Note: Byte and half-word write accesses are not allowed for this register. Such attempts do not result in an exception, however the new value is not loaded.

Address: 0x019C Access: User read/write, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DE	0	0	0	0	0	0	0	0	DIV						
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

Figure 262. Auxiliary Clock 4 Divider 5 Configuration Register (CGM_AC4_DC5)

Table 361. CGM_AC4_DC5 field descriptions

Field	Description
0 DE	Divider (and resultant clock) Enable 0 Disable 1 Enable
9:15 DIV	Divider Division Value The resultant clock has a period 'DIV + 1' times the selected auxiliary clock source. If the divider is disabled (DE = '0'), writes to the DIV field are ignored.

27.3.1.40 Auxiliary Clock 5 Select Control Register (CGM_AC5_SC)

This register is used to select the current clock source for Auxiliary clock selector 5 (Flexray).

See *Figure 263* for details.

Address : 0x01A0 Access: User read/write, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	SELCTL				0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

Figure 263. Auxiliary Clock 5 Select Control Register (CGM_AC5_SC)

Table 362. CGM_AC5_SC field descriptions

Field	Description
4:7 SELCTL	Auxiliary Clock Source Selection Control Selects the clock source for this auxiliary clock. Unused values are reserved. 0000 RCOSC 0010 PLL0 PHI

27.3.1.41 Auxiliary Clock 5 Select Status Register (CGM_AC5_SS)

This register reports the current source clock for Auxiliary Clock 5.

Address: 0x01A4 Access: User read, Supervisor read, Test read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	SELSTAT				0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

Figure 264. Auxiliary Clock 5 Select Status Register (CGM_AC5_SS)

Table 363. CGM_AC5_SS field descriptions

Field	Description
4:7 SELSTAT	Auxiliary Clock Source Selection Status Indicates the current clock source for this auxiliary clock. All unused values are reserved. 0000 RCOSC 0010 PLL0 PHI

27.3.1.42 Auxiliary Clock 5 Divider 0 Configuration Register (CGM_AC5_DC0)

This register controls the FlexRay clock.

Note: Byte and half-word write accesses are not allowed for this register. Such attempts do not result in an exception, however the new value is not loaded.

Address: 0x01A8 Access: User read/write, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DE	0	0	0	0	0	0	0	0	0	0	0	0	0	DIV	
W																
Reset	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

Figure 265. Auxiliary Clock 5 Divider 0 Configuration Register (CGM_AC5_DC0)

Table 364. CGM_AC5_DC0 field descriptions

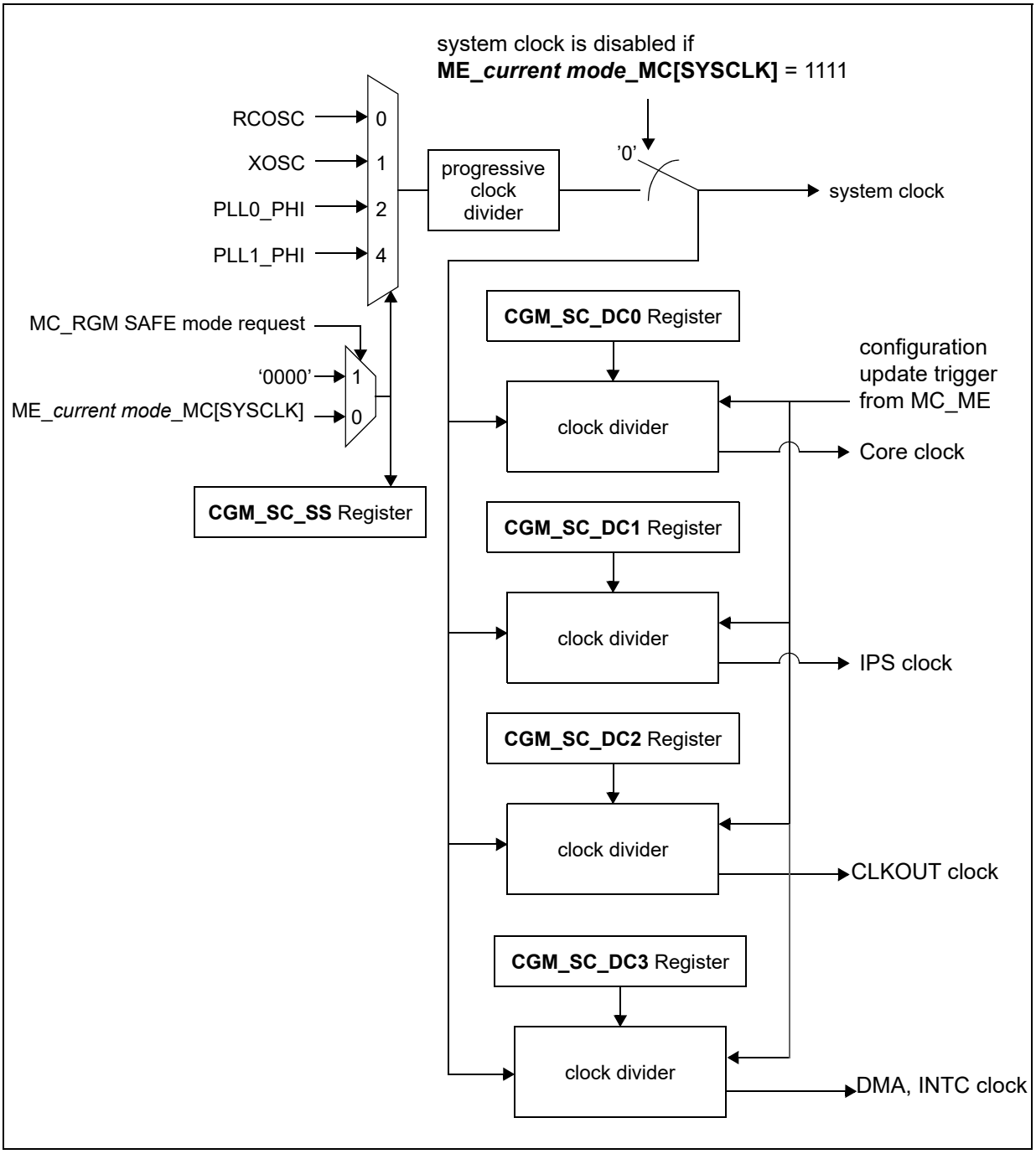
Field	Description
0 DE	Divider (and resultant clock) Enable 0 Disable. 1 Enable.
14:15 DIV	Divider Division Value The resultant clock has a period 'DIV + 1' times the selected auxiliary clock source. If the divider is disabled (DE = '0'), writes to the DIV field are ignored.

27.4 Functional Description

27.4.1 System Clock Generation

Figure 266 shows the block diagram of the system clock generation logic. The MC_ME provides the system clock select and switch mask (see MC_ME chapter for more details), and the MC_RGM provides the safe clock request (see MC_RGM chapter for more details). The safe clock request forces the selector to select the RCOSC as the system clock and ignore the system clock select.

Figure 266. MC_CGM System Clock Generation Overview



27.4.1.1 System Clock Source Selection

During normal operation, the system clock selection is controlled:

- On a SAFE mode or reset event, by the MC_RGM.
- Otherwise, by the MC_ME.

27.4.1.2 Progressive System Clock Switching

In order to prevent sudden voltage drops and overshoots due to frequency and load changes, the MC_ME requests the MC_CGM to ramp the system clock frequency up or down based on the power level values of the current and target modes.

During ramp-down, the rate of the frequency change is based on the CGM_PCS_SDUR, CGM_PCS_DIVCn, and CGM_PCS_DIVEn registers, where n corresponds to the current system clock source selection.

During ramp-up, the rate of the frequency change is based on the CGM_PCS_SDUR, CGM_PCS_DIVCn, and CGM_PCS_DIVSn registers, where n corresponds to the target system clock source selection.

27.4.1.2.1 Generic Clock Change Requirements

For a maximum allowed change of the frequency (f_{chg}) and for a given source clock frequency f_{src} (current or target clock source), the maximum allowed frequency change rate a_{max} is given in *Equation 19*.

Equation 19

$$a_{max} = \frac{f_{chg}}{f_{src}}$$

27.4.1.2.2 Configuration of CGM_PCS_SDUR

The switch duration field CGM_PCS_SDUR[SDUR] defines the duration of one system clock switch step in terms of RCOSC cycles. After the expiration of this time, the module changes the clock divider value which changes the frequency of the system clock.

27.4.1.2.3 Configuration of CGM_PCS_DIVC*n*[RATE]

For a given maximum allowed frequency change rate a_{max} the value *d* to be programmed into the PCS_DIVCn[RATE] register is given in *Table 365*.

Table 365. MC_CGM CGM_PCS_DIVCn[RATE] values

a_{max}	d = CGM_PCS_DIVCn[RATE]
0.05	0.012
0.10	0.048
0.15	0.112
0.20	0.184
0.25	0.334
0.30	0.429
0.35	0.539
0.40	0.667
0.45	0.819
0.50	1.000

27.4.1.2.4 Generic Clock Change Properties

The number *k* of required steps to reach or leave the divided system clock with frequency f_{div} from the source clock with frequency f_{src} is given in *Equation 20*.

Equation 20

$$k = \left\lceil 0.5 + \sqrt{0.25 - \frac{2\left(1 - \frac{f_{src}}{16MHz}\right)}{d}} \right\rceil$$

27.4.1.2.5 Clock Ramp-Down

The clock ramp-down starts with the divider value 1 and the given divider increment value PCD_DIVCn[RATE], and ends with the given divider value PCS_DIVEn[DIVE].

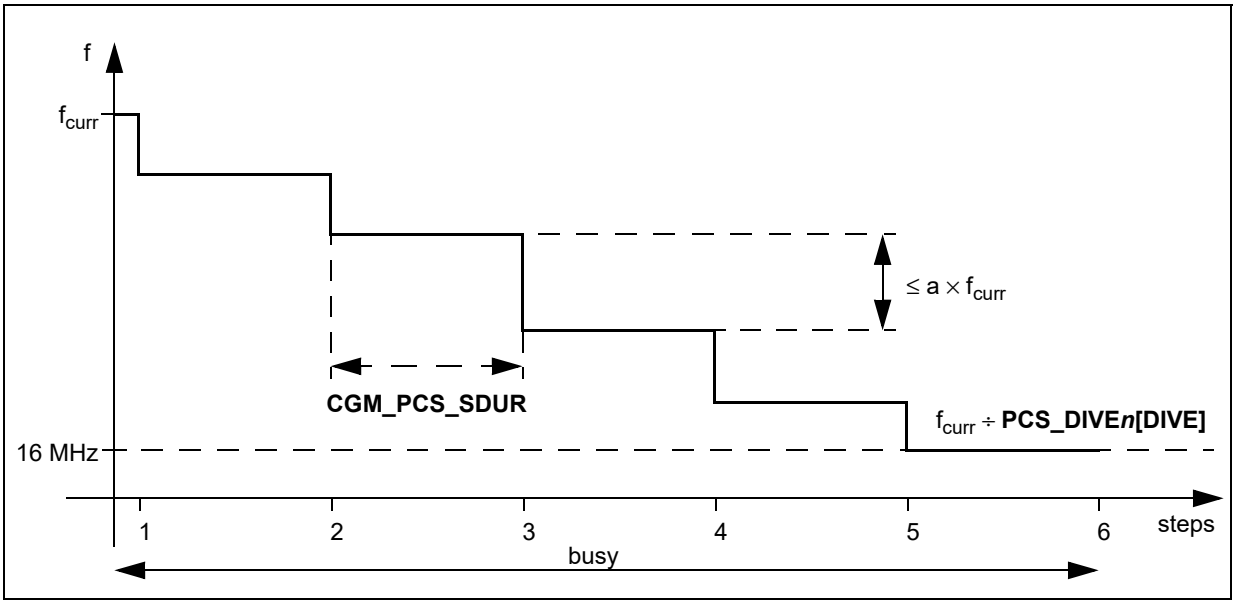
The divider end value for clock ramp-down is given in *Equation 21*.

Equation 21

$$\textbf{PCS_DIVEn[DIVE]} = \frac{f_{src}}{16MHz} \times 1000 - 1$$

Where f_{curr} is the frequency of the currently selected system clock source.

Figure 267. MC_CGM System Clock Ramp-Down Timing (k = 6 example)



27.4.1.2.6 Clock Ramp-Up

The clock ramp-up starts with the given divider value PCS_DIVSn[DIVS] and with the given divider decrement value PCD_DIVCn[INIT] and ends with the divider value 1.

The initial divider change start value PCS_DIVCn[INIT] for system clock ramp-up is given in *Equation 22*.

Equation 22

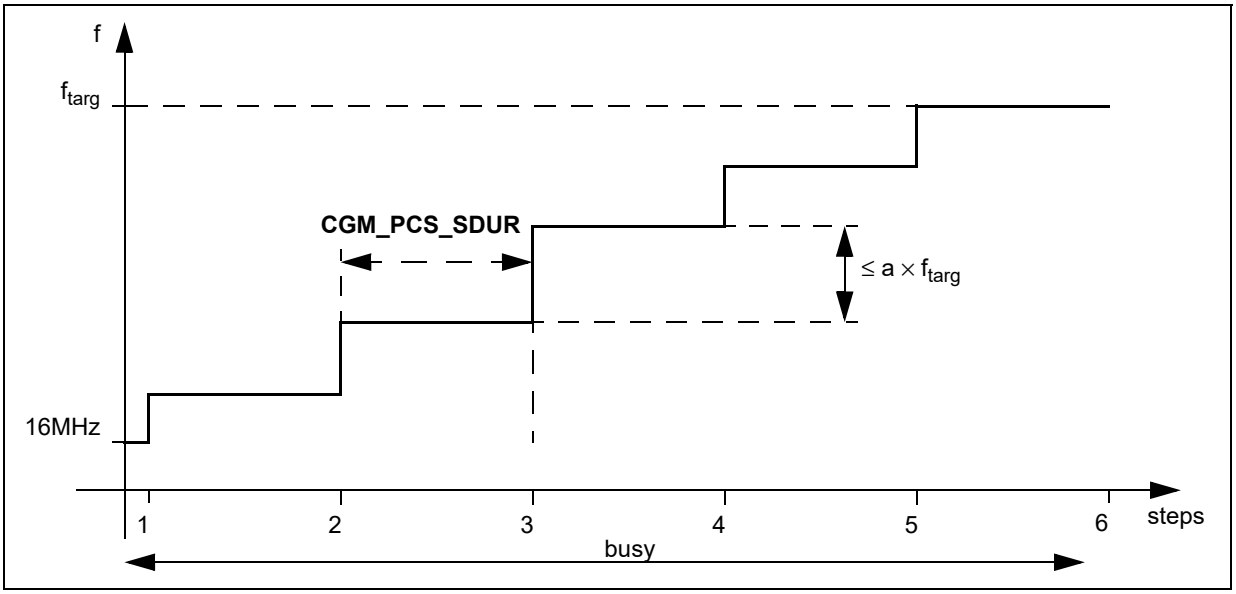
$$\textbf{PCS_DIVCn[INIT]} = d \times k \times 1000 - 1$$

Where *k* is calculated from *Equation 20* using the target system clock source frequency f_{targ}. The divider start value for clock ramp-up is given in *Equation 23*.

Equation 23

$$\textbf{PCS_DIVSn[DIVS]} = \left(1 + d\frac{k(k+1)}{2}\right) \times 1000 - 1$$

Figure 268. MC_CGM System Clock Ramp-Up Timing (k = 6 example)



27.4.1.3 System Clock Disable

During the TEST mode, the system clock can be disabled by the MC_ME.

27.4.1.4 System Clock Dividers

The MC_CGM generates the following derived clocks from the system clock:

- Core clock - controlled by the CGM_SC_DC0 register
- IPS clock - controlled by the CGM_SC_DC1 register
- CLKOUT clock - controlled by the CGM_SC_DC2 register
- DMA, INTC clock - controlled by the CGM_SC_DC3 register

27.4.1.4.1 System Clock Divider Synchronization

The system clock dividers are synchronized such that the rising edges of the lower frequency clocks are aligned with those of the higher frequency clocks. This imposes limitations on the division factors which can be used:

- Each division factor must be an integer multiple of each division factor that has a lower value.

Example 1 Correct System Clock Divider Configuration

Table 366. MC_CGM Example System Clock Division Values Compatible with Divider Synchronization

Divider	Division Factor	Register Value
0	1	**CGM_SC_DC0[DIV]** = 6b000000
1	2	**CGM_SC_DC1[DIV]** = 6b000001
2	6	**CGM_SC_DC2[DIV]** = 6b000101

The above example is correct because:

- System clock divider 1 has a division factor greater than divider 0; 2 is an integer multiple of 1.
- System clock divider 2 has a division factor greater than dividers 0 and 1; 6 is an integer multiple of 1 and 2.

Example 2 Incorrect System Clock Divider Configuration

Table 367. MC_CGM Example System Clock Division Values Incompatible with Divider Synchronization

Divider	Division Factor	Register Value
0	1	**CGM_SC_DC0[DIV]** = 6b000000
1	4	**CGM_SC_DC1[DIV]** = 6b000011
2	6	**CGM_SC_DC2[DIV]** = 6b000101

The above example is incorrect because:

- System clock divider 1 has a division factor greater than divider 0; 4 is an integer multiple of 1.
- System clock divider 2 has a division factor greater than dividers 0 and 1; 6 is not an integer multiple of 4.

Warning: Configuring the system clock dividers incorrectly causes the divided clocks to become unsynchronized. This may lead to lost communication between the clock domains and/or the inability to further change the system clock divider division factors without first disabling them or performing a reset.

Note: The same rule applicable on system clock dividers for system clock divider synchronization is applicable on Dividers present on Auxiliary clock4 dividers. This means The auxiliary clock4 dividers are synchronized such that the rising edges of the lower frequency clocks are aligned with those of the higher frequency clocks. The impacted registers are: CGM_AC4_DC0–5

27.4.1.4.2 System Clock Divider Update Triggering

System clock dividers are updated only with the next software triggered mode change (This is applicable to auxilliary mux 4 as well). This ensures that the system clock dividers are updated simultaneously without inconsistent intermediate configurations, and that the divider configurations are aligned with the system clock selection of the target mode.

The system clock divider configuration update sequence is:

1. Configure the target mode in the MC_ME to select the desired system clock source.
2. Configure the CGM_SC_DC0–3 registers as desired but aligned with the target system clock source frequency and compatible with divider synchronization described in *Section 27.4.1.4.1: System Clock Divider Synchronization*.
3. Request a mode change in the MC_ME to the desired target mode.
4. The new system clock divider configuration is effective on mode change completion and successful switching of system clock.

See the MC_ME chapter for details on how to configure modes and make mode change requests.

27.4.2 Auxiliary Clock Generation

Figure 269 through *Figure 274* show the block diagrams of the generation logic for the various auxiliary clocks. See the following sections for auxiliary clock selection control:

- *Section 27.3.1.20: Auxiliary Clock 0 Select Control Register (CGM_AC0_SC)*
- *Section 27.3.1.25: Auxiliary Clock 1 Select Control Register (CGM_AC1_SC)*
- *Section 27.3.1.28: Auxiliary Clock 2 Select Control Register (CGM_AC2_SC)*
- *Section 27.3.1.30: Auxiliary Clock 3 Select Control Register (CGM_AC3_SC)*
- *Section 27.3.1.32: Auxiliary Clock 4 Select Control Register (CGM_AC4_SC)*
- *Section 27.3.1.40: Auxiliary Clock 5 Select Control Register (CGM_AC5_SC)*

Figure 269. MC_CGM Auxiliary Clock 0 Generation Overview

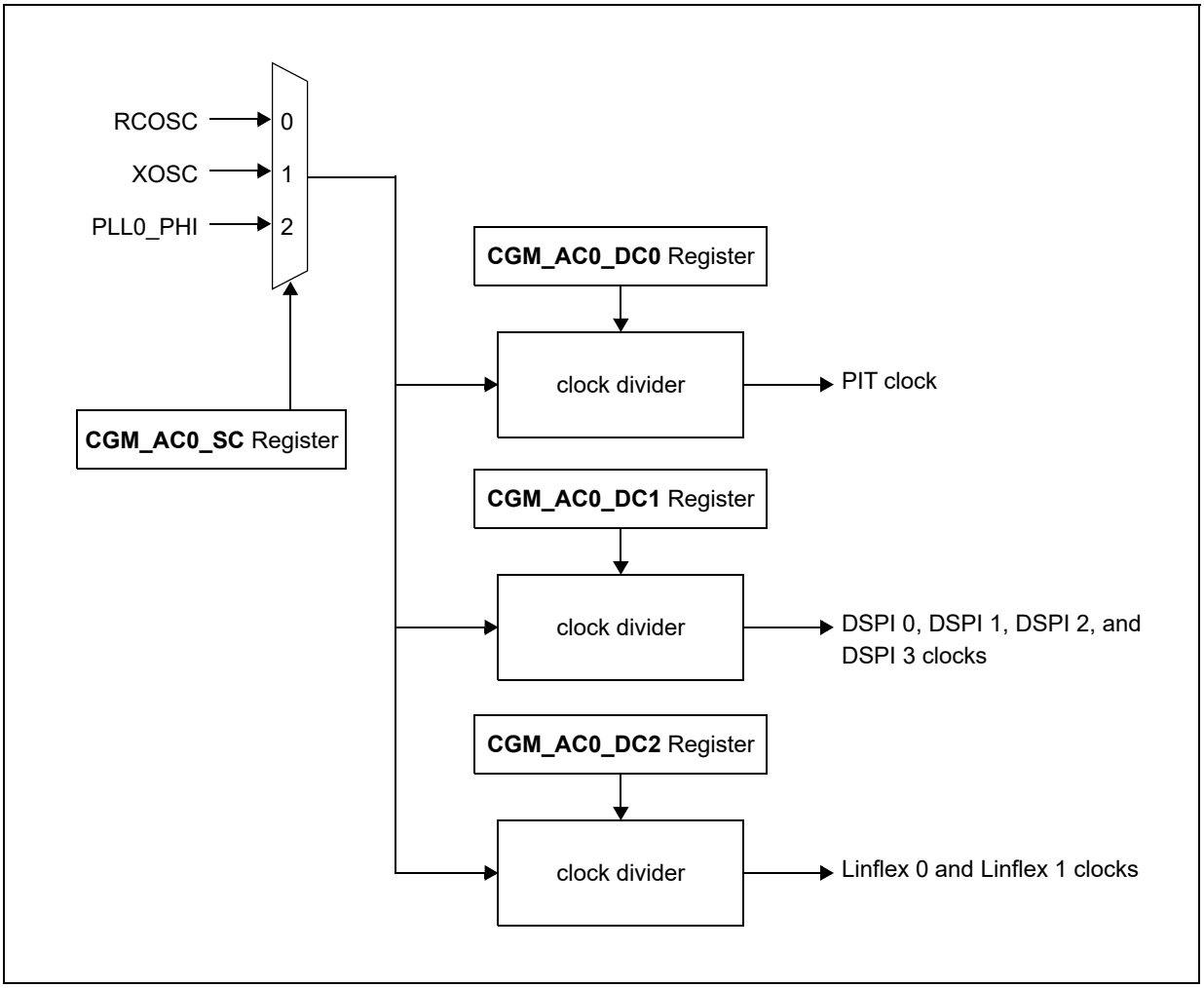


Figure 270. MC_CGM Auxiliary Clock 1 Generation Overview

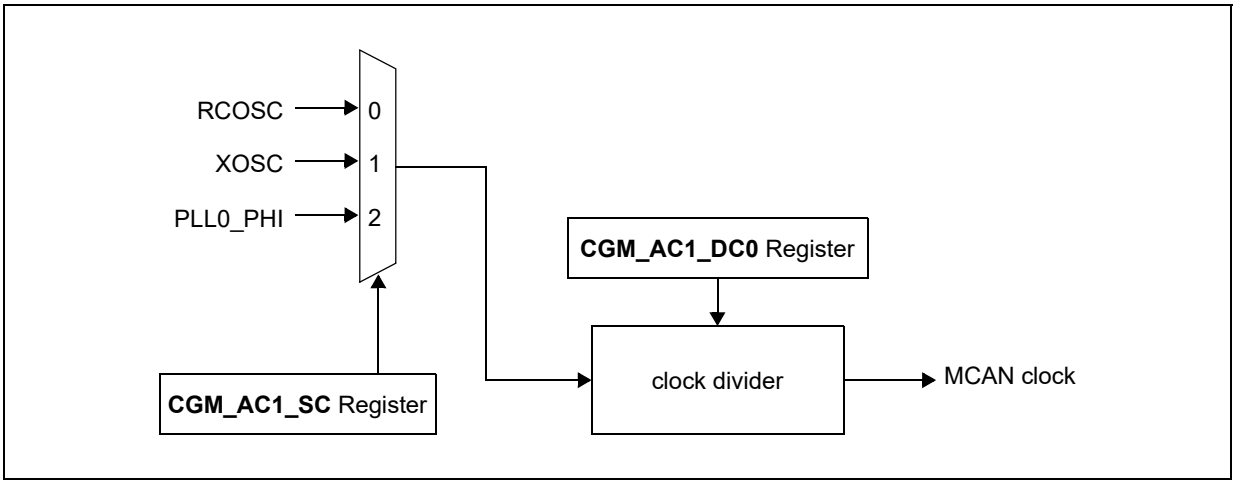


Figure 271. MC_CGM Auxiliary Clock 2 Generation Overview

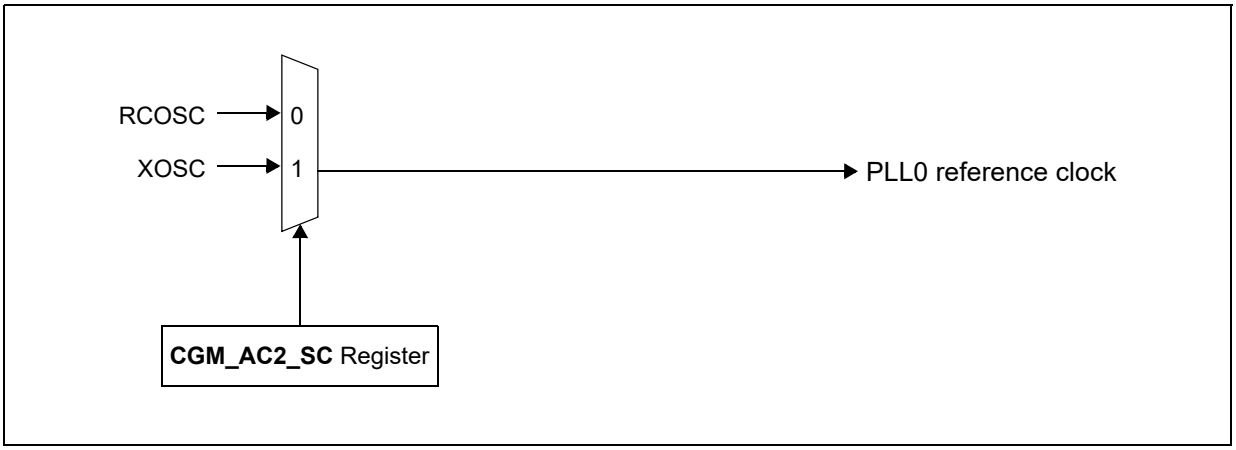


Figure 272. MC_CGM Auxiliary Clock 3 Generation Overview

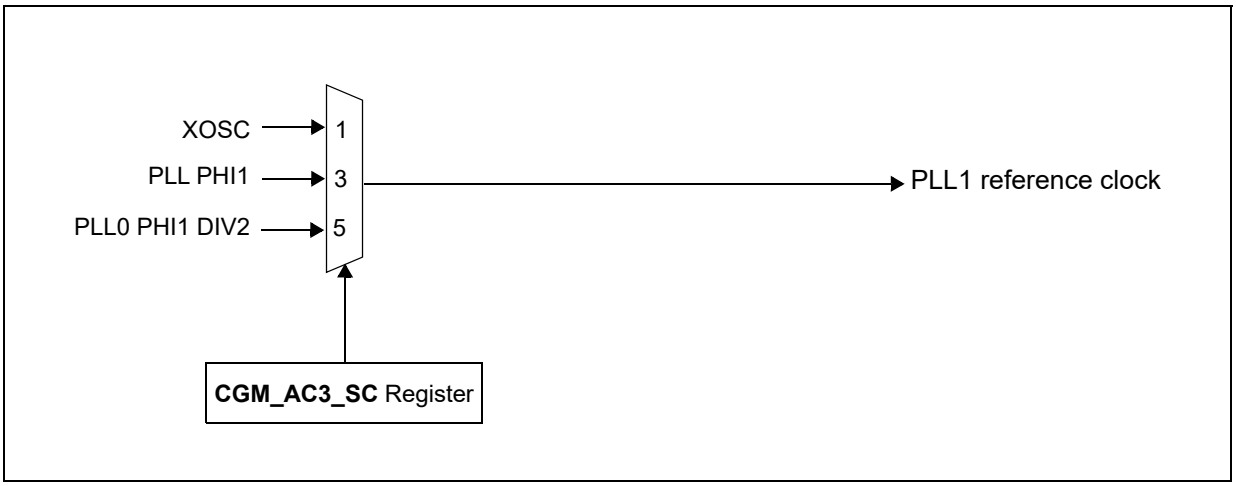


Figure 273. MC_CGM Auxiliary Clock 4 Generation Overview

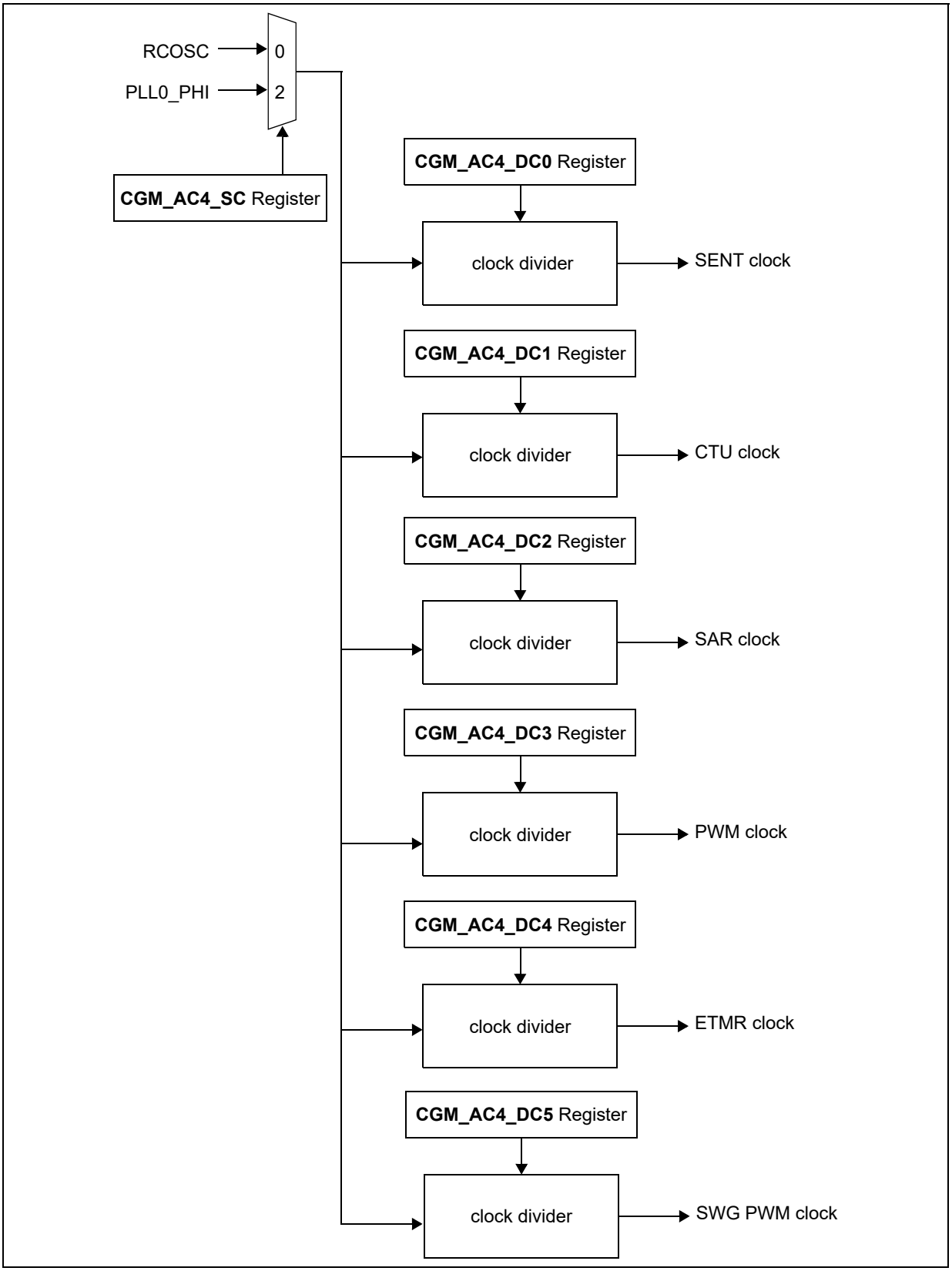
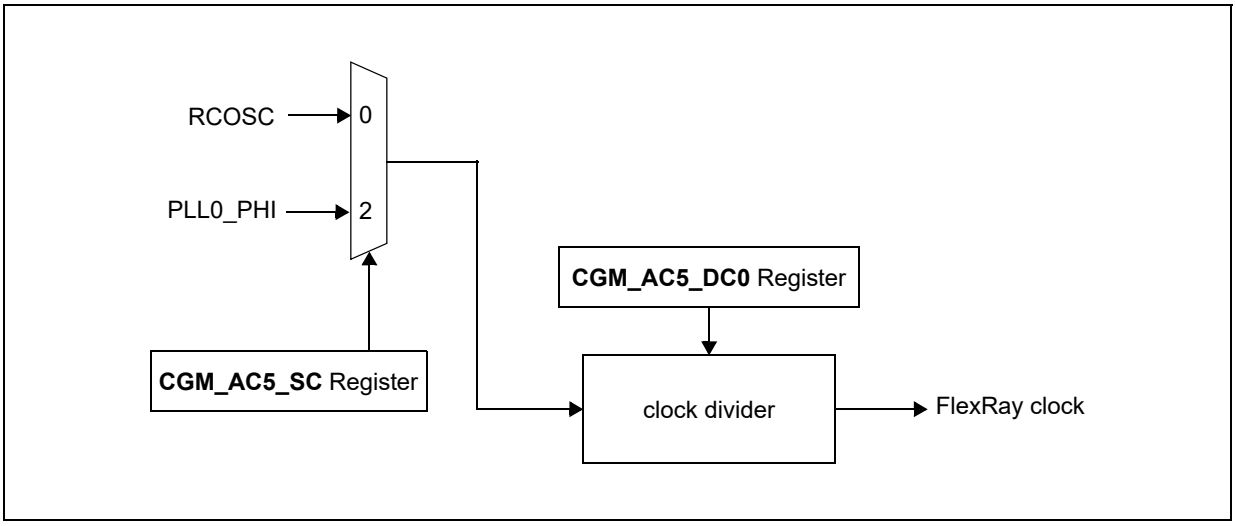


Figure 274. MC_CGM Auxiliary Clock 5 Generation Overview



27.4.2.1 Auxiliary Clock Dividers

The MC_CGM generates clocks the following derived clocks:

- Pit clock – controlled by the CGM_AC0_DC0 register.
- DSPI clock 0 – controlled by the CGM_AC0_DC1 register.
- DSPI clock 1 – controlled by the CGM_AC0_DC1 register.
- DSPI clock 2 – controlled by the CGM_AC0_DC1 register.
- DSPI clock 3 – controlled by the CGM_AC0_DC1 register.
- Linflex clock 0 – controlled by the CGM_AC0_DC2 register.
- Linflex clock 1 – controlled by the CGM_AC0_DC2 register.
- MCAN clock – controlled by the CGM_AC1_DC0 register.
- Sent clock – controlled by the CGM_AC4_DC0 register.
- CTU clock – controlled by the CGM_AC4_DC1 register.
- SAR clock – controlled by the CGM_AC4_DC2 register.
- PWM clock – controlled by the CGM_AC4_DC3 register.
- ETMR clock – controlled by the CGM_AC4_DC4 register.
- SWG PWM clock – controlled by the CGM_AC4_DC5 register.
- FlexRay clock – controlled by the CGM_AC5_DC0 register.

27.4.3 Dividers Functional Description

Dividers are used for the generation of divided system and peripheral clocks. The control registers for the built-in dividers can be referenced in *Section 27.3.1: Register descriptions*.

The reset value of all counters is ‘1’. If the DE bit of a divider in the respective configuration register is set to ‘0’, the divider is disabled and any value in the DIV field is ignored.

28 OSC Digital Interface (XOSC)

28.1 Introduction

The XOSC digital interface is used to control the on-chip oscillator (XOSC) and provides the register interface for the programmable features. Selection of the XOSC as a source clock is controlled the Clock Generation module (MC_CGM). The XOSC is enabled and disabled by configuring the Mode Entry module (MC_ME).

The programmable features of the XOSC digital interface are as follows:

- Oscillator power-down control and status
- Oscillator startup delay
- Oscillator clock available interrupt
- Oscillator bypass mode control

28.2 Functional description

The XOSC digital interface provides control of the on-chip oscillator and the register interface for the mode and startup delay control. The oscillator provides an output clock that can used as an input to PLL(s) and can be selected as the reference clock for many of the peripherals on the device.

Table 368 shows the possible configurations of the XOSC.

Table 368. XOSC configurations

OSC EN	XOSC_CTL[OSCBYP]	XTAL	EXTAL	XOSC Output	XOSC MODE
0	0	No crystal, high-impedance	No crystal, high-impedance	0	Power down, IDDQ
0	1	x	external clock	Undefined	Reserved, OSC Disabled
1	1	x	external clock	XTALOUT	Bypass, OSC Enabled
1	0	crystal	crystal	XTALOUT	Normal, OSC Enabled

28.2.1 Oscillator power-down control and status

Enabling and disabling of the oscillator is done in the Mode Entry (MC_ME) module (see “Mode Entry Module (MC_ME)” chapter). The ME_<*mode*>_MC[XOSCON] controls the power-down of oscillator based on the current device mode (see “MC_ME Memory Map” table for available *modes*). The status of the XOSC is shown in the MC_ME_GS[S_XOSC].

28.2.2 Oscillator startup delay

A bit in the UTEST flash memory determines whether the oscillator is enabled at power-up of the device (if the DCF client miscellaneous has been programmed, otherwise the valid configuration is the value present in *Chapter 9: Device Configuration Format (DCF) Records*, *Table 138: DCF client list*). If enabled, the oscillator is started after Phase3 of the

device reset sequence. In order to allow the oscillator to reach full amplitude before use, the internal counter (OSCCNT) is started when the device exits reset. The counter is driven by the oscillator output clock. When the counter reaches XOSC_CTL[EOCV] × 512, a signal is sent to the MC_ME module, allowing a mode switch to the oscillator (when used as a system clock or an input to PLL(s)).

The default value for this field after reset depends on implementation and can be found in the XOSC section of the Device Configuration chapter. After reset, software can clear or change the EOCV field in order to shorten or lengthen the delay until the oscillator is ready.

Note: It is advised that if the EOCV value is modified it should be done as early as possible in user code. This will ensure that the new value will be used before OSCCNT reaches the count value.

28.2.3 Oscillator clock available interrupt

An interrupt can be generated when the count value is reached (OSCCNT = XOSC_CTL[EOCV] × 512). The XOSC_CTL[I_OSC] will be set and an interrupt will be generated if the interrupt mask is set (XOSC_CTL[M_OSC] = 1).

28.2.4 Oscillator bypass mode

The oscillator circuit can be bypassed by setting XOSC_CTL[OSCBYP]. This field can only be set by software, and is only cleared by a system reset. In bypass mode, the oscillator is disabled, and the input clock on the EXTAL pins is level-shifted and driven to the logic on the device. The bypass configuration is independent of the power-down mode of the oscillator.

Note: The external oscillator must be running before the XOSC is turned off. If XOSC is being turned on, the status will be updated only after the clock starts toggling.

28.2.4.1 Full Swing Pierce mode

Full Swing Pierce mode (FSP) has the following attributes when compared with LCP:

- Provides a wider frequency range
- Less susceptible to noise
- The crystal amplitude is rail-to-rail
  - Distorted oscilation with several harmonics
  - Increased EMI
- Slower settling time
- Feedback resistor and 300 ohms of current limitation are already integrated.

Note: Consult with crystal supplier to determine the need of extra resistance in XTAL to avoid crystal overdrive.

28.3 Memory map and register definition

Table 369 shows the memory map of the XOSC registers.

Table 369. XOSC memory map

Offset	Register	Location
0x00	XOSC Control Register (XOSC_CTL)	*Section 28.3.1.1*

28.3.1 Register descriptions

Note: Module generates transfer error (ips_xfr_err) only for 32 bytes address space (00 : 1F). If the address space allocated for this module exceed 32 bytes, then transfer error has to be managed at SoC level for the remaining space.

28.3.1.1 XOSC control register (XOSC_CTL)

Offset: 0x00 Access: Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	OSCBYP(1)	0	0	0	0	0	0	0	EOCV(1)							
W																
Reset	–	0	0	0	0	0	0	0	–	–	–	–	–	–	–	–

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	M_OSC(1)	0	0	0	0	0	0	0	I_OSC	0	0	0	0	0	0	0
W																
Reset	–	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. See the Clocking chapter for reset value.

Figure 275. XOSC_CTL register

Table 370. XOSC_CTL register field descriptions

Field	Description
0 OSCBYP	Crystal Oscillator bypass This bit specifies whether the oscillator should be bypassed or not. Software can only set this bit. System reset is needed to reset this bit. 0 Oscillator output is used as root clock 1 EXTAL is used as root clock
8:15 EOCV	End of Count Value These bits specify the end of count value to be used for comparison by the oscillator stabilization counter OSCCNT after reset or whenever it is switched on from the off state. This counting period ensures that external oscillator clock signal is stable before it can be selected by the system. When oscillator counter reaches the value EOCV×512, oscillator available interrupt request is generated. The OSCCNT counter is kept under reset if oscillator bypass mode is selected.
16 M_OSC	Crystal oscillator clock interrupt mask This bit masks the I_OSC interrupt bit. 0 Crystal oscillator clock interrupt is masked 1 Crystal oscillator clock interrupt is enabled
24 I_OSC	Crystal oscillator clock interrupt This bit can only be set by hardware when OSCCNT counter reaches the count value EOCV × 512. It is cleared by software by writing 1. 0 No oscillator clock interrupt occurred 1 Oscillator clock interrupt pending

29 IRCOSC digital interface

29.1 Introduction

The Internal RC Oscillator digital interface (IRCOSC) controls the internal 16 MHz RC oscillator system. This oscillator system includes the main internal RC oscillator (MRC) , as well as an internal temperature sensor and an internal voltage regulator used to compensate external variations in temperature and voltage

It also provides access to trimming information used for safety implementations.

29.2 Functional description

The IRCOSC provides a high frequency (f_{MRC}) clock with a nominal frequency of 16 MHz. After deassertion of reset, the output clock is available after a short time (see Data Sheet for timing information) which is required for stabilization. The IRCOSC status indication is found in the MC_ME_GS[S_IRC] field (see the “Mode Entry Module MC_ME” chapter).

During a power on reset sequence the following are initialized from TEST flash memory:

- Native trimming bits
- Fine trimming bits
- TSENS code bits

For safety reasons, these bits can be accessed through the IRCOSC Native Trimming register (NT). The main IRCOSC output frequency can be trimmed by configuring IRCOSC_CTL[USER_TRIM[4:0]]. The USER_TRIM bits can be programmed to modify internal capacitor/resistor values to match output clock frequency with δf_{var_SW}, as defined in the Data Sheet.

29.3 Memory map and register definition

Table 371 shows the memory map of the IRCOSC registers.

Table 371. IRCOSC memory map

Offset (hex)	Register	Width (in bits)	Access	Reset Value (hex)	Location
0x00	IRCOSC Control Register (IRCOSC_CTL)	32	R/W	0x0000_0000	*Section 29.3.1.1*
0x04	IRCOSC Native Trimming Register (IRCOSC_NT)	32	R	0xXX00_0XXX(1)	*Section 29.3.1.2*
0x08	IRCOSC Temperature Trimming Register (IRCOSC_TT)	32	R	0xXXXX_XXXX(1)	*Section 29.3.1.3*

1. See register section for specific reset values.

29.3.1 Register descriptions

29.3.1.1 IRCOSC control register (IRCOSC_CTL)

Offset: 0x0000 Access: Read/Write(1)

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	USER_TRIM				
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

1. IRCOSC_CTL is writable only in supervisor mode.

Figure 276. IRCOSC control register (IRCOSC_CTL)

Table 372. IRCOSC_CTL field descriptions

Field	Description
11:15 USER_TRIM	User trimming bits with respect to nominal factory frequency The MSB of the USER_TRIM bits is used to determine whether the frequency will be increased or decreased. If MSB = 0 then a change in USER_TRIM[3:0] will decrease the frequency, and likewise, if MSB = 1 then a change in USER_TRIM[3:0] will increase the frequency. *Table 373* shows how the IRCOSC_CTL[USER_TRIM] field is used to trim the IRCOSC frequency.

Table 373. IRCOSC_CTL[USER_TRIM] frequency trimming calculation(1)

USER_TRIM[4:0] value	Frequency
....	....
10011	$f + (3 \times \delta f_{var_SW})$
10010	$f + (2 \times \delta f_{var_SW})$
10001	$f + \delta f_{var_SW}$
10000	f
00000	f
00001	$f - \delta f_{var_SW}$
00010	$f - (2 \times \delta f_{var_SW})$
00011	$f - (3 \times \delta f_{var_SW})$
....	....

1. See the Datasheet for df_{var_SW} value and min/max frequency variations.

29.3.1.2 IRCOSC Native Trimming register (IRCOSC_NT)

The NT register contains trim values loaded from TEST flash memory for the IRCOSC. Its status is enabled for voltage regulator and temperature sensor. This register is read only.

Offset: 0x0004 Access: User Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	FT_CODE						0	0	TX_CODE					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	REG_EN	TSENS_EN	0	0	RCTRIM							
W																
Reset	0	0	0	0	—(1)	—(1)	0	0	—(1)	—(1)	—(1)	—(1)	—(1)	—(1)	—(1)	—(1)

1. Reset value is coming from factory calibration (TEST Flash)

Figure 277. IRCOSC Native Trimming register (IRCOSC_NT)

Table 374. IRCOSC_NT field descriptions

Field	Description
2:7 FT_CODE	Output from INTERFACE inside RC to check the fine trimming code applied to RC_core
10:15 TX_CODE	Output from TSENS inside RC to check the TSENS code
20 REG_EN	Enables voltage regulator: trimming value for safety check 0 Voltage regulator is not enabled 1 Voltage regulator is enabled
21 TSENS_EN	Enables temperature sensor: trimming value for safety check 0 Temperature sensor is not enabled 1 Temperature sensor is enabled
24:31 RCTRIM	IRCOSC trimming value for safety check

29.3.1.3 IRCOSC Temperature Trimming register (IRCOSC_TT)

The TT register contains system trimming values for the IRCOSC that are loaded from TEST flash memory during reset. TT is a read only register.

Offset: 0x0008 Access: User Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	COLD_TRIM										0	0	HOT_TRIM [0:1]	
W																
Reset	0	0	_(1)	_(1)	_(1)	_(1)	_(1)	_(1)	_(1)	_(1)	_(1)	_(1)	0	0	_(1)	_(1)

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	HOT_TRIM[2:9]										TSENS_TRIM					
W																
Reset	_(1)	_(1)	_(1)	_(1)	_(1)	_(1)	_(1)	_(1)	0	0	_(1)	_(1)	_(1)	_(1)	_(1)	_(1)

1. Reset value is coming from factory calibration (TEST Flash)

Figure 278. IRCOSC Temperature Trimming register (IRCOSC_TT)

Table 375. IRCOSC_TT field descriptions

Field	Description
2:11 COLD_TRIM	Cold temperature trimming value for safety check
14:23 HOT_TRIM	Hot temperature trimming value for safety check
26:31 TSENS_TRIM	Ambient temperature reference code value for safety check

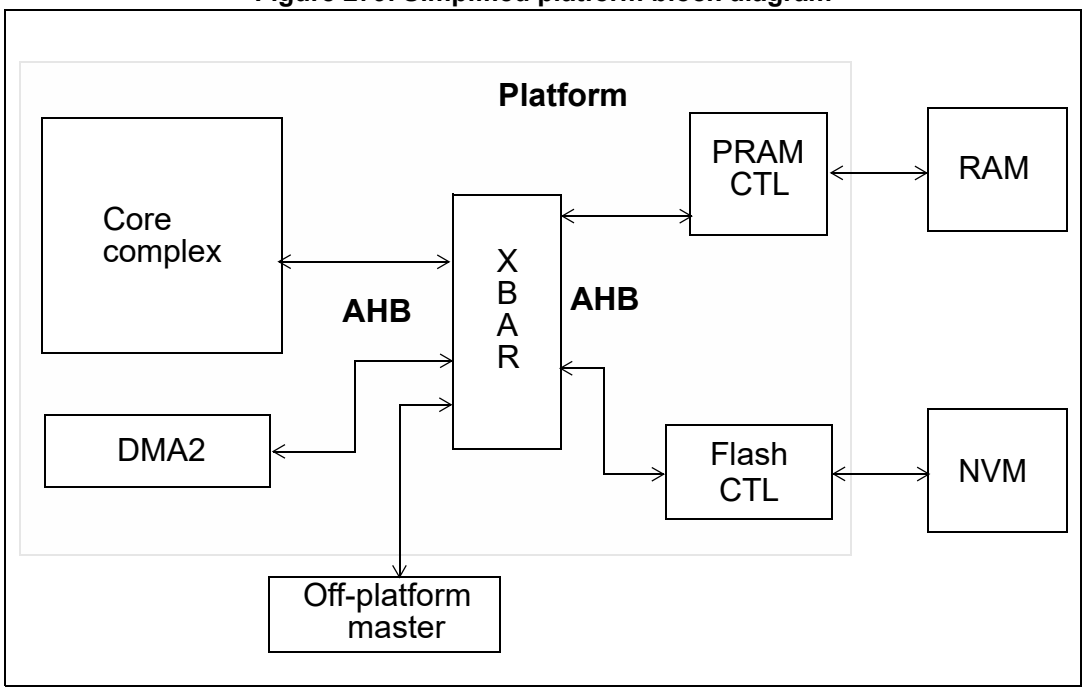
30 Platform RAM controller (PRAM)

30.1 Introduction

This section provides an overview of the Platform RAM Controller. The PRAM controller acts as an interface between the system bus (AHB-Lite 2.v6) and the integrated system RAM. It converts the protocols between the system bus and the dedicated RAM array interface.

The PRAM controller supports two 64-bit AHB and one 64-bit RAM array interfaces. The first AHB port provides a connection to the platform crossbar for direct RAM access from the various crossbar masters. The second AHB port provides a connection from the flash controller to facilitate calibration overlay access. *Figure 279* is a simplified block diagram depicting the position of the PRAM controller within a typical platform architecture:

Figure 279. Simplified platform block diagram



The following list summarizes the key features of the PRAM controller:

- System bus supports 64-bit data and 8-bit ECC AHB interfaces
- Array interface supports a 64-bit data and 8-bit ECC interface
- Late-write support via 64-bit data and 8-bit ECC storage buffer to support single-cycle write accesses
- Configurable read access timing (zero or one wait-state programmable) allowing use in large range of frequency targets
- Read-modify-write operation to support array write size less than 64-bit data

The address path of the PRAM controller contains a multiplexer that chooses among the addresses presented on the system bus during read/write requests to RAM. These addresses are transferred to RAM array either immediately, for read accesses, or, for write accesses, through a temporary write address buffer or through a late-write address buffer

The temporary write address buffer contains the address which was presented during the AHB address phase of the write request. The request is stalled from being presented to the RAM by one cycle in order to simultaneously present the address and associated write data, because the write data is not valid until the AHB data phase. The late-write buffer holds write requests which are delayed to facilitate single-cycle responses on the system bus.

The read data is the result of a RAM access. Along with the read and write data, the corresponding ECC codewords traverse the entire data path of the PRAM controller, including the late-write buffer, to support end-to-end ECC (e2eECC) coverage.

The write data path contains the multiplexer that chooses the source of the write data as well as the control logic for generating read-modify-write operations on writes that are less than 64 bits in size. A write operation can be performed to empty the contents of the late-write buffer. In the case of back-to-back writes, the write may be forced to bypass the late-write buffer and stored directly in the RAM.

30.2 SRAM controller memory map and register definitions

The PRAM controller module provides an IPS programming model mapped to a standard 16 KB on-platform peripheral slot. The programming model can only be referenced using a 32-bit (word) access. Attempted references using different access sizes or to undefined (reserved) addresses will generate an IPS error.

Table 376. PRAM controller memory map

Address offset (hex)	Register name	Reset value (hex)	Location
0x00	Platform RAM Configuration Register 1 (PRCR1)	0x0000_0200	*Section 30.2.1*

30.2.1 Platform RAM configuration register 1 (PRCR1)

The Platform RAM configuration register 1 (PRCR1) is used to specify the operation of the PRAM controller.

Note: This register is accessible only in supervisor mode. Accesses in user mode return a transfer error.

Offset: 0x00 Access: Read/Write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0					0	0	0	0	0	
W							PRI1	PRI0	P1_BO_DIS	P0_BO_DIS						FT_DIS
Reset	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0

Figure 280. Platform RAM configuration register 1 (PRCR1)

Table 377. PRCR1 field descriptions

Field	Description
22 PRI1	Not used, since port 1 is not used. This bit can be written by software.
23 PRI0	Not used, since port 1 is not used. This bit can be written by software.
24 P1_BO_DIS	Not used, since port 1 is not used. This bit can be written by software.
25 P0_BO_DIS	Port p0 read burst optimization disable 0 64-bit WRP4 read bursts are optimized such that the controller returns a 2-1-1-1 response(1) when PRCR[FT_DIS]=1 1 64-bit WRP4 read bursts are not optimized; the controller returns a 2-2-2-2 response(1) when PRCR[FT_DIS]=1
31 FT_DIS	Flow through disabled This field defines the AHB read response of the PRAM controller . The state of this field has no impact on the write response latency. This bit is cleared by hardware reset. **Note:** Do not change the FT_DIS bit value while accessing system RAM. Realocate code programming the FT_DIS bit to another area (for example, local core memory). 0 RAM read data is passed directly to the system bus, incurring no additional latency 1 RAM read data is registered prior to returning on the system bus, incurring 1 extra cycle of latency

30.3 Functional description

30.3.1 Read/Write introduction

The PRAM controller generates chip-select and write-enable, the array address, and write data as inputs to the RAM array. The PRAM controller captures read data from the RAM array and drives it onto the AHB along with the associated dataphase termination signals. To facilitate operating at higher frequencies, the PRAM controller can optionally incur a single wait state on the AHB response.

30.3.1.1 Reads

Read transfers, of any size, can be configured to complete with a zero or one additional wait state response on the AHB.

30.3.1.2 Optional read wait-state

The PRAM controller can be optionally programmed to register RAM read data prior to returning it on the system bus. Setting the PRCR1[FT_DIS] field inserts a register in the read data path for use when operating the controller at high frequencies. The state of PRCR1[FT_DIS] field has no effect on writes.

A random, initial access will take 2 clock cycles (2T) to complete if PRCR1[FT_DIS] = 0, and a WRAP4 burst will have an access time of 2-2-2-2. A random, initial access will take 3 clock

cycles (3T) to complete if PRCR1[FT_DIS] = 1, and a WRAP4 burst will have an access time of 3-2-2-2

30.3.2 Writes

30.3.2.1 64-bit writes

Aligned **64**-bit writes execute in a single AHB data phase cycle, allowing for zero wait states on back-to-back writes of these sizes. If, during the data phase of a write, a read is requested on the AHB, the write is registered in the late-write buffer, allowing the read to take place without a wait state. The valid buffered or late-write data is stored on the next available array address phase.

Back-to-back 64-bit writes execute slightly differently whereby the first write bypasses the late-write buffer. Instead, the write data is stored directly to the array in the same cycle in which it is valid on the AHB.

30.3.2.2 Less than 64-bit writes

Writes of size less than 64 bits incur a read-modify-write action as a consequence of the ECC coding scheme's 64-bit granularity. In the case of a read-modify-write action, the PRAM controller performs single-error correction, double-error detection on the read data. The write data is merged into the appropriate byte lanes along with the potentially-corrected read data. A new codeword is generated based on the newly formed doubleword. The newly formed doubleword and its associated checkbits are subsequently written to the RAM. For details on the ECC coding scheme, refer to *Section 30.5: Safety considerations*.

On a read-modify-write operation, the array read data is registered after it is decoded and before it is merged with the AHB write data. Therefore, writes of less than 64 bits require the insertion of one wait state before the data phase can be completed.

30.3.2.3 Unaligned writes

The PRAM controller is compliant with the AMBA-AHB2.v6 extensions specification concerning byte strobes. The size of the transfer is sufficient to cover all bytes being written and covers more bytes in the case of an unaligned transfer. The address of the transfer is rounded down to the nearest boundary of the size of the transaction.

Note: Unaligned writes always generate read-modify-write operations in the PRAM controller in order to preserve the validity of the ECC codeword.

30.4 Initialization/application information

Prior to reading, the ECC must be initialized by writing a known 64-bit value to each memory address. This includes reads generated from the read-modify-write operation which occurs when a write transfer of less than 64 bits or an unaligned write is requested. Reading from an uninitialized address is likely to generate an uncorrectable ECC event.

The PRAM memory space can be initialized by storing a 64-bit word instruction such as Store Multiple Word (e_stmw) in Power Architecture VLE.

30.5 Safety considerations

30.5.1 Hsiao ECC algorithm

The e2eECC scheme implements a single-error correction, double-error detection (SECDED) code using the so-called Hsiao odd-weight column criteria. These codes are named after M.Y. Hsiao, an IBM researcher who published a paper in the early 1970s about SECDED codes better suited for implementation in protecting (mainframe) computer memories than traditional Hamming codes.

The Hsiao codes are Hamming distance 4 implementations which provide the SECDED capabilities. The minimum odd-weight constraints defined by Hsiao are relatively simple in the resulting implementation of the parity check H matrix which defines the association between the data (and address) bits and the checkbits. They are:

1. There are no all-zeroes columns
2. Every column is distinct
3. Every column contains an odd number of ones, and hence is “odd weight”

In defining the H-matrix for this family of devices, these requirements from Hsiao were applied. Additionally, there are a variety of ECC codeword requirements associated with specific functional requirements associated with the system RAM that further dictated the specific column definitions. In any case, the resulting ECC is organized based on 64 data bits plus 29 address bits (the upper bits of the 32-bit address field minus the 3 bits which select the byte within the 64-bit (8-byte) data field.

The basic H-matrix for this (101, 93) code (93 is the total number of data bits, 101 is the total number of data bits (93) plus 8 checkbits) is shown in *Table 378*. A ‘*’ in *Table 378* indicates the corresponding data or address bit is XOR’ed to form the final checkbit value on the left. For 64-bit data writes, the table sections corresponding to D[63:32], D[31:0], and A[31:3] are logically summed (output of each table section is XOR’ed) together to the final value driven on the hwchkbit[7:0] outputs. Note that this table uses *the AHB bit numbering convention where bit[0] is the least significant bit*.

Table 378. PRAM controller basic H-matrix definition

Checkbits [7:0]	Data Bit(1)																															
	Byte 7								Byte 6								Byte 5								Byte 4							
	63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
7	*				*					*			*		*	*		*	*	*	*					*	*			*	*	
6			*	*			*	*					*		*		*				*				*			*	*			*
5	*	*	*	*					*				*	*		*		*	*				*	*	*				*			
4	*		*			*		*					*		*		*			*			*	*			*				*	
3									*	*	*	*	*		*				*		*				*	*	*	*				
2					*		*					*		*			*	*				*	*							*		*
1		*		*		*		*	*	*	*	*						*		*		*		*						*		*
0		*			*	*	*				*			*	*	*		*				*				*		*	*		*	

Table 378. PRAM controller basic H-matrix definition (continued)

Checkbits [7:0]	Byte 3								Byte 2								Byte 1								Byte 0							
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
7					*		*				*		*				*	*			*	*			*				*	*		*
6	*	*		*	*				*				*				*	*		*	*					*	*				*	*
5			*	*			*	*	*			*	*	*					*		*	*		*							
4	*		*	*				*		*	*				*	*	*			*			*	*				*		*		
3	*	*	*			*						*	*		*	*			*				*		*	*	*			*		
2		*		*	*	*	*	*	*	*	*			*					*				*		*		*	*		*		
1		*						*		*			*	*	*			*				*				*			*	*	*	
0		*		*		*		*				*				*				*				*				*	*		*	*
Checkbits [7:0]	**Address Bit(1)**																															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
7		*		*		*		*	*	*		*	*	*		*	*	*		*		*		*	*	*		*				
6		*		*	*	*	*			*	*			*	*	*	*		*		*	*		*	*	*		*				
5		*	*	*			*	*			*	*		*		*		*	*			*	*	*		*	*	*	*			
4	*	*	*		*	*		*		*	*	*	*		*					*	*		*	*	*		*	*				
3	*		*		*	*	*		*		*	*	*			*		*		*	*		*		*	*	*		*			
2	*	*			*		*	*	*	*				*	*	*	*	*	*	*		*	*		*		*		*			
1	*		*	*		*	*		*	*		*	*		*		*	*	*		*	*	*					*	*			
0	*		*	*	*			*	*		*		*	*	*		*		*	*	*			*		*	*		*			

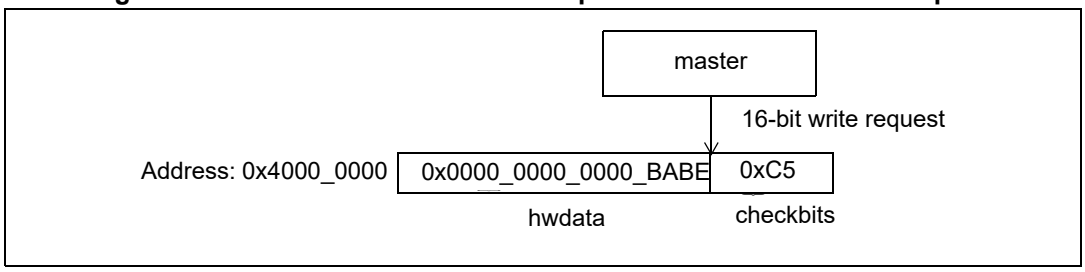
1. Bit numbering is AHB convention: bit 0 is LSB. D[7:0] corresponds to byte at address 0. D[63:56] corresponds to byte at address 7.

30.5.1.1 E2EECC considerations on less-than-64-bit write transactions

Recall that writes of less than 64-bits incur a read-modify-write action as a consequence of the ECC coding scheme 64-bits granularity. In the case of a read-modify-write action, the PRAM controller performs SEC/DED on the read data. The PRAM controller uses the following checkbit manipulation technique to ensure that any faults in the PRAM control logic, associated with performing the read-modify-write action, will ultimately be detectable by e2eECC:

- On an 8-bit, 16-bit or 32-bit write request, the write data is presented by the master on the AHB bus in the correct byte lanes, with the non pertinent byte lanes zeroed out, and the associated checkbit is based on this “zero-padded” write data.
  Consider the following example whereby the master presents a 16-bit write request at address 0x40000000:

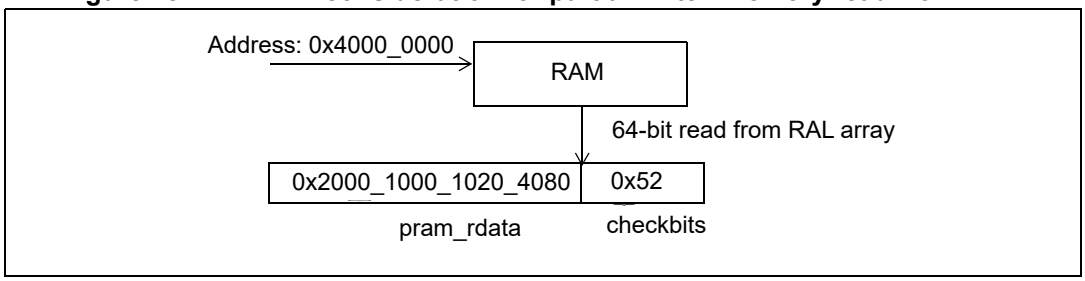
Figure 281. E2EECC consideration for partial write - 16-bit write request



- The PRAM controller detects that the request is a less-than-64-bit write request and initiates a read to address 0x40000000. An ECC check and potential single-bit correction is performed on the data returned from the RAM. In addition, the ECC event is reported to the Memory Error Management Unit (MEMU).

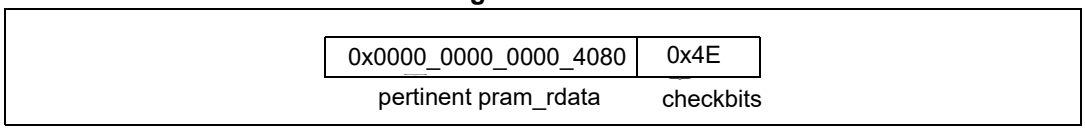
  If a non correctable ECC event is detected, the AHB request is terminated with error on the system bus. In addition, the ECC event is reported to the MEMU.

Figure 282. E2EECC consideration for partial write - memory read from RAM



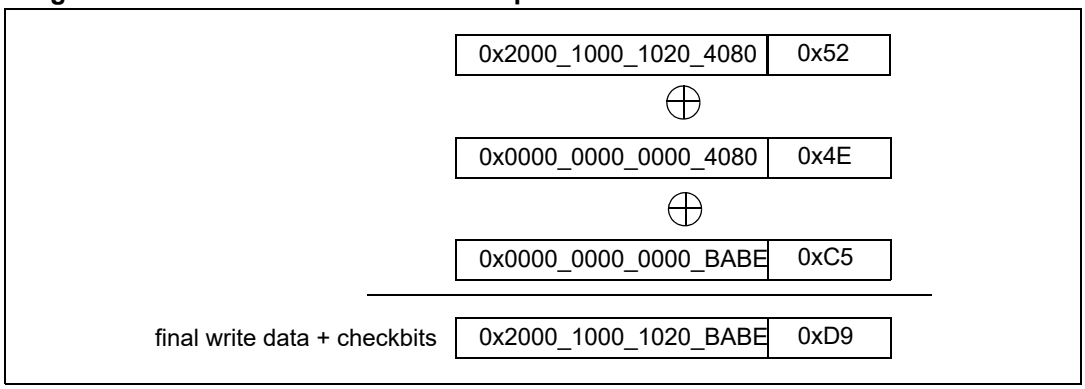
- The PRAM controller isolates the pertinent byte lanes of the read data by zero-padding the non pertinent byte lanes and calculating the checkbit contribution associated with the read data in the byte lanes to be updated as a result of the write request.

Figure 283. E2EECC consideration for partial write - Separation of check-bits of targeted bit-lines



- The PRAM controller merges the non-pertinent read data bytes with the write data in the appropriate byte lanes. Take the checkbit value that was stored in the RAM and remove the checkbit contribution associated with the read data in the byte lanes to be updated. Then introduce the checkbits associated with the write data. The result is ultimately a checkbit value associated with the complete 64-bit data to be written to the array.

Figure 284. E2EECC consideration for partial write - calculation of final check-bits



This technique calculates the ultimate checkbit value on a read-modify-write transaction through a series of data manipulations, taking care to avoid discarding the original checkbits provided by the requesting master, such that faults in the crossbar (XBAR) and PRAM controller datapath and control logic are covered by e2eECC.

Malfunction of the ECC logic described above may result in the corruption of the SEC/DED event reporting. The PRAM controller performs EDC after ECC check on all read-modify-write transactions. If a mismatch is detected, indicating a failure in the ECC logic, the event is reported to the FCCU module.

30.5.2 Transaction monitor

The integrity of the address and data on a system RAM transaction is covered by an e2eECC check performed by the requesting master. Functional safety coverage of the address path and control within the RAM controller is handled by a transaction monitor which verifies the integrity of the transactions between the PRAM controller and the RAM array. The transaction monitor verifies RAM transactions initiated to service the following types of transactions

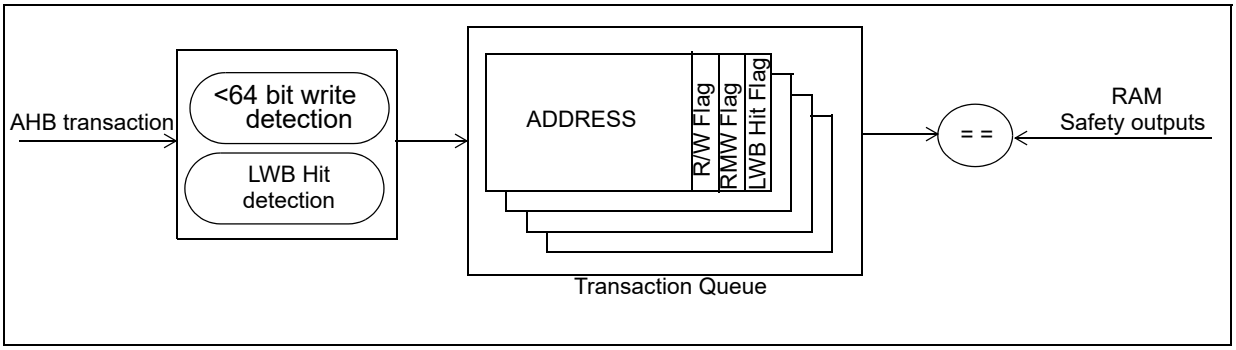
- Direct system bus read or write request
- Read followed by write to fulfill a read-modify write transaction
- Write transaction to empty the late-write buffer

The function of the transaction monitor relies on a feedback path between the RAM controller and RAM array, wherein the RAM provides latched address and control feedback outputs as an indication of received inputs when a RAM access is initiated.

This feedback information is used by the RAM controller transaction monitor to verify the expected transaction. If a mismatch is detected, indicating a failure in the address generation or control logic within the RAM controller or the transmission path between the RAM controller and RAM array, the event is forwarded to the Fault Collection and Control Unit (FCCU).

Since writes to the RAM can be buffered, the transaction monitor also tracks the contents of the late-write buffer. When the late-write buffer is emptied, the safety feature compare outputs from the RAM are evaluated against the expected contents of the late-write buffer as tracked by the transaction monitor.

Figure 285. RAM controller transaction monitor block diagram



31 Flash memory controller (PFLASH Controller)

31.1 Introduction

The flash memory controller:

- Acts as an interface between the system bus (AHB-Lite 2.v6) and the flash array.

The flash memory controller supports two 64-bit AHB buses and a 256-bit read data interface to the flash memory array. Each AHB port contains a 4-entry, 2-way set-associative mini-cache as well as an associated controller that prefetches sequential lines of data from the flash arrays into the mini-cache. This buffer mechanism serves to deliver flash read data with zero-wait state response on lines that reside in the cache. AHB requests that miss the cache generate the needed flash array access and are forwarded to the AHB upon completion. Each mini-cache entry is 256 bits in size, matching the code flash array page size and providing 256 bytes of high-speed local storage.

Along with the read and write data, the corresponding ECC codewords traverse the entire datapath of the flash memory controller, including the mini-cache, to support end-to-end ECC coverage (e2eECC).

31.2 Features

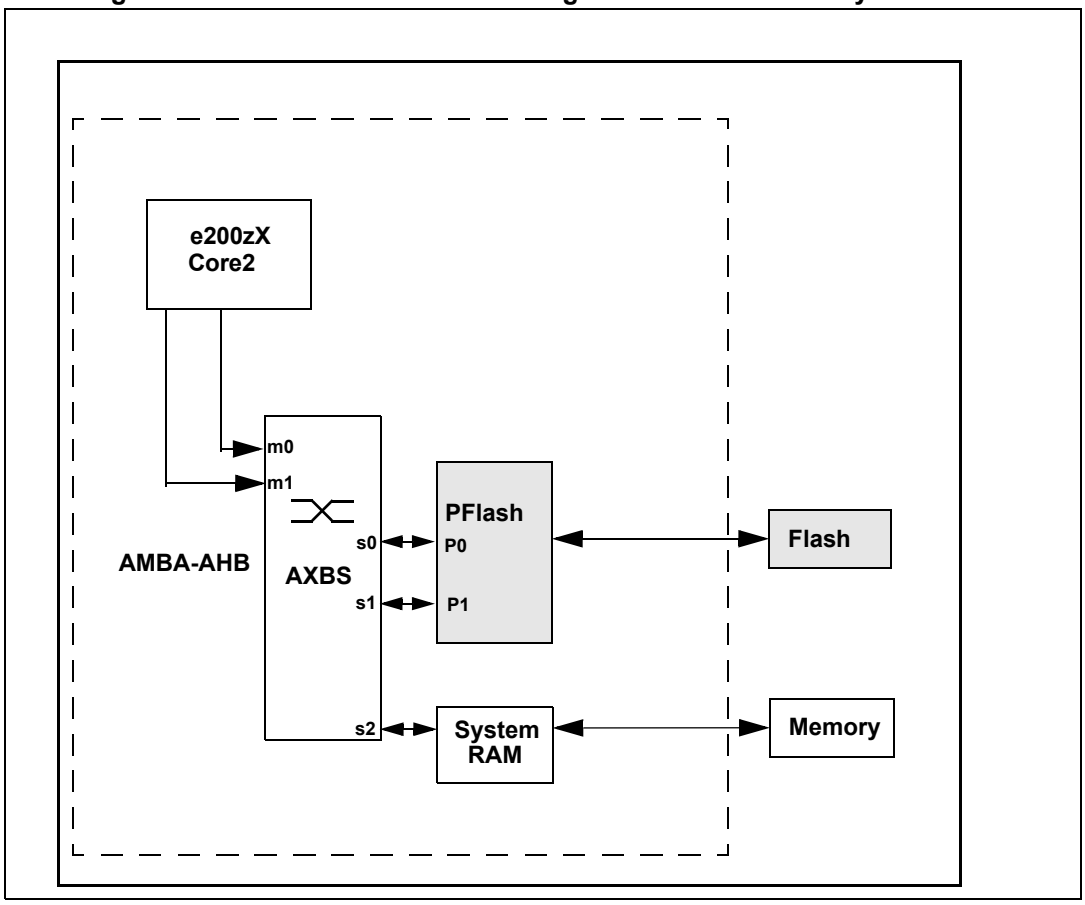
The following list summarizes the key features of the flash memory controller:

- Two 64-bit AHB ports (p0, p1)
- 256-bit read data bus + 64-bit write data bus
- Configurable read buffering and line prefetching support via 4-entry, 2-way set-associative mini-cache plus prefetch controller per AHB port to provide single-cycle “buffer hit” read response
- Configurable access control based on read/write and AHB master ID attributes
- Configurable access timing (wait-state programmable) allowing use in a wide range of frequency targets
- Support for reporting of single- and multi-bit flash ECC events on a 64-bit doubleword boundary

31.3 Block diagram

Figure 286 provides a block diagram showing the flash memory controller and the attached flash array.

Figure 286. Platform-centric block diagram with flash memory controller



31.4 Memory map and register definitions

31.4.1 Overview

The flash memory controller module provides an IPS programming model mapped to a standard 16 KB on-platform peripheral slot. The programming model consists of flash access configuration .

The programming model can only be referenced using a 32-bit (word) access. Attempted references using different access sizes or to undefined (reserved) addresses or in user mode generates an IPS error termination. The flash controller allows access to the programming model by all system bus masters.

The programming model can only be accessed in supervisor mode.

Attempted updates to the programming model while the flash memory controller module is in the middle of an operation results in non-deterministic behavior.

Table 379. Flash memory controller memory map

Address offset	Register	Location
0x000	Platform Flash Configuration Register 1 (PFCR1)	*Section 31.4.1.1*
0x004	Platform Flash Configuration Register 2 (PFCR2)	*Section 31.4.1.2*
0x008	Platform Flash Configuration Register 3 (PFCR3)	*Section 31.4.1.3*
0x00C	Platform Flash Access Protection Register (PFAPR)	*Section 31.4.1.4*
0x010–0x017	Reserved(1)	
0x018–0x0FF	Reserved(2)	
(0x100 + *n**0xF)–(0x10B + *n**0xF) (*n*=0 to 31)	Reserved(1)	
(0x10C + *n**0xF)–(0x10F + *n**0xF) (*n*=0 to 31)	Reserved(2)	
0x300–0x3FFF	Reserved(2)	

1. Access to this address space will not terminate in an error.
2. Access to this address space will terminate in an error.

31.4.1.1 Platform Flash Configuration Register 1 (PFCR1)

The PFlash Configuration Register 1 (PFCR1) is used to specify operation Port p0 of the flash memory controller.

Offset: 0x000 Access: Supervisor Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	P0_M15PFE	P0_M14PFE	P0_M13PFE	P0_M12PFE	P0_M11PFE	P0_M10PFE	P0_M9PFE	P0_M8PFE	P0_M7PFE	P0_M6PFE	P0_M5PFE	P0_M4PFE	P0_M3PFE	P0_M2PFE	P0_M1PFE	P0_M0PFE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	RWSC					0	P0_DPFEN	0	P0_IPFEN	0	P0_PFLIM		P0_BFEN
W																
Reset	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	1

Figure 287. Platform Flash Configuration Register 1 (PFCR1)

Table 380. PFCR1 field descriptions

Field	Description
0 P0_M15PFE	Port0 Master 15 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 15. This bit is cleared by hardware reset. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
1 P0_M14PFE	Port0 Master 14 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 14. This bit is cleared by hardware reset. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
2 P0_M13PFE	Port0 Master 13 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 13. This bit is cleared by hardware reset. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
3 P0_M12PFE	Port0 Master 12 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 12. This bit is cleared by hardware reset. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
4 P0_M11PFE	Port0 Master 11 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 11. This bit is cleared by hardware reset. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
5 P0_M10PFE	Port0 Master 10 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 10. This bit is cleared by hardware reset. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
6 P0_M9PFE	Port0 Master 9 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 9. This bit is cleared by hardware reset. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
7 P0_M8PFE	Port0 Master 8 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 8. This bit is cleared by hardware reset. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
8 P0_M7PFE	Port0 Master 7 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 7. This bit is cleared by hardware reset. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
9 P0_M6PFE	Port0 Master 6 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 6. This bit is cleared by hardware reset. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
10 P0_M5PFE	Port0 Master 5 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 5. This bit is cleared by hardware reset. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master

Table 380. PFCR1 field descriptions (continued)

Field	Description
11 P0_M4PFE	Port0 Master 4 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 4. This bit is cleared by hardware reset. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
12 P0_M3PFE	Port0 Master 3 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 3. This bit is cleared by hardware reset. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
13 P0_M2PFE	Port0 Master 2 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 2. This bit is cleared by hardware reset. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
14 P0_M1PFE	Port0 Master 1 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 1. This bit is cleared by hardware reset. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
15 P0_M0PFE	Port0 Master 0 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 0. This bit is cleared by hardware reset. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
19:23 RWSC	Read Wait State Control. This field controls the number of wait-states to be added to the best-case flash array access time for reads. The best-case flash array access time for reads is one cycle. This field must be set to a value corresponding to the operating frequency of the flash memory controller and the actual read access time of the flash memory controller. The required settings are documented in the SoC specification. Higher operating frequencies require non-zero settings for this field for proper flash operation. This field applies to the configuration of Port0 and Port1. 00000 No additional wait-states are added 00001 One additional wait-state is added ... 11111 Thirty-one additional wait-states are added **Note:** Refer to the device data sheet for wait-states configuration
25 P0_DPFEN	Port0 Data Prefetch Enable. This field enables or disables prefetching initiated by a data read access. This field is cleared by hardware reset. 0 No prefetching is triggered by a data read access 1 Prefetching may be triggered by any data read access
27 P0_IPFEN	Port0 Instruction Prefetch Enable. This bit enables or disables prefetching initiated by an instruction read access. This field is cleared by hardware reset. 0 No prefetching is triggered by an instruction read access 1 Prefetching may be triggered by any instruction read access

Table 380. PFCR1 field descriptions (continued)

Field	Description
29:30 P0_PFLIM	Port0 PFlash Prefetch Limit. This field controls the prefetch algorithm used by the prefetch controller. This field defines a limit on the maximum number of sequential prefetches which will be attempted between buffer misses. In all situations when enabled, only a single prefetch is initiated on each buffer miss or hit. This field is cleared by hardware reset. 00 No prefetching or buffering is performed. 01 The referenced line is prefetched on a buffer miss, that is, prefetch on miss. 10 The referenced line is prefetched on a buffer miss, or the next sequential line is prefetched on a buffer hit (if not already present), that is, prefetch on miss or hit. 11 The referenced line is prefetched on a buffer miss, or the next sequential line is prefetched on a buffer hit (if not already present), that is, prefetch on miss or hit.
31 P0_BFEN	Port0 PFlash Line Read Buffers Enable. This bit enables or disables line read buffer hits. It is also used to invalidate the buffers. This bit can only be updated while PFCR3[BAF_DIS] = 0. After execution of the boot code, as indicated by PFCR3[BAF_DIS] = 1, this bit becomes read-only until the next hardware reset. 0 The line read buffers are disabled from satisfying read requests, and all buffer valid bits are cleared. 1 The line read buffers are enabled to satisfy read requests on hits. Buffer valid bits may be set when the buffers are successfully filled.

31.4.1.2 Platform Flash Configuration Register 2 (PFCR2)

The PFlash Configuration Register 2 (PFCR2) is used to specify operation Port1 of the flash memory controller.

Offset: 0x004 Access: Supervisor Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	P1_M15PFE	P1_M14PFE	P1_M13PFE	P1_M12PFE	P1_M11PFE	P1_M10PFE	P1_M9PFE	P1_M8PFE	P1_M7PFE	P1_M6PFE	P1_M5PFE	P1_M4PFE	P1_M3PFE	P1_M2PFE	P1_M1PFE	P1_M0PFE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	P1_DPFEN	0	P1_IPFEN	0	P1_PFLIM		P1_BFEN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Figure 288. Platform Flash Configuration Register 2 (PFCR2)

Table 381. PFCR2 field descriptions

Field	Description
0 P1_M15PFE	Port1 Master 15 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 15. This bit is cleared by hardware reset. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
1 P1_M14PFE	Port1 Master 14 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 14. This bit is cleared by hardware reset. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
2 P1_M13PFE	Port1 Master 13 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 13. This bit is cleared by hardware reset. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
3 P1_M12PFE	Port1 Master 12 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 12. This bit is cleared by hardware reset. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
4 P1_M11PFE	Port1 Master 11 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 11. This bit is cleared by hardware reset. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
5 P1_M10PFE	Port1 Master 10 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 10. This bit is cleared by hardware reset. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
6 P1_M9PFE	Port1 Master 9 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 9. This bit is cleared by hardware reset. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
7 P1_M8PFE	Port1 Master 8 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 8. This bit is cleared by hardware reset. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
8 P1_M7PFE	Port1 Master 7 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 7. This bit is cleared by hardware reset. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
9 P1_M6PFE	Port1 Master 6 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 6. This bit is cleared by hardware reset. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
10 P1_M5PFE	Port1 Master 5 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 5. This bit is cleared by hardware reset. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master

Table 381. PFCR2 field descriptions (continued)

Field	Description
11 P1_M4PFE	Port1 Master 4 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 4. This bit is cleared by hardware reset. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
12 P1_M3PFE	Port1 Master 3 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 3. This bit is cleared by hardware reset. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
13 P1_M2PFE	Port1 Master 2 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 2. This bit is cleared by hardware reset. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
14 P1_M1PFE	Port1 Master 1 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 1. This bit is cleared by hardware reset. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
15 P1_M0PFE	Port1 Master 0 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 0. This bit is cleared by hardware reset. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
25 P1_DPFEN	Port1 Data Prefetch Enable. This field enables or disables prefetching initiated by a data read access. This field is cleared by hardware reset. 0 No prefetching is triggered by a data read access 1 Prefetching may be triggered by any data read access
27 P1_IPFEN	Port1 Instruction Prefetch Enable. This bit enables or disables prefetching initiated by an instruction read access. This field is cleared by hardware reset. 0 No prefetching is triggered by an instruction read access 1 Prefetching may be triggered by any instruction read access
29:30 P1_PFLIM	Port1 PFlash Prefetch Limit. This field controls the prefetch algorithm used by the prefetch controller. This field defines a limit on the maximum number of sequential prefetches which will be attempted between buffer misses. In all situations when enabled, only a single prefetch is initiated on each buffer miss or hit. This field is cleared by hardware reset. 00 No prefetching or buffering is performed. 01 The referenced line is prefetched on a buffer miss, that is, prefetch on miss. 1x The referenced line is prefetched on a buffer miss, or the next sequential line is prefetched on a buffer hit (if not already present), that is, prefetch on miss or hit.
31 P1_BFEN	Port1 PFlash Line Read Buffers Enable. This bit enables or disables line read buffer hits. It is also used to invalidate the buffers. This bit can only be updated while PFCR3[BAF_DIS]=0. After execution of the boot code, as indicated by PFCR3[BAF_DIS] = 1, this bit becomes read-only until the next hardware reset. 0 The line read buffers are disabled from satisfying read requests, and all buffer valid bits are cleared. 1 The line read buffers are enabled to satisfy read requests on hits. Buffer valid bits may be set when the buffers are successfully filled.

31.4.1.3 Platform Flash Configuration Register 3 (PFCR3)

Offset: 0x008 Access: Supervisor Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	P0_WCFG		P1_WCFG		0	0	0	0	0	0	0	Reserved(1)	0	0	0	BAF_DIS
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	ARBM		0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. This bit can be programmed but it has no functionality.

Figure 289. Platform Flash Configuration Register 3 (PFCR3)

Table 382. PFCR3 field descriptions

Field	Description
0:1 P0_WCFG	Port0 Way Configuration. This field controls the configuration of the line buffers for a given set across all four ways in the controller cache. The indexed set can be organized as a “pool” of available resources, or with a fixed partition between instruction and data buffers. In all cases, when a buffer miss occurs, the flash page is assigned a location within the controller cache. Within the indexed set, the way is selected using a least-recently-used replacement policy, and the entry is then marked as most-recently-used for that set. If the flash access is for the next-sequential line (prefetch), the buffer is not marked as most-recently-used until the given address produces a buffer hit. This field is initialized by hardware reset. 00 All four buffers in an indexed set are available for any flash access, that is, there is no partitioning of the buffers based on the access type. 01 Reserved 10 The buffers in an indexed set are partitioned into two groups with way 0 and 1 allocated for instruction fetches and way 2 and 3 for data accesses. 11 Reserved
2:3 P1_WCFG	Port1 Line Buffer Configuration. This field controls the configuration of the line buffers for a given set across all four ways in the controller cache. The indexed set can be organized as a “pool” of available resources, or with a fixed partition between instruction and data buffers. In all cases, when a buffer miss occurs, the flash page is assigned a location within the controller cache. Within the indexed set, the way is selected using a least-recently-used replacement policy, and the entry is then marked as most-recently-used for that set. If the flash access is for the next-sequential line (prefetch), the buffer is not marked as most-recently-used until the given address produces a buffer hit. This field is initialized by hardware reset. 00 All four buffers are available for any flash access, that is, there is no partitioning of the buffers based on the access type. 01 Reserved 10 The buffers are partitioned into two groups with buffers 0 and 1 allocated for instruction fetches and buffers 2 and 3 for data accesses. 11 Reserved
11	Reserved This is a Read only bit. It can be programmed but it has no functionality.

Table 382. PFCR3 field descriptions (continued)

Field	Description
15 BAF_DIS	BAF Disable. This field controls executable access to the BAF (Boot Assist Flash) region of the flash, where the access type is detected by evaluating {p0,p1}_hprot[0]. Once this field is set, attempted instruction access attempts targeting the BAF region are aborted and terminated with a system bus error. The affect of this field applies to system bus transfers through Port0 and Port1. Data-type accesses to the BAF region are not affected by this field. Once this field is set, it becomes a read-only field and can only be cleared by hardware reset. Once this field is set, any subsequent write attempts to modify this field are ignored with an error-free data transfer termination. This field is initialized by hardware reset. 0 Executable access to the BAF flash region is allowed. 1 Executable access to the BAF flash region is prohibited.
16:17 ARBM	Arbitration Mode. This 2-bit field controls the arbitration of concurrent flash access requests from the two AHB ports of the flash memory controller. In both fixed priority or round-robin modes, write requests are prioritized higher than read requests, and read requests are prioritized higher than speculative prefetch requests whenever both ports issue concurrent requests. This field is initialized by hardware reset. 00 Fixed priority arbitration with AHB p0>p1 01 Fixed priority arbitration with AHB p1>p0 1x Round-robin arbitration

31.4.1.4 Platform Flash Access Protection Register (PFAPR)

The PFlash Access Protection Register (PFAPR) is used to control read and write accesses to the flash array.

Offset: 0x00C Access: Supervisor Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	M0AP		M1AP		M2AP		M3AP		M4AP		M5AP		M6AP		M7AP	
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	M8AP		M9AP		M10AP		M11AP		M12AP		M13AP		M14AP		M15AP	
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Figure 290. Platform Flash Access Protection Register (PFAPR)

Table 383. PFAPR field descriptions

Field	Description
0:1 M0AP	Master 0 Access Protection. This field controls whether read and write accesses to the flash are allowed based on the master ID of a requesting master. These fields are initialized by hardware reset. 00 No accesses may be performed by this master 01 Only read accesses may be performed by this master 10 Only write accesses may be performed by this master 11 Both read and write accesses may be performed by this master
2:3 M1AP	Master 1 Access Protection. This field controls whether read and write accesses to the flash are allowed based on the master ID of a requesting master. These fields are initialized by hardware reset. 00 No accesses may be performed by this master 01 Only read accesses may be performed by this master 10 Only write accesses may be performed by this master 11 Both read and write accesses may be performed by this master
4:5 M2AP	Master 2 Access Protection. This field controls whether read and write accesses to the flash are allowed based on the master ID of a requesting master. These fields are initialized by hardware reset. 00 No accesses may be performed by this master 01 Only read accesses may be performed by this master 10 Only write accesses may be performed by this master 11 Both read and write accesses may be performed by this master
6:7 M3AP	Master 3 Access Protection. This field controls whether read and write accesses to the flash are allowed based on the master ID of a requesting master. These fields are initialized by hardware reset. 00 No accesses may be performed by this master 01 Only read accesses may be performed by this master 10 Only write accesses may be performed by this master 11 Both read and write accesses may be performed by this master
8:9 M4AP	Master 4 Access Protection. This field controls whether read and write accesses to the flash are allowed based on the master ID of a requesting master. These fields are initialized by hardware reset. 00 No accesses may be performed by this master 01 Only read accesses may be performed by this master 10 Only write accesses may be performed by this master 11 Both read and write accesses may be performed by this master
10:11 M5AP	Master 5 Access Protection. This field controls whether read and write accesses to the flash are allowed based on the master ID of a requesting master. These fields are initialized by hardware reset. 00 No accesses may be performed by this master 01 Only read accesses may be performed by this master 10 Only write accesses may be performed by this master 11 Both read and write accesses may be performed by this master
12:13 M6AP	Master 6 Access Protection. This field controls whether read and write accesses to the flash are allowed based on the master ID of a requesting master. These fields are initialized by hardware reset. 00 No accesses may be performed by this master 01 Only read accesses may be performed by this master 10 Only write accesses may be performed by this master 11 Both read and write accesses may be performed by this master

Table 383. PFAPR field descriptions (continued)

Field	Description
14:15 M7AP	Master 7 Access Protection. This field controls whether read and write accesses to the flash are allowed based on the master ID of a requesting master. These fields are initialized by hardware reset. 00 No accesses may be performed by this master 01 Only read accesses may be performed by this master 10 Only write accesses may be performed by this master 11 Both read and write accesses may be performed by this master
16:17 M8AP	Master 8 Access Protection. This field controls whether read and write accesses to the flash are allowed based on the master ID of a requesting master. These fields are initialized by hardware reset. 00 No accesses may be performed by this master 01 Only read accesses may be performed by this master 10 Only write accesses may be performed by this master 11 Both read and write accesses may be performed by this master
18:19 M9AP	Master 9 Access Protection. This field controls whether read and write accesses to the flash are allowed based on the master ID of a requesting master. These fields are initialized by hardware reset. 00 No accesses may be performed by this master 01 Only read accesses may be performed by this master 10 Only write accesses may be performed by this master 11 Both read and write accesses may be performed by this master
20:21 M10AP	Master 10 Access Protection. This field controls whether read and write accesses to the flash are allowed based on the master ID of a requesting master. These fields are initialized by hardware reset. 00 No accesses may be performed by this master 01 Only read accesses may be performed by this master 10 Only write accesses may be performed by this master 11 Both read and write accesses may be performed by this master
22:23 M11AP	Master 11 Access Protection. This field controls whether read and write accesses to the flash are allowed based on the master ID of a requesting master. These fields are initialized by hardware reset. 00 No accesses may be performed by this master 01 Only read accesses may be performed by this master 10 Only write accesses may be performed by this master 11 Both read and write accesses may be performed by this master
24:25 M12AP	Master 12 Access Protection. This field controls whether read and write accesses to the flash are allowed based on the master ID of a requesting master. These fields are initialized by hardware reset. 00 No accesses may be performed by this master 01 Only read accesses may be performed by this master 10 Only write accesses may be performed by this master 11 Both read and write accesses may be performed by this master
26:27 M13AP	Master 13 Access Protection. This field controls whether read and write accesses to the flash are allowed based on the master ID of a requesting master. These fields are initialized by hardware reset. 00 No accesses may be performed by this master 01 Only read accesses may be performed by this master 10 Only write accesses may be performed by this master 11 Both read and write accesses may be performed by this master

Table 383. PFAPR field descriptions (continued)

Field	Description
28:29 M14AP	Master 14 Access Protection. This field controls whether read and write accesses to the flash are allowed based on the master ID of a requesting master. These fields are initialized by hardware reset. 00 No accesses may be performed by this master 01 Only read accesses may be performed by this master 10 Only write accesses may be performed by this master 11 Both read and write accesses may be performed by this master
30:31 M15AP	Master 15 Access Protection. This field controls whether read and write accesses to the flash are allowed based on the master ID of a requesting master. These fields are initialized by hardware reset. 00 No accesses may be performed by this master 01 Only read accesses may be performed by this master 10 Only write accesses may be performed by this master 11 Both read and write accesses may be performed by this master

31.5 Functional description

As shown in *Figure 286* the flash memory controller interfaces between the dual AHB system bus ports, flash memory array,.

For accesses targeting the flash array, the flash memory controller generates read and write enables, array address, write size and write data as inputs to the flash array. The flash memory controller captures read data from the flash array and drives it onto the AHB system bus. Up to four pages of data (256-bit page size) may be buffered in each of two ways of the flash memory controller mini-cache. Lines may be prefetched in advance of being requested, allowing single-cycle (zero AHB wait-states) read data responses on buffer hits.

Several prefetch control algorithms are available for controlling line read buffer fills. Prefetch triggering may be restricted to instruction accesses only, data accesses only, or may be unrestricted. Prefetch triggering may also be controlled on a per-master basis.

Buffers may also be selectively enabled or disabled for allocation by instruction and data prefetch.

Access protections may be applied on a per-master basis for both reads and writes to support security and privilege mechanisms.

31.5.1 Basic interface protocol

Read accesses are terminated under control of the appropriate wait state settings. Thus, the access time of the operation is determined by the setting of the PFCR1[RWSC] field. Access timing can be varied to account for the operating conditions of the SoC (frequency, voltage, temp) by appropriately setting the PFCR1[RWSC] field.

31.5.2 Access protections

31.5.2.1 PFAPR - Platform Flash Access Protection Register (PFAPR)

The flash memory controller provides programmable, configurable access protections for both read and write cycles on a per-master basis via the PFlash Access Protection Register (PFAPR). It allows restriction of read and write requests on a per-master basis. This functionality is described in *Section 31.4.1.4: Platform Flash Access Protection Register (PFAPR)*. Detection of a protection violation results in an error response from the flash memory controller on the AHB transfer.

31.5.2.2 BAF execution disable

The flash memory controller provides a mechanism to disable instruction execution of the BAF (Boot Assist Flash) code, which is loaded in a predetermined OTP flash block. Setting the BAF Disable field (BAF_DIS) in the PFlash Configuration Register 3 (PFCR3) prevents execution of the BAF instructions. When this field is set, all incoming requests from the system bus on either Port0 or Port1 are evaluated. If the flash memory controller detects an attempted instruction fetch access to the BAF region of the flash, the flash memory controller aborts the access and returns a system bus error. Data accesses to the BAF region are not affected by the state of PFCR3[BAF_DIS].

31.5.3 Read cycles - buffer miss

On an incoming AHB read request, a buffer lookup and access checks are performed during the AHB address phase. In the event of a buffer miss the AHB address and attributes are registered and the flash access is initiated the cycle following the AHB address phase (the first AHB data phase cycle).

If the flash access was the direct result of an AHB transaction, the corresponding page buffer is marked as most-recently-used as it is being loaded. If the flash access was the result of a speculative prefetch to the next sequential line, it is first loaded into the least-recently-used buffer. The status of this buffer is not changed to most-recently-used until a subsequent buffer hit occurs as a result of an AHB read request.

31.5.4 Read cycles – buffer hit

Single cycle read responses to the AHB are possible with the flash memory controller when the requested read access was previously loaded into one of the page buffers. In these “buffer hit” cases, read data is returned to the AHB data phase with a zero wait-state response.

31.5.5 Error termination

The flash memory controller may invoke a system bus error termination in the following scenarios:

- Attempted access by an AHB master whose corresponding Read Access Control or Write Access Control settings do not allow the access, thus causing a protection violation, see *Section 31.4.1.4: Platform Flash Access Protection Register (PFAPR)* for more detail. In this case the flash memory controller does not initiate a flash array access.
- Attempted instruction fetch by an AHB master to the BAF flash region when the PFCR3[BAF_DISABLE] field in Platform Flash Configuration Register 3 (PFCR3) is

set. The assertion of PFCR3[BAF_DISABLE] indicates that code stored in the BAF region should not be executed, see *Section 31.4.1.3: Platform Flash Configuration Register 3 (PFCR3)* for more detail. In this case, the flash memory controller does not initiate a flash array access.

- Attempted access by an AHB master to a flash region, where the corresponding censorship control is asserted.
- The flash returns an error response on an attempted flash access (See *Section 31.5.6: Flash error response operation*).

31.5.6 Flash error response operation

The flash array may signal an error response to terminate a requested access due to improper sequencing during program/erase operations. When an error response is received, the flash memory controller does not update or validate a page read buffer. An error response may be signaled on read operations. For more information on the specifics related to signaling of flash errors including flash ECC events and read-while-write events, refer to the Embedded Flash Memory chapter.

31.5.7 Censorship

The entire flash space is defined by three major regions:

- Code flash (instruction and constant data)
- Data flash (data for EEPROM emulation)
- UTest flash

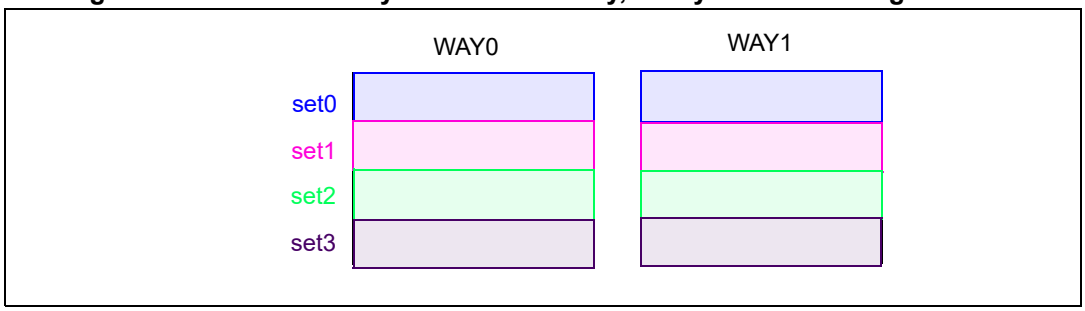
Each section can be independently censored, and the flash memory controller provides independent censorship control inputs for each region on a read vs. write basis.

31.5.8 Line read buffers and prefetch operation

Each AHB port of the flash memory controller contains a two-way set-associative mini cache, where each way contains four page buffers which are used to hold data read from the flash arrays. Each 256-bit buffer operates independently, and is filled using a single array access. The buffers are used for both prefetch and normal demand fetches.

Prefetch triggering is controllable on a per-master and access-type basis (see *Section 31.4.1.1: Platform Flash Configuration Register 1 (PFCR1)*). Bus masters may be enabled or disabled from triggering prefetches, and triggering may be further restricted based on whether a read access is for instruction or data. A read access by the flash memory controller may trigger a prefetch to the next sequential line of array data on the cycle following the request. The access address is incremented by 32-bytes, and a subsequent flash access is initiated. A flash array prefetch is initiated if the data is not already resident in a line read buffer. Prefetched data is always loaded into the least-recently-used buffer.

Figure 291. Flash memory controller 4-entry, 2-way mini-cache organization



Once the candidate line buffer has been selected, the flash array is accessed and read data loaded into the buffer. If the buffer load was in response to a miss, the just-loaded buffer is immediately marked as most-recently-used. If the buffer load was in response to a speculative fetch to the next-sequential line address after a buffer hit, *the recently-used status is not changed*. Rather, it is marked as most-recently-used only after a subsequent buffer hit.

This policy maximizes performance based on reference patterns of flash accesses and allows for prefetched data to remain valid when non-prefetch enabled bus masters are granted flash access.

Several algorithms are available for prefetch control which trade off performance for power. They are described in *Section 31.4.1.1: Platform Flash Configuration Register 1 (PFCR1)*. More aggressive prefetching may increase power due to the number of potentially discarded prefetches, but may increase performance by lowering average read latency.

In order for prefetching to occur, PFCR{1,2}[P{0,1}_BFEN] must be set to '1', PFCR{1,2}[P{0,1}_PFLIM] must be non-zero, and either PFCR{1,2}[P{0,1}_IPFEN] or PFCR{1,2}[P{0,1}_DPFEN] must be asserted. Refer to *Section 31.4.1.1: Platform Flash Configuration Register 1 (PFCR1)* for a description of these controls.

31.5.9 Instruction/Data prefetch triggering

Port0 prefetch triggering may be enabled for instruction reads via the PFCR1[IPFEN] control field, while Port0 prefetching for data reads is enabled via the PFCR1[DPFEN] control field. Additionally, the PFCR1[PFLIM] must also be set to enable prefetching on Port0. Refer to *Section 31.4.1.1: Platform Flash Configuration Register 1 (PFCR1)* for a description of these controls. Prefetches are never triggered by write cycles.

Port1 prefetch triggering may be enabled for instruction reads via the PFCR2[IPFEN] control field, while Port1 prefetching for data reads is enabled via the PFCR2[DPFEN] control field. Additionally, the PFCR2[PFLIM] must also be set to enable prefetching on Port1. Refer to *Section 31.4.1.2: Platform Flash Configuration Register 2 (PFCR2)* for a description of these controls. Prefetches are never triggered by write cycles.

31.5.10 Per-Master prefetch triggering

Prefetch triggering may be controlled for individual bus masters. Refer to *Section 31.4.1.1: Platform Flash Configuration Register 1 (PFCR1)* for a description of these controls.

31.5.11 Buffer allocation

Allocation of the line read buffers is controlled via the PFCR3 control register, specifically the line buffer configuration ({P0,P1}_WCFG) field. Refer to *Section 31.4.1.3: Platform*

Flash Configuration Register 3 (PFCR3). The buffers can be organized as a “pool” of available resources (with both ways within a given set) or with a fixed partition between ways allocated to instruction or data accesses. For the fixed partitions, ways 0 is allocated for instruction fetches and way1 for data accesses.

31.5.12 Safety considerations

31.5.12.1 e2eECC (End-to-End ECC)

31.5.12.1.1 e2eECC and flash accesses

There are multiple complications associated with the use of the standard e2eECC algorithm with flash memory. The basic issues involve: the default state of a flash block that is erased (all ones) and the fact that programming an erased flash involves changing the state of memory bit from a logical 1 to a logical 0. These factors require that the system level e2eECC must be modified on references to the flash memory in three ways.

First, the all ones codeword, since it reflects the state of an erased flash location, is treated as a valid, error-free encoding. Second, since an erased flash block generates an all ones data for all locations, the address field cannot be included in the ECC code used by the flash. The flash array implements special address re-encoding logic based on the decoded array address to protect against address faults. Third, there are additional ECC implications associated with the flash memory and its support for EEPROM emulation; in particular, it is required that the checkbits associated with the all ones data value and the all zeroes data value are required to be the same with a checkbit value of 0xFF.

In summary, the basic operation of an NVM bit cell impacts the e2eECC algorithm used in the flash memory in multiple ways. The required ECC adjustments are handled by the platform flash controller before data is written to the flash array(s) or applied to read data accessed from the array(s). These adjustments are two-fold. Consider a flash write during a programming event:

1. Remove the address field from the ECC codeword by XOR’ing the address calculation of the H-matrix from the checkbits.
2. Invert the resulting 8-bit ECC checkbit vector. This step is required to support the all ones erased state.

On a flash read operation, a similar but “reversed” set of steps are performed as the data is driven into the system bus interconnect:

1. Invert the 8-bit ECC checkbit vector read from the flash.
2. Factor the address field into the ECC checkbits by XOR'ing the address calculation of the H-matrix.

Consider the following flash ECC example. Let the flash block containing address 0x00345678 be erased. The following is the sequence of operations:

1. Read address 0x00345678. Flash array returns fl_rdata = 0xFFFFFFFF_FFFFFFFF, fl_cdata = 0xFF. The flash memory controller calculates the addr_chkbit = 0xC4 using the H-matrix; it inverts the fl_cdata vector to 0x00 and factors in the addr_chkbit = 0xC4 to produce the following valid e2eECC codeword:

addr = 0x00345678, rdata = 0xFFFFFFFF_FFFFFFFF, rchkbit = 0xC4 (0x00 ^ 0xC4)

2. Program address 0x00345678 with data = 0xBABEFACE_DEADBEEF. The initiating bus master calculates the e2eECC codeword as:

addr = 0x00345678, wdata = 0xBABEFACE_DEADBEEF, wchkbit = 0xE3

Before performing the interlock write to the flash array, the flash memory controller adjusts the chkbit value as fl_wchkbit = 0xD8 (0xE3 ^ 0xff ^ 0xC4) by toggling the original write checkbits and then factoring in the addr_ecc. Accordingly, the codeword stored in the flash array is:

fl_data = 0xBABEFACE_DEADBEEF, fl_checkbit = 0xD8

3. Read address 0x00345678. The flash array returns the stored codeword, fl_rdata = 0xBABEFACE_DEADBEEF, fl_rchkbit = 0xD8. The flash memory controller inverts the checkbit vector and factors in the address checkbits (0xD8 ^ 0xFF ^ 0xC4) to produce:

addr = 0x00345678, rdata = 0xBABEFACE_DEADBEEF, rchkbit = 0xE3

This read codeword matches the original e2eECC codeword generated by the write.

Malfunction of the ECC logic described above may result in the corruption of the SEC/DED event reporting. The flash memory controller performs EDC after ECC check on all flash transactions. If a mismatch is detected, indicating a failure in the ECC logic, the event is reported to the fault collection module.

31.5.12.1.2 e2eECC and data flash accesses

ECC events detected on accesses to the data flash blocks are suppressed from being reported to the ECC event collection unit.

In the event of a single-bit correction, the corrected data and checkbits are returned to the requesting master, and the single-bit correction event is suppressed from being reported to the MEMU.

In the event of a non-correctable error detection, a fixed, illegal opcode, value is returned to the requesting master along with the associated ECC checkbits as determined by the requesting address and the non-correctable error event is suppressed from being reported to the MEMU.

31.5.12.2 Flash address generation check

Functional safety coverage of the address and data are handled by ECC performed within the flash and e2eECC performed at the master. Functional safety coverage of the address path and control within the flash memory controller rely on a feedback path between the flash controller and flash. On a requested access to flash, the flash memory controller must decode the system AHB bus signals to generate the corresponding flash interface signals which invoke a flash lookup. In addition to providing the requested read data, the flash also provides output sidebands reflecting the encoded address and block selects used to perform the actual row lookup:

- Encoded address as received by flash memory
- Address encoding error provided by flash memory

This sideband information is used by the flash memory controller to verify the expected transaction. If a mismatch is detected, indicating a failure in the address generation or control logic within the flash memory controller or the transmission path between the flash memory controller and flash array, the event is forwarded to the Fault Collection and Control Unit (FCCU) and the corresponding buffer is invalidated. The flash memory controller has a separate MEMU reporting interface associated with AHB port p0 and p1.

31.5.13 Array integrity considerations

During an array integrity sequence, the flash array ignores any incoming read requests. The flash provides an output sideband, which indicates when an array integrity sequence is in progress. When it is asserted, the flash memory controller aborts any incoming flash access requests and terminates the system bus transfer with an error.

32 Embedded Flash Memory

32.1 Introduction

The Flash memory module serves as electrically programmable and erasable nonvolatile memory, for instructions and data storage.

The Flash memory module is a nonvolatile, solid-state silicon memory device consisting of blocks of single transistor storage elements, an electrical means for selectively adding (programming) and removing (erasing) charge from these elements, and a means of selectively sensing (reading) the charge stored in these elements.

The Flash memory module has two functional units:

- **The Flash core.**
  It is composed of a common high voltage HV block and a Flash array which can contain one or more Flash partitions. Each Flash partition is composed of arrayed nonvolatile storage elements, sense amplifiers, row decoders and column decoders. The arrayed storage elements in the Flash partitions are subdivided into physically separate units referred to as blocks.
- **The Memory interface.**
  It contains the registers and logic which control the operation of the Flash core and comprises two main ports: the Array interface and the Registers Interface. The memory interface is also the interface between the Flash memory module and a Bus Interface Unit (BIU) and contains the ECC and redundancy logic.

A BIU connects the Flash memory module to a system bus, and contains all the system level customization required for the SoC application. The Flash memory module is generic and requires a BIU to configure it for different SoC applications. A BIU is not included.

32.1.1 Overview

The 1632 KB Flash memory module contains two array partitions. Read-While-Write operations are only possible for separate partitions, meaning fetching/reading from one (or more) partition(s) while a Program/Erase operation is active on another partition. Each module counts as an RWW partition.

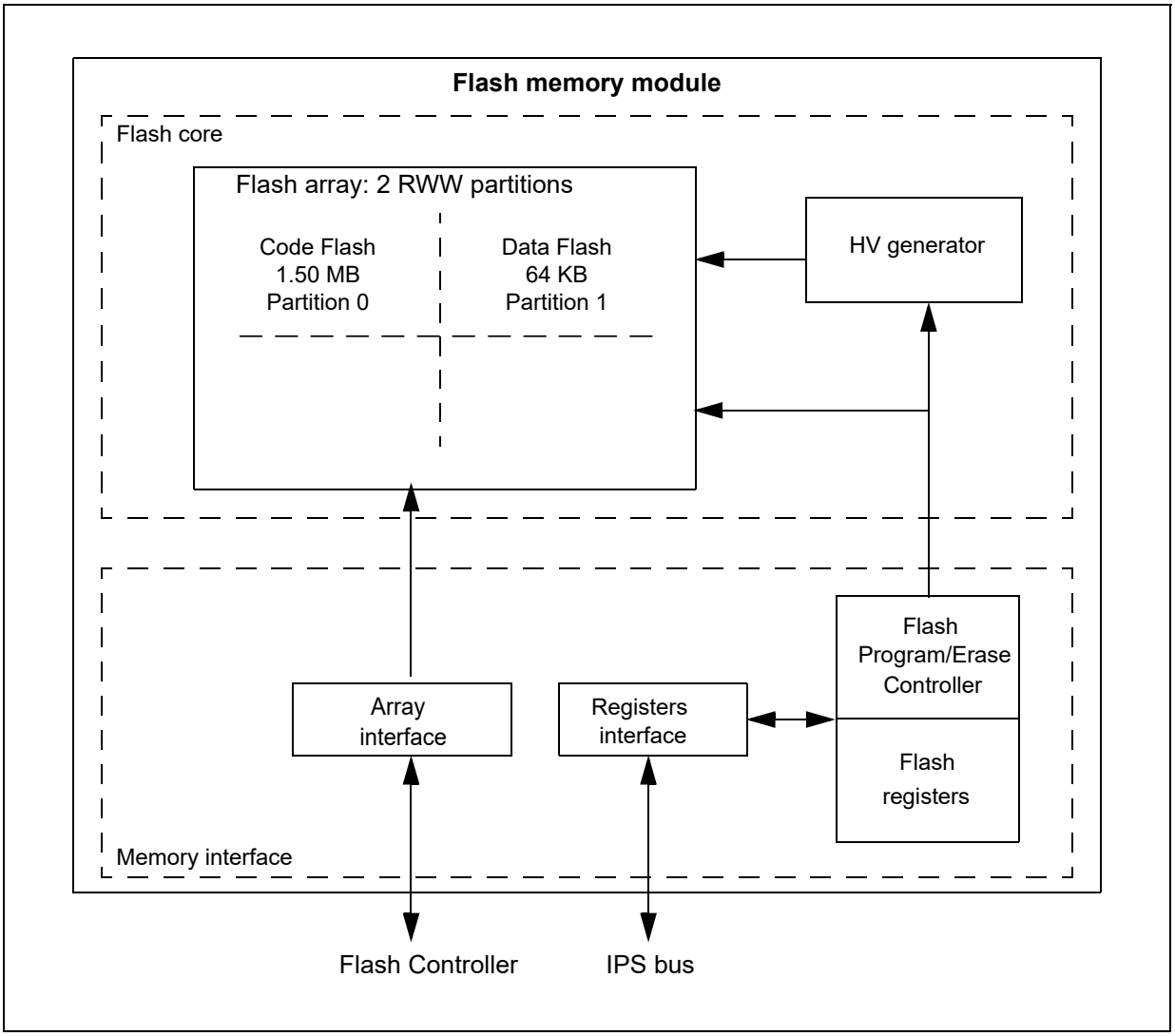
EEPROM emulation is managed in one block of the module, referred to as Data Flash.

The Modify operations are managed by an embedded Flash Program/Erase Controller (FPEC). Commands to the FPEC are delivered through a User Registers interface.

The read data bus is 256 wide, while the Flash registers are on a separate, 32-bit bus.

The high voltages needed for Program and Erase operations are internally generated and are common for the two partitions.

Figure 292. Flash memory module structure



The Flash memory module is designed for use in embedded MCU/SoC applications which require high density nonvolatile memories with high-speed read access.

The Flash array is addressable:

- For programs: by word (32 bits) or doubleword (64 bits).
- For reading code partitions: by page (256 bits).
- For reading data partitions: by half-page (128 bits).

Multipleword or doubleword Writes to the Flash memory may be performed to fill the program page buffer (256 bits), enabling page programming (256 bits, requiring 4 doubleword Writes) and quad-page programming (1024 bits, requiring 16 doubleword Writes). Code Flash memory Reads always return 256 bits, although read buffering may be performed by the Bus Interface Unit (BIU). The address for each word retrieved within a page differs from the other addresses in the page by address bits (3:2).

Data Flash memory Reads always return 128 bits, or two consecutive doublewords of information. The address for each word retrieved within a doubleword differs from the other addresses by address bits (3:2).

The Flash memory module supports fault tolerance through Error Correction Code (ECC) and error detection. The ECC implemented within the Flash memory module corrects single bit failures and detects double bit failures.

The Flash memory module uses an embedded hardware algorithm implemented in the Flash Program/Erase Controller to program and erase the Flash partitions.

Control logic that works with the software block-enables and software lock mechanisms are included in the embedded hardware algorithm to guard against accidental program/erase. The hardware algorithm performs the necessary steps to ensure that the storage elements are programmed and erased with sufficient margin to guarantee data integrity and reliability.

A programmed bit in the Flash array reads as logic level ‘0’ (or low). An erased bit in the Flash array reads as logic level ‘1’ (or high).

Programming and erasing of the Flash array requires multiple system clock cycles to complete.

The Program and Erase sequences may be suspended or aborted.

32.1.2 Features

- 30 ns code/data array access time
- Code Partition High Read parallelism: 256 bits
- Data Partition Read parallelism: 128 bits
- Error Correction Code (SEC-DED) to enhance data integrity
- Doubleword program granularity: 64 bits
- Erase with fast erase time at factory
- Blocks one time programmable (OTP) marking
- Read-While-Modify capability when the accesses are to different partitions
- Erase/Program Suspend
- Programming during Erase Suspend
- Software programmable program/erase protection to avoid unwanted writings
- Temperature range (-40 °C to 150 junction)
- UTEST mode (user-accessible Test modes)
- Critical flops, which may affect Flash content, are triple-voted for safety application.

32.1.3 Modes of operation

The Flash memory module supports the following modes of operation:

- Reset: the Flash memory module initializes when the reset is deasserted.
- Power-down (Disable mode): the Flash memory module removes all static consumption. The embedded Flash memory (array and registers) is not accessible to Read, Program or Erase once disabled. The wake-up time from Disable mode is longer than the wake-up time from Sleep mode.
- Low-power mode (Sleep mode): the Flash memory module removes most of the static consumption. It is not accessible to Read, Program or Erase once disabled, however it does maintain a faster exit to Read mode with respect to power-down.
- Read mode: the Flash memory module accepts Read accesses to the array.

- Modify mode: it is possible to read and write registers, interlock write the memory array, program the memory array, and erase the memory array. Program and Erase operations are initiated by performing array and register writes, and controlled by an internal state machine. Program/Erase operations or User Test modes (Margin Read or Array Integrity Check) are considered Modify modes.
- Read-While-Write: The Flash memory module accepts Read accesses to array partitions that are not in Modify mode.

Such modes of operation are described in detail in *Section 32.4: Functional description*.

32.2 Flash memory map and description

The Flash memory module general architecture supports total memory sizes ranging from 32 KB to 8 MB of user memory, plus up to 48 KB of Test memory (a portion of which is One Time Programmable by the user).

Optionally, an extra memory space ranging from 32 KB to 512 KB can be used as data space for EEPROM emulation, and can be organized in different RWW partitions (in addition to the main user memory partition).

There are four User Address Spaces: Low, Mid, High and 256K Address Space:

- Low Address Space
  it must always be present and be up to 256 KB in size and may be organized in a maximum of 14 blocks (16 KB[k], 32 KB and 64 KB blocks).
- Mid Address Space
  it can be present and be up to 512 KB in size and may be organized in a maximum of 16 blocks (16 KB, 32 KB and 64 KB blocks).
- High Address Space
  it can be present and be up to 512 KB may be organized in a maximum of 16 small blocks (8 KB[k], 16 KB, 32 KB and 64 KB blocks).
- 256K Address Space
  it can be present and be up to 7.5 MB in size and may be organized in a maximum of 31 blocks (256 KB).

Note: There is also a special Test Address space which must always be present and be up to 48 KB in size. Test Address space is part of Flash Partition 0.

The Flash memory module can be divided into a minimum of one to a maximum of eight Flash partitions.

The total size of each Code Flash partition cannot be greater than 2 MB and the maximum number of blocks cannot exceed 16, including TestFlash. The total size of each Data Flash partition cannot be greater than 256 KB and the maximum number of blocks cannot exceed eight.

There are up to six sizes of blocks available for user Flash partitions: 256 KB, 128 KB, 64 KB, 32 KB, 16 KB and 8 KB.
The 64 KB, 32 KB and 16 KB blocks can be mapped anywhere in the Low, Mid and High address spaces.

k. If the BAF is 8 KB in size, it is integrated with the 8 KB UTEST area to form a single 16 KB block in Low Address Space.

The 256 KB blocks can be mapped in the 256K address space. 8 KB[(k)] blocks may only be mapped in the High address space.

The Flash memory module block subdivision also facilitates independent Erase and Program protection. A software mechanism is provided to independently lock and unlock each block in Low, Mid, High and Data address spaces against program and erase.

32.2.1 Flash array memory map

32.2.1.1 Code Flash

Partition 0 accounts for 1.50 MB of the total Flash memory and is divided into 13 user blocks for code storage. The first 256 KB block is organized as 4 × 16 KB, 2 × 32 KB and 2 × 64 KB. Remaining blocks are 256 KB.

Partition 0 also contains a reserved block named TestFlash in which some One Time Programmable (OTP) user data is stored.

The sectorization of partition 0 of the 1632 KB Flash memory module is shown in *Table 384*.

Table 384. Flash memory partition 0 memory map

Block	Absolute addresses	Size	Address space	LOCK	SEL
B0BF	0x0040_4000 to 0x0040_7FFF	16 KB	Low	LOWLOCK[0]	—
B0F1	0x00FC_0000 to 0x00FC_3FFF	16 KB	Low	LOWLOCK[1]	LOWSEL[1]
B0F2	0x00FC_4000 to 0x00FC_7FFF	16 KB	Low	LOWLOCK[2]	LOWSEL[2]
B0F3	0x00FC_8000 to 0x00FC_BFFF	16 KB	Low	LOWLOCK[3]	LOWSEL[3]
B0F4	0x00FC_C000 to 0x00FC_FFFF	16 KB	Low	LOWLOCK[4]	LOWSEL[4]
B0F5	0x00FD_0000 to 0x00FD_7FFF	32 KB	Low	LOWLOCK[5]	LOWSEL[5]
B0F6	0x00FD_8000 to 0x00FD_FFFF	32 KB	Low	LOWLOCK[6]	LOWSEL[6]
B0F7	0x00FE_0000 to 0x00FE_FFFF	64 KB	Low	LOWLOCK[7]	LOWSEL[7]
B0F8	0x00FF_0000 to 0x00FF_FFFF	64 KB	Low	LOWLOCK[8]	LOWSEL[8]
B0F9	0x0100_0000 to 0x0103_FFFF	256 KB	256 K	A256KLOCK[0]	A256KSEL[0]
B0F10	0x0104_0000 to 0x0107_FFFF	256 KB	256 K	A256KLOCK[1]	A256KSEL[1]
B0F11	0x0108_0000 to 0x010B_FFFF	256 KB	256 K	A256KLOCK[2]	A256KSEL[2]
B0F12	0x010C_0000 to 0x010F_FFFF	256 KB	256 K	A256KLOCK[3]	A256KSEL[3]
B0F13	0x0110_0000 to 0x0113_FFFF	256 KB	256 K	A256KLOCK[4]	A256KSEL[4]
B0UT	0x0040_0000 to 0x0040_3FFF	16 KB	Low	TSLOCK	—
B0TF	0x0040_8000 to 0x0040_BFFF	16 KB	Reserved	—	—

32.2.1.2 Data Flash

Data Flash is allocated in partition one of the Flash memory module in 4 × 16 blocks destined for data storage.

The Data Flash sectorization of the Flash memory module is listed below.

Table 385. Flash memory Partition 1 memory map

Block	Absolute addresses	Size	Address Space	LOCK	SEL
B1F0	0x0080_0000 to 0x0080_3FFF	16 KB	High	HIGHLOCK[0]	HIGHSEL[0]
B1F1	0x0080_4000 to 0x0080_7FFF	16 KB	High	HIGHLOCK[1]	HIGHSEL[1]
B1F2	0x0080_8000 to 0x0080_BFFF	16 KB	High	HIGHLOCK[2]	HIGHSEL[2]
B1F3	0x0080_C000 to 0x0080_FFFF	16 KB	High	HIGHLOCK[3]	HIGHSEL[3]
—	—	—	—	—	—

32.2.1.3 TestFlash block memory map

The TestFlash block exists outside the normal address space and is programmed, erased and read independently of the other blocks. The independent TestFlash block is also included to support systems which require nonvolatile memory for security and to store system initialization information.

A section of the TestFlash is reserved for storing the nonvolatile information related to Redundancy, Configuration and Protection. The usage of reserved TestFlash block is detailed in *Table 386*.

Table 386. TestFlash block memory map

Address Offset	Description	Size	Access
UTEST (Base Address 0x0040_0000 to 0x0040_3FFF)			
0x0000	OTP area for Device Serial Number	8 bytes	R/P
0x0008	Reserved	4 bytes	R/P
0x000C	Test Mode Disable Seal	4 bytes	R/P
0x0010	Test Mode Disable Block Select 0	16 bytes	R/P
0x0020	Factory Erase Diary Location	16 bytes	R/P
0x0030	Test Mode Disable Block Select 1	16 bytes	R/P
0x0040	Customer Single Bit Correction area	32 bytes	R
0x0060	Customer Double Bit Detection area	32 bytes	R
0x0080	Customer Programmable Detection area	32 bytes	R/P
0x00A0	Reserved	96 bytes	R/P
0x00B4	Flash Firmware — Minor identifier (f)	1 byte	R
0x00B5	Flash Firmware — Major identifier (F)	1 byte	R
0x00B6	Flash parameters identifiers (P)	1 byte	R
0x00B7	Flash Firmware and parameters — unused	1 byte	R
0x0100	Test Mode Disable Override Passcode0	4 bytes	R/P
0x0104	Reserved	252 bytes	R/P
0x0200	Reserved	3584 bytes	R/P
0x1000	Customer OTP data	12288 bytes	R/P

Table 386. TestFlash block memory map (continued)

Address Offset	Description	Size	Access
BAF (Base Address 0x0040_4000 to 0x0040_7FFF)			
0x0000	User OTP – BAF Area	16 KB	R/P

The TestFlash block can be enabled by the BIU. When the Test space is enabled, all the operations are mapped to the Test block. The TestFlash supports RWW, and is grouped with the Code Flash in partition 0.

Erase of the TestFlash block is always locked in User mode.

Programming of any section of the TestFlash block is subject to similar restrictions as the array in terms of how ECC is calculated. Only one program is allowed per 64-bit ECC segment.

The TestFlash block contains two different user sections: UTEST (16 KB) and BAF (16 KB).

The UTEST nonvolatile memory section is OTP and erase is not allowed. User mode programming of the UTEST section is enabled only when MCR.PEAS is high. The UTEST section is divided into a *Flexible* area where content is managed at SoC level, and a *Reserved* area where addresses are pre-allocated for storing certain content. The UTEST section may be locked and unlocked against programming via LOCK0.TSLOCK in conjunction with the relative sideband (SoC specific). For locking purposes, UTEST is treated as a standalone block even though it resides in the TestFlash block. UTEST access (both Read and Modify) is disabled in Test mode once the test mode disable seal password is written. As UTEST is part of the TestFlash physical block, it is also protected.

The UTEST section is also set as OPP once the test mode disable seal password is written, thus programming is only allowed on virgin 64-bit locations.

The other area of the TestFlash block is BAF and may be used for defined functions (possibly to store boot code, other configuration words or factory process codes). This nonvolatile memory area is OTP and cannot be erased once programmed.

Although the BAF (like UTEST) is part of the TestFlash block, OPP functionality is managed through sideband (SoC specific). Program protection is also managed through sideband (SoC specific), in conjunction with LOCK0.LOWLOCK. For locking and OPP purposes, BAF is treated as a standalone block.

When the UTEST section (via the entire TestFlash block) is protected via test mode disable seal, the BAF is also protected from any high voltage operation which may alter the content. Access to BAF is read only for FPEC in the event that the test interface is open (Failure Analysis scenario).

32.3 Register memory maps and descriptions

In this section some nonvolatile registers are described as well as the usual volatile registers. Note that such entities are not Flip-Flops, but TestFlash block locations with a specific purpose.

During the Flash initialization phase, the FPEC reads these nonvolatile registers and updates their corresponding volatile registers. When the FPEC detects ECC double errors in these special locations, it behaves in the following way:

- In the case of failing system locations (configurations, redundancy, EmbAlgo firmware), the initialization phase is interrupted and a Fatal Error is flagged.
- In the case of failing user locations (protections, censorship, BIU, etc.), the volatile registers are filled with ‘1’s and the Flash Initialization ends setting MCR.PEG to low.

32.3.1 Register memory maps

This section presents the memory mapping of all the Flash memory registers.

32.3.1.1 User Register memory maps

The Flash user registers represent the communication interface between the host CPU and the FPEC.

Some register bits (command bits) are Read/Write for the CPU and read only for the FPEC. Other register bits (status bits) are Read/Write for the FPEC and read only for the CPU.

The Flash user registers mapping is shown in *Table 387*.

Table 387. Flash user registers memory map

Offset address	Description	Access(1)	Reset value(2)	Location
0x0000	Module Configuration Register (MCR)	R/W	0x0000_0600	*Figure 4*
0x0004	Reserved	—	—	—
0x0008	Extended Module Configuration Register (MCRE)	R	0x0502002A(3)	*Figure 294*
0x000C	Reserved	—	—	—
0x0010	Locking 0 register (LOCK0)	R/W	0xBFFF_FFFF	*Figure 295*
0x0014	Locking 1 register (LOCK1)	R/W	0x0000_FFFF	*Figure 296*
0x0018	Locking 2 register (LOCK2)	R/W	0x7FFF_FFFF	*Figure 297*
0x001C	Reserved	—	—	—
0x0020	Reserved	—	—	—
0x0024	Reserved	—	—	—
0x0028	Reserved	—	—	—
0x002C	Reserved	—	—	—
0x0030	Reserved	—	—	—
0x0034	Reserved	—	—	—
0x0038	Select 0 register (SEL0)	R/W	0x0000_0000	*Figure 298*
0x003C	Select 1 register (SEL1)	R/W	0x0000_0000	*Figure 299*
0x0040	Select 2 register (SEL2)	R/W	0x0000_0000	*Figure 300*
0x0044	Reserved	—	—	—
0x0048	Reserved	—	—	—

Table 387. Flash user registers memory map (continued)

Offset address	Description	Access(1)	Reset value(2)	Location
0x004C	Reserved	—	—	—
0x0050	Address Register (ADR)	R	0x0000_0000	*Figure 301*
0x0054	User Test register 0 (UT0)	R/W	0x0000_0001	*Figure 302*
0x0058	User Multiple Input Signature Register 0 (UM0)	R/W	0x0000_0000	*Figure 303*
0x005C	User Multiple Input Signature Register 1 (UM1)	R/W	0x0000_0000	*Figure 304*
0x0060	User Multiple Input Signature Register 2 (UM2)	R/W	0x0000_0000	*Figure 305*
0x0064	User Multiple Input Signature Register 3 (UM3)	R/W	0x0000_0000	*Figure 306*
0x0068	User Multiple Input Signature Register 4 (UM4)	R/W	0x0000_0000	*Figure 307*
0x006C	User Multiple Input Signature Register 5 (UM5)	R/W	0x0000_0000	*Figure 308*
0x0070	User Multiple Input Signature Register 6 (UM6)	R/W	0x0000_0000	*Figure 309*
0x0074	User Multiple Input Signature Register 7 (UM7)	R/W	0x0000_0000	*Figure 310*
0x0078	User Multiple Input Signature Register 8 (UM8)	R/W	0x0000_0000	*Figure 311*
0x007C	User Multiple Input Signature Register 9 (UM9)	R/W	0x0000_0000	*Figure 312*
0x0080	Over Program Protection 0 (OPP0)	R	0x0000_0000	*Figure 313*
0x0084	Over Program Protection 1 (OPP1)	R	0x0000_0000	*Figure 314*
0x0088	Over Program Protection 2 (OPP2)	R	0x0000_0000	*Figure 315*
0x008C	Reserved	—	—	—
0x0090	Test Mode Disable Password Check (TMD)	W	—	*Figure 316*

1. Note that R/W registers may contain some read only or write only bits.
2. Reset value comes from UTEST are within TestFlash, and this assumes the location is erased and all possible blocks are available.
3. Note that R/W register may contain some read or write only bits.

32.3.2 User register descriptions

This section consists of user register descriptions in address order.

32.3.2.1 Module Configuration Register (MCR)

The Module Configuration Register is used to enable and monitor all the Modify operations of the Flash memory module.

Address: 0x0000 Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	RRE	AEE	EEE	DWEE				0	0	0	0	0	0	0	0
W		w1c	w1c	w1c												
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	EER	RWE	SBC	0	PEAS	DONE	PEG	PECIE	FERS	0	0	PGM	PSUS	ERS	ESUS	EHV
W	w1c	w1c	w1c													
Reset	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0

Figure 293. Module Configuration Register (MCR)

Table 388. MCR field descriptions

Field	Description
1 RRE	Read Reference Error RRE provides information on previous reads monitoring the analog Read references. If the current and voltage Read references are out of range, this bit is set to signify that previous reads requested may have been corrupted. This status flag is cleared to ‘0’ by writing ‘1’ to the register location. A Write of ‘0’ has no effect. 0 Reads are occurring normally 1 A Read Reference Error occurred during a previous Read
2 AEE	Address Encode Error On every Read request to the Flash, the incoming address is compared to an encoded address (row, column, and block) from the memory array using the read data sense amp timing. If these two values do not match (zero selected, multiple selected, wrong selected), or the timing is incorrect, an address encode error is recorded. This status flag is cleared to ‘0’ by writing ‘1’ to the register location. A Write of ‘0’ has no effect. 0 Reads are occurring normally 1 An Address Encode Error occurred during a previous Read.
3 EEE	ECC after ECC Error EEE provides information on previous reads monitoring the ECC after ECC feature. On every Read request to the Flash, ECC parity is recomputed (encoding) just after ECC decoding, and if there is a mismatch between the ECC calculations (taking into account corrections or detections) a late error is reported. This status flag is cleared to ‘0’ by writing ‘1’ to the register location. A Write of ‘0’ has no effect. 0 Reads are occurring normally 1 A previous Read may have been corrupted.
4:7 DWEE	Double Word Uncorrectable Ecc Error DWEE provides indication about which Double Word has been involved in Ecc Uncorrectable Event Error. DWEE is a read only. Flags bits get cleared when EER is cleared.

Table 388. MCR field descriptions (continued)

Field	Description
16 EER	ECC Event Error EER provides information on previous reads. This bit must then be cleared or a reset must occur before this bit returns to a ‘0’ state. EER may not be set to ‘1’ by the user. EER is not set for a ECC Single Error detection and correction event. This status flag is cleared to ‘0’ by writing ‘1’ to the register location. A Write of ‘0’ has no effect. 0 Reads are occurring normally. 1 An ECC Double Error occurred during a previous Read.
17 RWE	Read-While-Write event Error RWE provides information on previous Reads during a Modify operation. Read-While-Write Error means that a Read access to the Flash Array partition has occurred while the FPEC was performing a Program or Erase operation or an Array Integrity Check on the same partition. This bit must then be cleared or a reset must occur before this bit returns to a ‘0’ state. This bit may not be set to ‘1’ by the user. This status flag is cleared to ‘0’ by writing ‘1’ to the register location. A Write of ‘0’ has no effect. 0 Reads are occurring normally. 1 A RWW Error occurred during a previous Read.
18 SBC	Single Bit Correction SBC provides information on previous reads. This bit must then be cleared or a reset must occur before this bit returns to a ‘0’ state. This bit may not be set to ‘1’ by the user. In the event of an ECC Double Error detection, this bit is not set. This status flag is cleared to ‘0’ by writing ‘1’ to the register location. A Write of ‘0’ has no effect. The function of this bit is SoC dependent and it can be disabled (through UT0.SBCE). 0 Reads are occurring normally. 1 An ECC Single Error occurred and was corrected during a previous Read.
20 PEAS	Program/Erase Access Space PEAS is used to indicate which space is valid for Program and Erase operations: main array space or UTEST section within Test Flash block. – PEAS is captured and held with the first Interlock Write done for Modify operations. – PEAS is retained between sampling events (such as subsequent to first Interlock Writes). – PEAS is changed during erase-suspended program (when program is in UTEST) and reverts back to its original state (related to erase-suspended operation) once the erase-suspended program is completed (at MCR.PGM 1-0 transition). 0 UTEST address space is disabled for program and main address space enabled. 1 UTEST address space is enabled for program and main address space disabled.

Table 388. MCR field descriptions (continued)

Field	Description
21 DONE	Modify operation DONE DONE indicates if the Flash memory module is performing a high voltage operation. DONE is set to ‘1’: – On termination of the Flash memory module reset. – At the end of program and Erase high voltage sequences. – After 1-0 transition of EHV, which aborts a high voltage Program/Erase operation (within t_{PABT} or t_{EABT}, equal to the P/E Abort Latency). – After 0-1 transition of ESUS/PSUS which suspends an Erase/Program operation (within t_{ESUS}/t_{PSUS} time equal to the Erase/Program Suspend Latency). – DONE is cleared to ‘0’: – just after 0-1 transition of EHV, which initiates a high voltage operation, or after resuming a suspended operation. 0 Flash is executing a high voltage operation. 1 Flash is not executing a high voltage operation.
22 PEG	Program/Erase Good PEG is automatically updated to show the completion status of the last Flash Program or Erase sequence involving high voltage operations. PEG is evaluated on the completion status of only the interlocked doubleword during page program or quad-page programming where some doubleword addresses are not interlocked (already programmed or intended to remain virgin). PEG is set to ‘0’ after Aborting a Program/Erase high voltage operation, indicating sequence failure. PEG is set to ‘1’ when the Flash memory module is reset, unless a Flash Initialization error has been detected. **Note:** PEG is valid only when PGM=1 and/or ERS=1 and after DONE 0-1 transitions due to an Abort or the completion of a Program/Erase operation. PEG remains valid until PGM/ERS 1-0 transitions or EHV 0-1 transitions. PEG is not valid after DONE 0-1 transitions when ESUS/PSUS is set to logic ‘1’. If Program or Erase is attempted on blocks that are locked, PEG=1 indicating that the operation was successful and the content of the block were properly protected from the Program or Erase operation. Blocks can be locked through LOCK0/1/2 registers OR with f90_plock0/1/2 sideband signals. If a Program operation tries to write ‘1’ to bits that are ‘0’, the program operation is correctly executed on the new bits to be programmed at ‘0’, but PEG is cleared indicating that the requested operation has failed. If programming during an Erase Suspend a location that was under suspended erase, program is not executed and PEG is set to ‘0’. With end2end ECC scheme, during an Interlock Write also ECC parity bits are input. After setting EHV, the FPEC checks the integrity of the data received in terms of ECC. If any doubleword in the doubleword program, in the page program or in the quad-page program, contains an inconsistency between data and parity, the whole operation is rejected and a PEG=0 is set. PEG behavior with blocks marked as OPP (though f90_otpen0/1/2 sideband signals) is the following: PEG=0 if the location to be programmed is not virgin, and thus would not be over programmed. In Array Integrity Check or Margin Read PEG is not altered when the operation is completed, regardless of the occurrence of any error, abort, break point or suspend. The presence of errors can only be detected by comparing the checksum value stored in UM0-9 and by checking the MCR flags. 0 Program, Erase operation failed or Program, Erase aborted. 1 Program or Erase operation successful.

Table 388. MCR field descriptions (continued)

Field	Description
23 PECIE	Program/Erase Complete Interrupt Enable PECIE provides a mechanism to trigger an interrupt request upon the assertion of the DONE flag. 0 An interrupt request is not generated when the DONE flag is set. 1 An interrupt request is generated when the DONE flag is set.
24 FERS	Factory Erase FERS is used for Erase and Program operations to enable a faster operation. Factory Erase must adhere to *Initial maximum (All Temperatures)* conditions, with respect to voltage conditions and Write/Erase cycling. The temperature can range from -40 °C to 150 °C. Factory erase does not support Read and RWW operations. – FERS may be set to enable a Factory Erase or Program after setting ERS or PGM but prior to setting EHV. – FERS can be cleared only when EHV is low and DONE is high. – FERS is cleared on reset **Note:** Factory Erase may also be disabled by writing a location in the UTEST nonvolatile memory space. Address 0x800020 in the UTEST nonvolatile memory space is checked at reset, and if programmed with data that contains at least one zero, a Factory Erase is not allowed, and FERS is locked. Writes attempted that violate the above cases result in FERS remaining cleared. 0 Factor Erase is disabled. 1 Factory Erase is enabled.
27 PGM	Program PGM is used to setup the Flash memory module for a Program operation. – 0-1 transition of PGM initiates a program sequence. – 1-0 transition of PGM ends the program sequence. PGM can only be set in the following conditions: – User Mode Read (ERS is low and UT0.UTE is low); – Erase Suspend (ERS and ESUS high) with EHV low. PGM can only be cleared by the user when PSUS and EHV are low and DONE is high. PGM is cleared on reset. **Note:** In an erase-suspended program, programming Flash memory locations in blocks which were being operated on in the Erase is forbidden. Command is not accepted and PEG is set to ‘0’ 0 Flash is not executing a program sequence. 1 Flash is executing a program sequence.

Table 388. MCR field descriptions (continued)

Field	Description
28 PSUS	Program Suspend PSUS is used to indicate that the Flash memory module is in Program Suspend or in the process of entering a Suspend state. The Flash memory module is in Program Suspend when PSUS=1 and DONE=1. – PSUS can be set high only when PGM and EHV are high. – 0-1 transition of PSUS starts the sequence which sets DONE and places the Flash in Program Suspend. The Flash memory module enters Suspend within t_{PSUS} of this transition. – PSUS can be cleared only when DONE and EHV are high. – 1-0 transition of PSUS with EHV=1 starts the sequence which clears DONE and returns the module to Program. The Flash memory module cannot exit Program Suspend and clear DONE while EHV is low. In the case of Program Suspend while in Erase Suspend (PSUS=1, ESUS=1), PSUS has to be cleared first in order to resume currently higher level suspended operation that is program. PSUS is cleared on reset. 0 Program sequence is not suspended. 1 Program sequence is suspended.
29 ERS	Erase ERS is used to setup the Flash memory module for an Erase operation. – 0-1 transition of ERS initiates an Erase sequence. – 1-0 transition of ERS ends the Erase sequence. ERS can be set only under User Mode Read (PGM is low and UT0.UTE is low). ERS can be cleared by the user only when ESUS and EHV are low and DONE is high. ERS is cleared on reset. 0 Flash is not executing an Erase sequence. 1 Flash is executing an Erase sequence.

Table 388. MCR field descriptions (continued)

Field	Description
30 ESUS	Erase Suspend ESUS is used to indicate that the Flash memory module is in Erase Suspend or in the process of entering a Suspend state. The Flash memory module is in Erase Suspend when ESUS = 1 and DONE=1. – ESUS can be set high only when ERS and EHV are high and PGM is low. – 0-1 transition of ESUS starts the sequence which sets DONE and places the Flash in Erase Suspend. The Flash memory module enters Suspend within t_{ESUS} of this transition. – ESUS can be cleared only when DONE and EHV are high and PGM is low. – 1-0 transition of ESUS with EHV=1 starts the sequence which clears DONE and returns the module to Erase. The Flash memory module cannot exit Erase Suspend and clear DONE while EHV is low. In case of Program Suspend while in Erase Suspend (PSUS=1, ESUS=1), ESUS cannot be cleared first. ESUS is cleared on reset. 0 Erase sequence is not suspended. 1 Erase sequence is suspended.
31 EHV	Enable High Voltage The EHV bit enables the Flash memory module for a high voltage Program/Erase operation. EHV is cleared on reset. EHV must be set after an Interlock Write to start a Program/Erase sequence. EHV may be set under one of the following conditions: – Erase (ERS=1, ESUS=0, UT0.AIE=0) – Program (ERS=0, ESUS=0, PGM=1, UT0.AIE=0) – Erase-suspended program (ERS=1, ESUS=1, PGM=1, PSUS=0, UT0.AIE=0). For a program operation to initiate while an Erase is suspended, EHV must be cleared while in Erase Suspend (after DONE transitions to 1) before setting PGM. In normal operation, a 1-0 transition of EHV with DONE high, PSUS low and ESUS low terminates the current Program/Erase high voltage operation. In an erase-suspended program operation, a 1-0 transition of EHV with DONE, ESUS, PGM and ERS high and PSUS low terminates the program that is the current high voltage operation. When an operation is aborted, there is a 1-0 transition of EHV with DONE low and the eventual Suspend bit for the current Program/Erase operation low. An Abort causes the value of PEG to be cleared, indicating failed Program/Erase. Address locations being operated on by the aborted operation contain indeterminate data after an Abort. EHV cannot be set to ‘1’ after an Abort request (EHV being cleared) until the Abort sequence is completed (DONE transitions to 1). Once the Abort sequence is completed (DONE=1) a new operation of the kind aborted can start provided that a new interlock is given and then EHV is set to ‘1’ again. A suspended operation cannot be aborted. Aborting a high voltage operation leaves the Flash memory module addresses in an indeterminate data state. This may be recovered by executing an Erase on the affected blocks. EHV may be written during Suspend. EHV must be high to exit Suspend, resuming previously suspended command. EHV cannot not be written during transition into Suspend state (after Suspend bit (ESUS or PSUS) is set high) until DONE transitions to ‘1’. EHV may not be written after ESUS is set and before DONE transitions high. EHV may not be cleared after ESUS is cleared and before DONE transitions low. 0 Flash is not enabled to perform a high voltage operation. 1 Flash is enabled to perform a high voltage operation.

A number of MCR bits are protected against Write when another bit, or set of bits, are in a specific state. These Write locks are covered on a bit-by-bit basis in the preceding description, but those locks do not consider the effects of trying to write two or more bits simultaneously.

The Flash memory module does not allow the user to write bits simultaneously (which puts the device into an illegal state). This is implemented through a priority mechanism among the bits shown in *Table 389*.

Table 389. MCR bits Set/Clear priority levels

Priority level	MCR bits
1	ERS
2	PGM
3	EHV
4	ESUS, PSUS

If the user attempts to write two or more MCR bits simultaneously then only the bit with the lowest priority level is written.

32.3.2.2 Extended Module Configuration Register (MCRE)

The Extended Module Configuration Register value is based on the size, blocks and address space configuration of the Flash memory module.

Address: 0x0008 Access: User Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	N8KH			N256K				N64KH			N32KH		N16KH		
W																
Reset	0	0	0	0	0	1	0	1	0	0	0	0	0	0	1	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	N64KM			N32KM		N16KM			N64KL			N32KL		N16KL		
W																
Reset	0	0	0	0	0	0	0	0	0	0	1	0	1	0	1	0

Figure 294. Extended Module Configuration Register (MCRE)

Table 390. MCRE field descriptions

Field	Description
1:2 N8KH	Number of 8 KB blocks in high space. 00 Zero 8 KB blocks. 01 Two 8 KB blocks. 10 Four 8 KB blocks. 11 Six 8 KB blocks.
3:7 N256K	Number of 256 KB blocks. The value of the N256K field is dependent upon the size of the embedded Flash memory. N256K is read only. N256K value represents in the number of blocks effectively available. 00000 Zero 256 KB blocks. 00001 One 256 KB block. 00010 Two 256 KB blocks. 00011 Three 256 KB blocks. 00100 Four 256 KB blocks. 00101 Five 256 KB blocks. 00110 Six 256 KB blocks. 00111 Seven 256 KB blocks 01000 Eight 256 KB blocks. 01001 Nine 256 KB blocks. 01010 Ten 256 KB blocks. … 11100 28 256 KB blocks. 11101 29 256 KB blocks. 11110 30 256 KB blocks. 11111 31 256 KB blocks.
8:10 N64KH	Number of 64 KB blocks in high space. N64KH is read only. 000 Zero 64 KB blocks. 001 Two 64 KB blocks. 010 Four 64 KB blocks. 011 Six 64 KB blocks. 100 Eight 64 KB blocks. 101 Reserved. 110 Reserved. 111 Reserved.
11:12 N32KH	Number of 32 KB blocks in high space. N32K is read only. 00 Zero 32 KB blocks. 01 Two 32 KB blocks. 10 Four 32 KB blocks. 11 Reserved.

Table 390. MCRE field descriptions (continued)

Field	Description
13:15 N16KH	Number of 16 KB blocks in high space. N16K is read only. 000 Zero 16 KB blocks. 001 Two 16 KB blocks. 010 Four 16 KB blocks. 011 Six 16 KB blocks. 100 Eight 16 KB blocks. 101 Reserved. 110 Reserved. 111 Reserved.
16:18 N64KM	Number of 64 KB blocks in mid space. N64KM is read only. 000 Zero 64 KB blocks. 001 Two 64 KB blocks. 010 Four 64 KB blocks. 011 Six 64 KB blocks. 100 Eight 64 KB blocks. 101 Reserved. 110 Reserved. 111 Reserved.
19:20 N32KM	Number of 32 KB blocks in mid space. N32KM is read only. 00 Zero 32 KB blocks. 01 Two 32 KB blocks. 10 Four 32 KB blocks. 11 Reserved.
21:23 N16KM	Number of 16 KB blocks in mid space. N16KM is read only. 000 Zero 16 KB blocks. 001 Two 16 KB blocks. 010 Four 16 KB blocks. 011 Six 16 KB blocks. 100 Eight 16 KB blocks. 101 Reserved. 110 Reserved. 111 Reserved.
24:26 N64KL	Number of 64 KB blocks in low space. N64KL is read only. 000 Zero 64 KB blocks. 001 Two 64 KB blocks. 010 Four 64 KB blocks. 011 Six 64 KB blocks. 100 Eight 64 KB blocks. 101 Reserved. 110 Reserved. 111 Reserved.

Table 390. MCRE field descriptions (continued)

Field	Description
27:28 N32KL	Number of 32 KB blocks in low space. N32KL is read only. 00 Zero 32 KB blocks. 01 Two 32 KB blocks. 10 Four 32 KB blocks. 11 Reserved.
29:31 N16KL	Number of 16 KB blocks in low space. N16KL is read only. 000 Zero 16 KB blocks. 001 Two 16 KB blocks. 010 Four 16 KB blocks. 011 Six 16 KB blocks. 100 Eight 16 KB blocks. 101 Reserved. 110 Reserved. 111 Reserved.

32.3.2.3 Lock 0 register (LOCK0)

The LOCK0 (Low/Mid Address Space Block Locking) register provides a means of protecting blocks from being modified. LOWLOCK and MIDLOCK bits corresponding to non existent blocks are read as ‘1’.

To determine the effective status of the Low/Mid Address Space, combine (Or) LOCK0 with sidebands:

- f90_plock0 – in order to evaluate protection against programming.
- f90_elock0 – in order to evaluate protection against erase.

The effect of programming or erasing a block protected with LOCK0 is that no modification occurs and PEG is set to ‘1’.

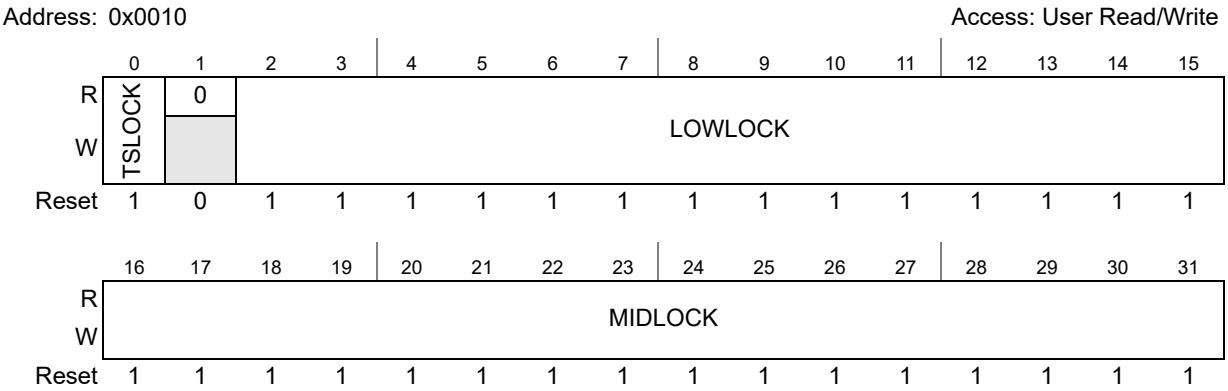


Figure 295. Lock 0 register (LOCK0)

Table 391. LOCK0 field descriptions

Field	Description
0 TSLOCK	UTEST NVM block lock This bit is used to lock the UTEST NVM from programming (UTEST NVM is not erasable). The LOCK field is not writable: – Once an Interlock Write is completed until MCR.DONE is set. – If a high voltage operation is suspended. The default value of the TSLOCK bit is locked. The effective status of the UTEST protection is done combining (ORed) LOCK0.TSLOCK with sidebands *f90_plock0.31* in order to evaluate protection against program. 0 Test Address Space Block is unlocked and can be modified. 1 Test Address Space Block is locked and cannot be modified.
2:15 LOWLOCK	Low address space block lock 13-0 These bits are used to lock the existing blocks of Low Address Space from Program and Erase. The block numbering for the Low Blocks starts at LOWLOCK[0] with16 KB block region (if available), then the 32 KB block region (if available), and lastly the 64 KB block region (if available). This continues until all blocks are accounted for. The LOCK field is not writable: – Once an Interlock Write is completed until MCR.DONE is set. – If a high voltage operation is suspended. The default value of the LOWLOCK bits is locked. In the event that blocks are not present (due to configuration or total memory size), the LOWLOCK bits default to locked and are not writable. The reset value is always ‘1’ and register writes have no effect. 0 Low Address Space Block is unlocked and can be modified. 1 Low Address Space Block is locked and cannot be modified.
16:31 MIDLOCK	Mid address space block lock 15-0 These bits are used to lock the blocks of Mid Address Space from Program and Erase. The block numbering for the Mid Blocks starts at MIDLOCK[0] with16 KB block region (if available), then the 32 KB block region (if available), and lastly the 64 KB block region (if available). This continues until all blocks are accounted for. The LOCK field is not writable: – Once an Interlock Write is completed until MCR.DONE is set. – If a high voltage operation is suspended. The default value of the MIDLOCK bits is locked. In the event that blocks are not present (due to configuration or total memory size), the MIDLOCK bits default to locked, and are not writable. The reset value is always ‘1’ and register writes have no effect. 0 Mid Address Space Block is unlocked and can be modified. 1 Mid Address Space Block is locked and cannot be modified.

32.3.2.4 Lock 1 register (LOCK1)

The LOCK1 (High/Data Address Space Block Locking) register provides a means of protecting blocks from being modified. HIGHLOCK bits corresponding to nonexistent blocks on are read as ‘1’.

- To determine the effective status of the High Address Space, combine (Or) LOCK1 with sidebandsf90_plock1 – in order to evaluate protection against program
- f90_elock1 – in order to evaluate protection against erase.

The effect of programming/erasing a block protected with LOCK1 is that no modification occurs and PEG is set to ‘1’.

Address: 0x0014 Access: User Read/Write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	HIGHLOCK															
W																
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Figure 296. Lock 1 register (LOCK1)

Table 392. LOCK1 field descriptions

Field	Description
16:31 HIGHLOCK	High address space block lock 15-0 These bits are used to lock the blocks of High Address Space from Program and Erase. The block numbering for the High Blocks starts at HIGHLOCK[0] with16 KB block region (if available), then the 32 KB block region (if available), and lastly the 64 KB block region (if available). This continues until all blocks are accounted for. The LOCK field is not writable: – Once an Interlock Write is completed until MCR.DONE is set. – If a high voltage operation is suspended. The default value of the HIGHLOCK bits is locked. When blocks are not present (due to configuration or total memory size), the HIGHLOCK bits default to locked, and are not writable. The reset value is always ‘1’ and register writes have no effect. 0 High Address Space Block is unlocked and can be modified. 1 High Address Space Block is locked and cannot be modified.

32.3.2.5 Lock 2 Register (LOCK2)

The LOCK2 (256K Address Space Block Locking) register provides a means of protecting blocks from being modified. The A256KLOCK bits corresponding to nonexistent blocks are read as ‘1’.

To determine the effective status of the A256KLOCK Address Space, combine (Or) LOCK2 with sidebands:

- f90_plock2 – in order to evaluate protection against program
- f90_elock2 – in order to evaluate protection against erase.

The effect of programming/erasing a block protected with LOCK2 is that no modification occurs and PEG is set to ‘1’.

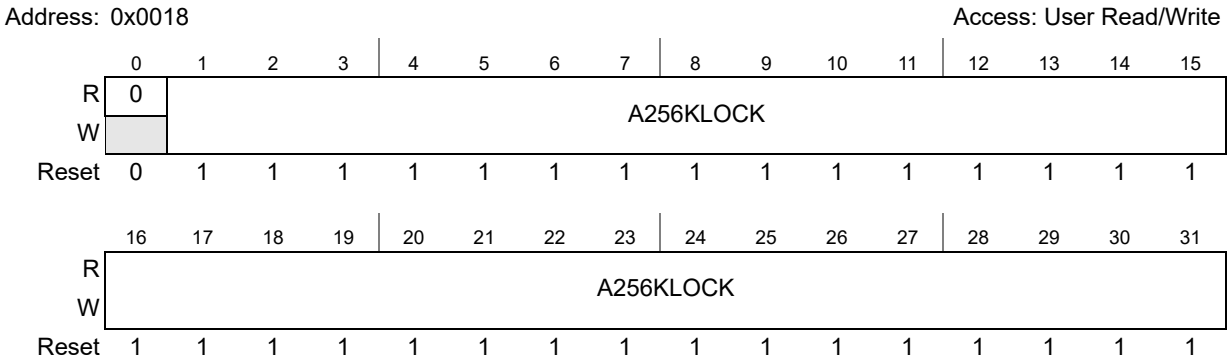


Figure 297. Lock 2 Register (LOCK2)

Table 393. LOCK2 field descriptions

Field	Description
1:31 A256KLOCK	256K address space block lock 30-0 These bits are used to lock the existing blocks of Low Address Space from Program and Erase. The block numbering for the 256 KB blocks starts with A256KLOCK[0] and continues until all blocks are accounted for. The LOCK field is not writable: – Once an Interlock Write is completed until MCR.DONE is set. – If a high voltage operation is suspended. The default value of the A256KLOCK bits is locked. When blocks are not present (due to configuration or total memory size), the A256KLOCK bits default to locked and are not writable. The reset value is always ‘1’ and register writes have no effect. 0 A256K Address Space Block is unlocked and can be modified. 1 A256K Address Space Block is locked and cannot be modified.

32.3.2.6 Select 0 register (SEL0)

The SEL0 (Low/Mid Address Space Block Select 0) register provides a means to select blocks to be operated on during Erase or Array Integrity Check – Margin Read.

Starting an Erase operation, SEL0 bits are latched and become non-alterable from the first interlock cycle that follows the rising transition of MCR.ERS (see step 4 in *Section 32.4.5.5: Erase*) until the completion of the erase high voltage operation (1–0 transition of MCR.EHV with MCR.ERS=1 and with MCR.ESUS=0). Reading returns latched data (image of the SEL0 at the start of the erase operation) while writing has no effect.

Address: 0x0038 Access: User Read/Write

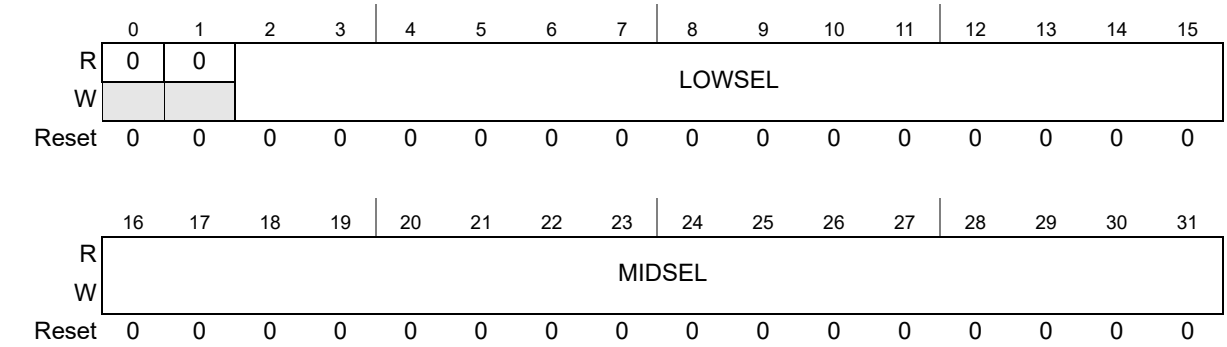


Figure 298. Select 0 register (SEL0)

Table 394. SEL0 field descriptions

Field	Description
2:15 LOWSEL	Low address space block Select 13-0 The block numbering for the Low Blocks starts at LOWSEL[0] with16 KB block region (if available), then the 32 KB block region (if available), and lastly the 64 KB block region (if available). This continues until all blocks are accounted. The blocks must be selected (or unselected) before doing an Erase Interlock Write as part of the Erase sequence. The select field is not writable: – once an Interlock Write is completed until DONE is set – a high voltage operation is suspended. The select field must not be written when UT0[AIE] is high. When blocks are not present (due to configuration or total memory size) the corresponding LOWSEL bits default to unselected and are not writable. The reset value is always ‘0’, and register writes have no effect. 0 Low Address Space Block is unselected for Erase. 1 Low Address Space Block is selected for Erase.
16:31 MIDSEL	Mid address space block Select 15-0 The block numbering for the Mid Blocks starts at MIDLSEL[0] with16 KB block region (if available), then the 32 KB block region (if available), and lastly the 64 KB block region (if available). This continues until all blocks are accounted for. The blocks must be selected (or unselected) before doing an Erase Interlock Write as part of the Erase sequence. The select field is not writable: – once an Interlock Write is completed until DONE is set – a high voltage operation is suspended. The select field must not be written when UT0[AIE] is high. When blocks are not present (due to configuration or total memory size), the corresponding MIDSEL bits default to unselected and are not writable.The reset value is always ‘0’, and register writes have no effect. 0 Mid Address Space Block is unselected for Erase. 1 Mid Address Space Block is selected for Erase.

32.3.2.7 Select 1 register (SEL1)

The SEL1 (High Address Space Block Select Register 1) provides a means of selecting blocks to be operated on during Erase or Array Integrity Check – Margin Read.

Starting an Erase operation, SEL1 bits are latched and become non-alterable from the first interlock cycle that follows the rising transition of MCR.ERS (see step 4 in *Section 32.4.5.5: Erase*) until the completion of the erase high voltage operation (1–0 transition of MCR.EHV with MCR.ERS=1 and with MCR.ESUS=0). Reading returns latched data (image of the SEL1 at the start of the erase operation) while writing has no effect.

Address: 0x003C Access: User Read/Write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	HIGHSEL															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 299. Select 1 register (SEL1)

Table 395. SEL1 field descriptions

Field	Description
16:31 HIGHSEL	High/Data address space block Select 1 15-0 AThe block numbering for the High Blocks starts at HIGHSEL[0] with16 KB block region (if available), then the 32 KB block region (if available), and lastly the 64 KB block region (if available). This continues until all blocks are accounted for. The blocks must be selected (or unselected) before doing an Erase Interlock Write as part of the Erase sequence. The select field is not writable: – once an Interlock Write is completed until DONE is set – a high voltage operation is suspended. The select field must not be written when UT0[AIE] is high. When blocks are not present (due to configuration or total memory size) the corresponding HIGHSEL bits default to unselected and are not writable. The reset value is always ‘0’ and register writes have no effect. 0 High Address Space Block is unselected for Erase. 1 High Address Space Block is selected for Erase.

32.3.2.8 Select 2 register (SEL2)

The SEL2 (256K Address Space Block Select 2) register provides a means of selecting blocks to be operated on during Erase or Array Integrity Check – Margin Read.

Starting an Erase operation, SEL2 bits are latched and become non-alterable from the first interlock cycle that follows the rising transition of MCR.ERS (see step 4 in *Section 32.4.5.5: Erase*) until the completion of the erase high voltage operation (1–0 transition of MCR.EHV

with MCR.ERS=1 and with MCR.ESUS=0). Reading returns latched data (image of the SEL2 at the start of the erase operation) while writing has no effect.

Address: 0x0040 Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	A256KSEL														
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	A256KSEL															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 300. Select 2 register (SEL2)

Table 396. SEL2 field descriptions

Field	Description
1:31 A256KSEL	256K address space block select 30-0 The block numbering for the 256 KB blocks starts with A256KSEL[0] and continues until all blocks are accounted for. The blocks must be selected (or unselected) before doing an Erase Interlock Write as part of the Erase sequence. The select field is not writable: – once an Interlock Write is completed until DONE is set – a high voltage operation is suspended. The select field must not be written when UT0[AIE] is high. When blocks are not present (due to configuration or total memory size) the corresponding A256KSEL bits default to unselected and are not writable. The reset value is always ‘0’, and register writes have no effect. 0 A256K Address Space Block is unselected for Erase. 1 A256K Address Space Block is selected for Erase.

32.3.2.9 Address register (ADR)

The address register provides the first failing address in the event of module failures (ECC, RWW or FPEC) or the first address at which ECC single error correction occurs. ADR captures the first failing address for the modes listed in *Table 397* according to the priority rules in *Table 398*.

Table 397. ADR update mode list

Mode	Update on error	Note
Read	ECC Double Error, ECC Single Error (if UT0.SBCE=1)	—
Modify	FPEC error	—

Table 397. ADR update mode list

Mode	Update on error	Note
RWW	ECC Double Error, RWW error, FPEC error, ECC Single Error (if UT0.SBCE=1)	—
User Test	ECC Double Error, ECC Single Error (if UT0.SBCE=1)	With Break Point enabled (UT0.AIBPE=1), ADR requires MCR.ERR and MCR.SBC to be cleared

In case of Array Integrity Check or Margin Read with Break Point enabled (UT0.AIBPE=1), the ADR value matches the address where the Break Point was triggered only when the current run (first start or re-start after a breakpoint (NAIBP cleared)) starts with MCR.ERR and MCR.SBC flags cleared, otherwise the ADR priority rule overrides Break Point.

Table 398. ADR content: priority list

Priority level	Error flag	ADR content
1	MCR.EER = 1	Address of first ECC Double Error
2	MCR.RWE = 1	Address of first RWW Error
3	MCR.PEG = 0	Address of first FPEC Error
4	MCR.SBC = 1	Address of first ECC Single Error Correction

Address: 0x0050 Access: User Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	SAD	0	0	0	0	0	0	0	ADDR							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	ADDR													0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 301. Address register (ADR)

Table 399. ADR field descriptions

Field	Description
0 SAD	Test Address When this bit is high, the address indicated by ADDR[23:3] belongs to the Test Block.
8:28 ADDR	Address 23-3 The Address Register provides the first failing address in the event of ECC error (MCR.EER set) or the first failing address in the event of RWW error (MCR.RWE set), or the address of a failure that may have occurred in a FPEC operation (MCR.PEG cleared). The Address Register also provides the first address at which a ECC single error correction occurs (MCR.EDC set) if the SoC is configured to show this feature through UT0.SBCE. The ECC double error detection takes the highest priority, followed by the RWW error, the FPEC error and the ECC single error correction. When accessed, ADR provides the address related to the first event occurred with the highest priority. This address is always a doubleword address that selects 64 bits. In case of a simultaneous ECC Double Error Detection on more than one doubleword of the same page, bits ADDR[4] and ADDR[3] output '0'. The same is valid for a simultaneous ECC Single Error Correction on more than one doubleword of the same page. In User mode the Address Register is read only. Address given is in Flash format and has to be intended as in the Flash memory Map: offset respect to absolute address (See *Table 400: Flash memory partition 0 addresses mapping*, *Table 401: Flash memory Partition 1 addresses mapping*).

Table 400. Flash memory partition 0 addresses mapping

Block	Flash format addresses	Absolute addresses
B0BF	0x0080_4000 to 0x0080_7FFF	0x0040_4000 to 0x0040_7FFF
B0F1	0x0000_0000 to 0x0000_3FFF	0x00FC_0000 to 0x00FC_3FFF
B0F2	0x0000_4000 to 0x0000_7FFF	0x00FC_4000 to 0x00FC_7FFF
B0F3	0x0000_8000 to 0x0000_BFFF	0x00FC_8000 to 0x00FC_BFFF
B0F4	0x0000_C000 to 0x0000_FFFF	0x00FC_C000 to 0x00FC_FFFF
B0F5	0x0001_0000 to 0x0001_7FFF	0x00FD_0000 to 0x00FD_7FFF
B0F6	0x0001_8000 to 0x0001_FFFF	0x00FD_8000 to 0x00FD_FFFF
B0F7	0x0002_0000 to 0x0002_FFFF	0x00FE_0000 to 0x00FE_FFFF
B0F8	0x0003_0000 to 0x0003_FFFF	0x00FF_0000 to 0x00FF_FFFF
B0F9	0x0004_0000 to 0x0007_FFFF	0x0100_0000 to 0x0103_FFFF
B0F10	0x0008_0000 to 0x000B_FFFF	0x0104_0000 to 0x0107_FFFF
B0F11	0x000C_0000 to 0x000F_FFFF	0x0108_0000 to 0x010B_FFFF
B0F12	0x0010_0000 to 0x0013_FFFF	0x010C_0000 to 0x010F_FFFF
B0F13	0x0014_0000 to 0x0017_FFFF	0x0110_0000 to 0x0113_FFFF
B0UT	0x0080_0000 to 0x0080_3FFF	0x0040_0000 to 0x0040_3FFF
B0TF	0x0080_8000 to 0x0080_BFFF	0x0040_8000 to 0x0040_BFFF

Table 401. Flash memory Partition 1 addresses mapping

Block	Flash format addresses	Absolute addresses
B1F0	0x0018_0000 to 0x0018_3FFF	0x0080_0000 to 0x0080_3FFF
B1F1	0x0018_4000 to 0x0018_7FFF	0x0080_4000 to 0x0080_7FFF
B1F2	0x0018_8000 to 0x0018_BFFF	0x0080_8000 to 0x0080_BFFF
B1F3	0x0018_C000 to 0x0018_FFFF	0x0080_C000 to 0x0080_FFFF
—	—	—

32.3.2.10 User Test 0 register (UT0)

The User Test 0 register feature gives the user of the Flash memory module the ability to perform test features on the Flash.

The User Test 0 register allows control of the way the Flash content check is performed. Bits MRE, MRV and AIS of the User Test 0 register are not accessible when MCR.DONE or UT0.AID are low. Reading returns indeterminate data while writing has no effect.

Address: 0x0054 Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	UTE	SBCE	0	0	0	0	0	0	0	0	0	0	0	CPR	CPA	CPE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	NAIBP	AIBPE	0	AISUS	MRE	MRV	0	AIS	AIE	AID
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Figure 302. User Test 0 register (UT0)

Table 402. UT0 field descriptions

Field	Description
0 UTE	User Test Enable This status bit gives indication when User Test is enabled. All bits in UT0 and UM0–UM9 are locked when this bit is ‘0’. This bit is not writable to a ‘1’, but may be cleared. The reset value is ‘0’. To set UTE, write the password 0xF9F99999 to the UT0 register. The UTE password is only accepted if PGM=0 and ERS=0 (Program and Erase are not being requested). UTE can only be cleared if AID=1, MRE=0 and AIE=0. While clearing UTE, writes to set AIE or MRE are ignored.
1 SBCE	Single Bit Correction Enable This bit, when high, enables to flag the information about any eventual ECC Single Bit Correction in the Flash array (MCR.SBC and ADR). This is true in read mode and in Array Integrity Check. SBCE is also used as an Enable for interrupt signals created by the embedded Flash memory. ECC corrections that occur when SBCE is cleared are not logged.
13 CPR	Customer Programmable Read Voltage and Reference Detection This bit is used to control the error generation for the Customer Programmable Detection area in the UTEST block. If this bit is set and the Customer Programmable Detection location in UTEST is read, a Read Voltage and Reference Error detection is set. 0 Customer Programmable Read Voltage and Reference Detection Error is not enabled 1 Customer Programmable Read Voltage and Reference Detection Error is enabled.
14 CPA	Customer Programmable Address Encode Detection This bit is used to control the error generation for the Customer Programmable Detection area in the UTEST block. If this bit is set, and the Customer Programmable Detection location in UTEST is read an Address Encode Error detection is set. 0 Customer Programmable Address Encode Detection Error is not enabled 1 Customer Programmable Address Encode Detection Error is enabled
15 CPE	Customer Programmable EDC after ECC Detection This bit is used to control the error generation for the Customer Programmable Detection area in the UTEST block. If this bit is set, and the Customer Programmable Detection location in UTEST is read a EDC after ECC Error detection is set. 0 Customer Programmable EDC after ECC Detection Error is not enabled 1 Customer Programmable EDC after ECC Detection Error is enabled
22 NAIBP	Next Array Integrity Break Point If AIBPE is set, NAIBP is set once a single bit correction (if enabled) or double bit detection is noted during the Array Integrity test. NAIBP is not writable to ‘1’ but may be cleared to ‘0’. Clearing NAIBP to ‘0’ enables the Array Integrity operation to be re-started after a breakpoint is encountered. If the Array Integrity operation completes without encountering another correction or detection, AID is set with NAIBP remaining ‘0’. 0 Array Integrity is not currently at a break point. 1 Array Integrity is at a break point.
23 AIBPE	Array Integrity Break Point Enable. To enable breakpoints during an Array Integrity test, AIBPE may be set. When breakpoints are enabled, if the run has been started with MCR.ERR (and MCR.SBC when UT0.SBCE=1) flag cleared, on the eventual occurrence of a breakpoint condition, MCR flags and ADR value may be updated. See NAIBP description for more information about Array Integrity breakpoints. 0 Array Integrity breakpoints are not enabled. 1 Array Integrity breakpoints are enabled during Array Integrity Checks.

Table 402. UT0 field descriptions (continued)

Field	Description
25 AISUS	Array Integrity Suspend. AISUS toggles Suspend of an Array Integrity - Margin Read operation. AISUS can be written to ‘1’ only when AID is low AISUS can be written to ‘0’ only when AID is high. Attempting to Write AISUS and AIE on the same clock cycle results in only AIE being written. 0 Array Integrity sequence not suspended. 1 Array Integrity sequence is suspended.
26 MRE	Margin Read Enable MRE enables Margin Reads to be done. MRE combined with MRV enables User Margin Reads to be done. Margin Reads run on a block-by-block basis in linear address sequence selected by SEL0, SEL1 and SEL2. This bit is not accessible when MCR.DONE or UT0.AID are low or if AISUS or NAIBP are high: reading returns indeterminate data while writing has no effect. 0 Margin Read is not enabled. 1 Margin Read is enabled.
27 MRV	Margin Read Value If MRE is high, MRV selects the margin level that is being checked. Margin can be checked to an erased level (MRV=1) or to a programmed level (MRV=0). This bit is not accessible whenever MCR.DONE or UT0.AID are low or if AISUS or NAIBP are high: reading returns indeterminate data while writing has no effect. 0 Zero’s (programmed) Margin Reads are requested (if MRE=1). 1 One’s (erased) Margin Reads are requested (if MRE=1).
29 AIS	Array Integrity Sequence AIS determines the address sequence to be used during Array Integrity checks. The default AIS=0 checks the read propagation paths along proprietary sequences followed by user code. The sequential run sequence time is significantly shorter than the proprietary sequence. AIS is not writable if AIE is high. This bit is not accessible whenever MCR.DONE or UT0.AID are low: reading returns indeterminate data while writing has no effect. 0 Array Integrity sequence is proprietary sequence. 1 Array Integrity sequence is sequential.

Table 402. UT0 field descriptions (continued)

Field	Description
30 AIE	Array Integrity Enable Setting AIE to ‘1’ starts the Array Integrity Check and the Margin Read on all selected blocks. The sequence is selected by AIS and the MISR (UM0–UM9) is checked on completion to determine if a correct signature is obtained. AIE can be set only if MCR.ERS, MCR.PGM and MCR.EHV are all low. AIE is not simultaneously writable to ‘1’ while UTE is being cleared to ‘0’. Clearing AIE terminates the check when an Array Integrity operation finishes (AID=1) and aborts the check when and Array Integrity operation is ongoing (AID=0). AIE remains ‘1’ when the Array Integrity Check - Margin Read are suspended or are in breakpoint and cannot be cleared. (AIE can be cleared if NAIBP=0 and AISU=0) 0 Array Integrity Check and Margin Read are not enabled. 1 Array Integrity Check and Margin Read are enabled.
31 AID	Array Integrity Done AID is set to indicate that the Array Integrity Check - Margin Read is complete and the MISR (UM0–9) can be checked. AID may also assert if breakpoints are enabled (AIBPE is set) and the breakpoint condition is met. AID may also assert if Suspend is set (AISUS 0–1). AID can not be written, and is status only. 0 Array Integrity Check is ongoing. 1 Array Integrity Check is done.

32.3.2.11 User Multiple Input Signature registers

The Multiple Input Signature registers allow evaluation of the Array Integrity or Margin Read run on a block-by-block basis by accumulating signatures from data bit fields.

These registers are not accessible whenever MCR.DONE or UT0.AID are low. Reading returns indeterminate data while writing has no effect.

The registers can be written in order to seed signatures before starting an AIC or MR operation.

Note: Do not write to the MISR registers during an Array Integrity operation (including Suspend or Breakpoint) as it may alter the final signature in an unpredictable way.

If reads extend into an invalid address location during address sequencing (such as greater than the maximum address for a given array size), reads are still executed to the array but the results are not deterministic: MISR registers are not recalculated and the previous value is retained.

32.3.2.11.1 User Multiple Input Signature 0 register (UM0)

UM0 represents bits 31–0 of the whole 288-bit word (4 doublewords including ECC).

Address: 0x0058 Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	MISR31	MISR30	MISR29	MISR28	MISR27	MISR26	MISR25	MISR24	MISR23	MISR22	MISR21	MISR20	MISR19	MISR18	MISR17	MISR16
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	MISR15	MISR14	MISR13	MISR12	MISR11	MISR10	MISR9	MISR8	MISR7	MISR6	MISR5	MISR4	MISR3	MISR2	MISR1	MISR0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 303. User Multiple Input Signature 0 register (UM0)

Table 403. UM0 field descriptions

Field	Description
0:31 MISR[31:0]	Multiple input signature 31-0. MISR registers accumulate a signature from an array integrity event. The MISR captures all data fields, as well as ECC fields and the ECC error signal. Data (read and ECC) in the MISR represents corrected data (if required) captured after ECC logic is applied.

32.3.2.11.2 User Multiple Input Signature 1 register (UM1)

UM1 represents bits 63–32 of the whole 288-bit word (4 doublewords including ECC).

Address: 0x005C Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	MISR63	MISR62	MISR61	MISR60	MISR59	MISR58	MISR57	MISR56	MISR55	MISR54	MISR53	MISR52	MISR51	MISR50	MISR49	MISR48
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	MISR47	MISR46	MISR45	MISR44	MISR43	MISR42	MISR41	MISR40	MISR39	MISR38	MISR37	MISR36	MISR35	MISR34	MISR33	MISR32
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 304. User Multiple Input Signature 1 register (UM1)

Table 404. UM1 field descriptions

Field	Description
0:31 MISR[63:32]	Multiple input signature 63-32. MISR registers accumulate a signature from an array integrity event. The MISR captures all data fields, as well as ECC fields and the ECC error signal. Data (read and ECC) in the MISR represents corrected data (if required) captured after ECC logic is applied.

32.3.2.11.3 User Multiple Input Signature 2 register (UM2)

UM2 represents bits 95–64 of the whole 288-bit word (4 doublewords including ECC).

Address: 0x0060 Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	MISR95	MISR94	MISR93	MISR92	MISR91	MISR90	MISR89	MISR88	MISR87	MISR86	MISR85	MISR84	MISR83	MISR82	MISR81	MISR80
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	MISR79	MISR78	MISR77	MISR76	MISR75	MISR74	MISR73	MISR72	MISR71	MISR70	MISR69	MISR68	MISR67	MISR66	MISR65	MISR64
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 305. User Multiple Input Signature 2 register (UM2)

Table 405. UM2 field descriptions

Field	Description
0:31 MISR[95:64]	Multiple input signature 95-64. MISR registers accumulate a signature from an array integrity event. The MISR captures all data fields, as well as ECC fields and the ECC error signal. Data (read and ECC) in the MISR represents corrected data (if required) captured after ECC logic is applied.

32.3.2.11.4 User Multiple Input Signature 3 register (UM3)

UM3 represents bits 127–96 of the whole 288-bit word (4 doublewords including ECC).

Address: 0x0064 Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	MISR127	MISR126	MISR125	MISR124	MISR123	MISR122	MISR121	MISR120	MISR119	MISR118	MISR117	MISR116	MISR115	MISR114	MISR113	MISR112
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	MISR111	MISR110	MISR109	MISR108	MISR107	MISR106	MISR105	MISR104	MISR103	MISR102	MISR101	MISR100	MISR99	MISR98	MISR97	MISR96
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 306. User Multiple Input Signature 3 register (UM3)

Table 406. UM3 field descriptions

Field	Description
0:31 MISR[127:96]	Multiple input signature 127-96. MISR registers accumulate a signature from an array integrity event. The MISR captures all data fields, as well as ECC fields and the ECC error signal. Data (read and ECC) in the MISR represents corrected data (if required) captured after ECC logic is applied.

32.3.2.11.5 User Multiple Input Signature 4 register (UM4)

UM4 represents bits 159–128 of the whole 288-bit word (4 doublewords including ECC).

Address: 0x0068 Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	MISR159	MISR158	MISR157	MISR156	MISR155	MISR154	MISR153	MISR152	MISR151	MISR150	MISR149	MISR148	MISR147	MISR146	MISR145	MISR144
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	MISR143	MISR142	MISR141	MISR140	MISR139	MISR138	MISR137	MISR136	MISR135	MISR134	MISR133	MISR132	MISR131	MISR130	MISR129	MISR128
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 307. User Multiple Input Signature 4 register (UM4)

Table 407. UM4 field descriptions

Field	Description
0:31 MISR[159:128]	Multiple input signature 159-128. MISR registers accumulate a signature from an array integrity event. The MISR captures all data fields, as well as ECC fields and the ECC error signal. Data (read and ECC) in the MISR represents corrected data (if required) captured after ECC logic is applied.

32.3.2.11.6 User Multiple Input Signature 5 register (UM5)

UM5 represents bits 191–160 of the whole 288-bit word (4 doublewords including ECC).

Address: 0x006C Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	MISR191	MISR190	MISR189	MISR188	MISR187	MISR186	MISR185	MISR184	MISR183	MISR182	MISR181	MISR180	MISR179	MISR178	MISR177	MISR176
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	MISR175	MISR174	MISR173	MISR172	MISR171	MISR170	MISR169	MISR168	MISR167	MISR166	MISR165	MISR164	MISR163	MISR162	MISR161	MISR160
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 308. User Multiple Input Signature 5 register (UM5)

Table 408. UM5 field descriptions

Field	Description
0:31 MISR[191:160]	Multiple input signature 191–160. MISR registers accumulate a signature from an array integrity event. The MISR captures all data fields, as well as ECC fields and the ECC error signal. Data (read and ECC) in the MISR represents corrected data (if required) captured after ECC logic is applied.

32.3.2.11.7 User Multiple Input Signature 6 register (UM6)

UM6 represents bits 223–192 of the whole 288 bits word (4 doublewords including ECC).

Address: 0x0070 Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	MISR223	MISR222	MISR221	MISR220	MISR219	MISR218	MISR217	MISR216	MISR215	MISR214	MISR213	MISR212	MISR211	MISR210	MISR209	MISR208
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	MISR207	MISR206	MISR205	MISR204	MISR203	MISR202	MISR201	MISR200	MISR199	MISR198	MISR197	MISR196	MISR195	MISR194	MISR193	MISR192
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 309. User Multiple Input Signature 6 register (UM6)

Table 409. UM6 field descriptions

Field	Description
0:31 MISR[223:192]	Multiple input signature 223–192. MISR registers accumulate a signature from an array integrity event. The MISR captures all data fields, as well as ECC fields and the ECC error signal. Data (read and ECC) in the MISR represents corrected data (if required) captured after ECC logic is applied.

32.3.2.11.8 User Multiple Input Signature 7 register (UM7)

UM7 represents bits 255–224 of the whole 288 bits word (4 doublewords including ECC).

Address: 0x0074 Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	MISR255	MISR254	MISR253	MISR252	MISR251	MISR250	MISR249	MISR248	MISR247	MISR246	MISR245	MISR244	MISR243	MISR242	MISR241	MISR240
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	MISR239	MISR238	MISR237	MISR236	MISR235	MISR234	MISR233	MISR232	MISR231	MISR230	MISR229	MISR228	MISR227	MISR226	MISR225	MISR224
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 310. User Multiple Input Signature 7 register (UM7)

Table 410. UM7 field descriptions

Field	Description
0:31 MISR[255:224]	Multiple input signature 255–224. MISR registers accumulate a signature from an array integrity event. The MISR captures all data fields, as well as ECC fields and the ECC error signal. Data (read and ECC) in the MISR represents corrected data (if required) captured after ECC logic is applied.

32.3.2.11.9 User Multiple Input Signature 8 register (UM8)

UM8 represents the ECC parity bits of the whole 288-bits word (8 parity bits for each of the 4 doublewords).

Address: 0x0078 Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	MISR287	MISR286	MISR285	MISR284	MISR283	MISR282	MISR281	MISR280	MISR279	MISR278	MISR277	MISR276	MISR275	MISR274	MISR273	MISR272
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	MISR271	MISR270	MISR269	MISR268	MISR267	MISR266	MISR265	MISR264	MISR263	MISR262	MISR261	MISR260	MISR259	MISR258	MISR257	MISR256
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 311. User Multiple Input Signature 8 register (UM8)

Table 411. UM8 field descriptions

Field	Description
0:31 MISR[287:256]	Multiple input signature 287-256 MISR registers accumulate a signature from an array integrity event. The MISR captures all data fields, as well as ECC fields and the ECC error signal. Data (read and ECC) in the MISR represents corrected data (if required) captured after ECC logic is applied.

32.3.2.11.10 User Multiple Input Signature 9 register (UM9)

UM9 represents the ERR bits.

Address: 0x007C Access: User Read/Write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	MISR288
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 312. User Multiple Input Signature 9 register (UM9)

Table 412. UM9 field descriptions

Field	Description
31 MISR288	Multiple input signature 288 MISR registers accumulate a signature from an array integrity event. The MISR captures all data fields, as well as ECC fields and the ECC error signal. Data (read and ECC) in the MISR represents corrected data (if required) captured after ECC logic is applied.

32.3.2.12 Over Program Protection registers

Over program protection prevents blocks from being over-programmed. These registers show the over-program protection status. See *Section 32.4.5.3: Over programming protection (OPP) enable*.

32.3.2.12.1 Over Program Protection 0 register (OPP0)

The Low/Mid address space is covered by the Over Program Protection 0 (OPP0) register.

Address: 0x0080 Access: User Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	LOWOPP													
W																
Reset(1)	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	MIDOPP															
W																
Reset(1)	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—

1. A reset value of "—" indicates that the value is set at the factory and can vary among different applications

Figure 313. Over Program Protection 0 register (OPP0)

Table 413. OPP0 field descriptions

Field	Description
2:15 LOWOPP	Low address space Over Program Protection 13–0 The block numbering for the Low Blocks starts at LOWOPP[0] with 16 KB block region (if available), then the 32 KB block region (if available), and lastly the 64 KB block region (if available). This continues until all blocks are accounted for. The LOWOPP field is not writable, and is status only. The default value of the LOWOPP bits is not over-programming protected. When blocks are not present (due to configuration or total memory size), the LOWOPP bits default to not over-programming protected and will remain 0. 0 Low Address Space Block is unprotected from over-programming. 1 Low Address Space Block is protected from over-programming.
16:31 MIDOPP	Mid address space Over Program Protection 15-0 The block numbering for the Mid Blocks starts at MIDOPP[0] with 16 KB block region (if available), then the 32 KB block region (if available), and lastly the 64 KB block region (if available). This continues until all blocks are accounted for. The MIDOPP field is not writable, and is status only. The default value of the MIDOPP bits is not over-programming protected. When blocks are not present (due to configuration or total memory size), the MIDOPP bits default to not over-programming protected, and will remain 0. 0 Mid Address Space Block is unprotected from over-programming. 1 Mid Address Space Block is protected from over-programming.

32.3.2.12.2 Over Program Protection 1 register (OPP1)

The High address space is covered by the Over Program Protection 1 (OPP1) register.

Address: 0x0084 Access: User Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset(1)	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	HIGHOPP															
W																
Reset(1)	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—

1. A reset value of "—" indicates that the value is set at the factory and can vary among different applications

Figure 314. Over Program Protection 1 register (OPP1)

Table 414. OPP1 field descriptions

Field	Description
16:31 HIGHOPP	High address space Over Program Protection 15-0 The block numbering for the High Blocks starts at HIGHOPP[0] with 16 KB block region (if available), then the 32 KB block region (if available), and lastly the 64 KB block region (if available). This continues until all blocks are accounted for. The HIGHOPP field is not writable, and is status only. The default value of the HIGHOPP bits is not over-programming protected. When blocks are not present (due to configuration or total memory size), the HIGHOPP bits default to not over-programming protected and will remain 0. 0 High Address Space Block is unprotected from over-programming. 1 High Address Space Block is protected from over-programming.

32.3.2.12.3 Over Program Protection 2 register (OPP2)

The 256K high address space is covered by the Over Program Protection 2 (OPP2) register.

Address: 0x0088 Access: User Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	A256KOPP														
W																
Reset(1)	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	A256KOPP															
W																
Reset(1)	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—

1. A reset value of "—" indicates that the value is set at the factory and can vary among different applications

Figure 315. Over Program Protection 2 register (OPP2)

Table 415. OPP2 field descriptions

Field	Description
1:31 A256KOPP	256K High address space Over Program Protection 1-31 The block numbering for the 256 KB Blocks starts at A256KOPP[0] and continues until all blocks are accounted for. The A256KOPP field is not writable, and is status only. The default value of the A256KOPP bits is not over-programming protected. When blocks are not present (due to configuration or total memory size), the A256KOPP bits default to not over-programming protected and will remain 0. 0 High Address Space Block is unprotected from over-programming. 1 High Address Space Block is protected from over-programming.

32.3.2.13 Test Mode Disable Password Check register (TMD)

The Test Mode Disable Password Check (TDM) register provides a means of entering the password to be matched with the one stored in UTEST location in order to allow manufacturer access to protected Test Mode blocks by writing 0x5A4B_3C2D in the Test Mode Disable Seal location.

Address: 0x0090 Access: User Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W	PWD															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0															
W	PWD															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 316. Test Mode Disable Password Check register (TMD)

Table 416. TDM field descriptions

Field	Description
0:31 PWD	The Test Mode Disable Check register is used as a location to provide a password challenge to unlock blocks selected to be sealed as part of the Test Mode Disable seal mechanism. Please see *Section 32.4.6.2: Test mode disable* for more information. Writes to the register have no effect except for a password challenge. Reads to this register will always return 0. The register's reset value is 0. **Note:** PWD bitfield is byte-reversed compared to UTEST TMD password : for instance, if UTEST TMD password is written 0x01020304, the PWD bitfield must be written 0x04030201

32.4 Functional description

This chapter describes the details of all the modes of operation available to the Flash memory module.

32.4.1 Reset

A reset is the highest priority operation for the Flash memory module and terminates all other operations.

The Flash memory module uses reset to initialize register and status bits to their default reset values. If the Flash memory module is executing a Program or Erase operation (MCR.PGM=1 or MCR.ERS=1) and a reset is issued, the operation is immediately terminated and the module disables the high voltage logic without damage to the high voltage circuits. Reset terminates all operations and forces the Flash memory module into Read mode ready to receive Read/Write accesses.

Reset and power-off must not be used as a systematic way to terminate a Program or Erase operation.

After reset is negated read register access may be performed, although registers that require updating from shadow information or other inputs may not read updated values until MCR.DONE transforms. MCR.DONE may be polled to determine if the Flash memory module has transitioned out of reset. Note that the registers cannot be written until MCR.DONE is high.

32.4.2 Power-down mode (Disable mode)

The power-down (or Disable) mode allows the turning-off of all Flash DC current sources. All power dissipation in this mode is due solely to leakage.

In power-down mode no Read from or Write to the Flash array is possible. Every Flash array Read access executed when the power-down mode is active outputs invalid data.

The user may not read some registers (UM0–UM9 and part of UT0) until the power-down mode is exited. On the contrary, write access is locked on all the registers in power-down mode.

When enabled the Flash memory module returns to its pre-disable state in all cases unless in the process of executing an Erase or a Program high voltage operation at the time of disable. If the Flash memory module is disabled during an Erase operation, the operation is

suspended and MCR.ESUS bits are set to ‘1’. The user may resume the Erase operation at the time the module is enabled by clearing MCR.ESUS bit. MCR.EHV must be high to resume the Erase operation.

If the Flash memory module is disabled during a Program operation, the operation is suspended and MCR.PSUS bit is set to ‘1’. The user may resume the Program operation at the time the module is enabled by clearing MCR.PSUS bit. MCR.EHV must be high to resume the Erase operation.

If the Flash memory module is disabled during an erase-suspended program operation, the program operation is suspended and both ESUS and PSUS are set to a ‘1’. When the module is enabled PSUS must be cleared first to resume the program. Erase may be resumed after the program ends. EHV must be high to resume both operations.

If the Flash memory module is disabled during AIC or MR, the operation is aborted and the MISR value is not deterministic.

If the Flash memory module is put in power-down mode and the vector table remains mapped in the Flash address space, the user must be mindful that the Flash memory module significantly increases the interrupt response time by adding several Wait states.

Note: Do not enter low-power mode when the power-down mode is active.

32.4.3 Low-power mode (Sleep mode)

The low-power (or Sleep) mode turns off most of the DC current sources within the Flash memory module. Wake-up time from low-power mode is faster than wake-up time from power-down mode.

In low-power mode no Read from or Write to the Flash array is possible. Every Read access executed when the low-power mode is active outputs invalid data.

The user may not read some registers (UM0–UM9 and part of UT0) until the low-power mode is exited. Write access is locked on all the registers in low-power mode.

When exiting from low-power mode the Flash memory module returns to its pre-sleep state in all cases unless in the process of executing an Erase high voltage operation at the time of entering sleep mode. If the Flash memory module is put in sleep mode during an Erase operation, MCR.ESUS is set to ‘1’. The user may resume the Erase operation at the time the module exits from sleep by clearing MCR.ESUS. MCR.EHV must be high to resume the Erase operation.

If the Flash memory module is put in low-power mode during a Program operation, the operation is suspended and MCR.PSUS is set to ‘1’. The user may resume the Program operation at the time the module is enabled by clearing MCR.PSUS. MCR.EHV must be high to resume the Erase operation.

If the Flash memory module is put in low-power mode during an erase-suspended program operation, the program operation is suspended and both ESUS and PSUS are set to a ‘1’. When the module is enabled, PSUS must first be cleared for the program to resume. The Erase may be resumed after the program ends. EHV must be high to resume both operations.

If the Flash memory module is disabled during AIC or MR, the operation is aborted and the MISR value is not deterministic.

Note: Do not enter power-down mode when the low-power mode is active.

32.4.4 Read mode

Read mode is the active mode of operation of the Flash memory module in which the Flash array can be read or ‘written’ (the so-called Interlock Write), or the Flash registers can be read or written.

The Flash memory module enters Read mode when exiting from reset. The default state of the Flash memory module is Read mode.

The main spaces (Low, Mid, High, and 256K) and Test address space of Flash array can only be read in Read mode.

Flash registers can even be read when the module is in power-down or low-power mode (except a few documented registers) as well as in Read mode. However, Flash registers can only be written while in Read mode.

Flash memory module is in Read mode, granting the same access time, under the following conditions:

- The Read state is active when the module is enabled (user mode Read)
- The Read state is active when MCR.ERS and MCR.ESUS are high and MCR.PGM is low (Erase Suspend)
- The Read state is active when MCR.PGM and MCR.PSUS are high (Program Suspend)
- The Read state is active when MCR.PGM or MCR.ERS are high and high voltage operation is ongoing on one partition, thus MCR.EHV is high (condition referred as Read-While-Write). In this case Reads done on the other partitions are allowed, while Reads done to the partition being operated on (either Erase or Program) result in an error with MCR.RWE set and the data read back possibly being random including the possibility of ECC errors.

Code Flash partitions Reads return 256 (1 page = 4 doublewords).
Data Flash partition Reads return 128 bits.
Register Reads return 32 bits (1 word).

Flash array Reads are performed through the Bus Interface Unit. In many cases the BIU applies “Read Page Buffering” to allow sequential reads to be executed with higher performance. This can provide data coherency issues that must be handled by software. Data coherency may be an issue after a Program or an Erase operation, as well as Test block operations.

Flash array Reads to invalid locations result in indeterminate data. Invalid locations occur when addressing points to blocks that are not included in the Flash memory map of any partition. Interlock Writes to invalid locations result in an interlock occurring, but attempts to program these blocks do not eventuate since they are forced into a locked state. Erase occurs to selected and unlocked blocks even if the Interlock Write is to an invalid location.

Register Reads to unmapped register address spaces return all 0’s. Registers Writes to unmapped register address spaces have no effect.

Simultaneous Read cycles on the Flash array and Read/Write cycles on the registers are possible, but not simultaneous register Read/Write accesses with Flash Array Interlock Writes are forbidden.

32.4.4.1 Error correction code

The Flash memory module implements a strategy for improving the reliability of the data stored in Flash via the use of Error Correction Code. The word size is fixed to 64 bits.

For each doubleword of 64 bits, 8 ECC bits are associated and programmed in such a way as to guarantee Single Error Correction and Double Error Detection (SEC-DED). ECC coding has all 0's (intended as data and parity) as an invalid word and all 1's as a valid word.

In addition to providing the requested Read data, the Flash also provides output sidebands reflecting the status of the single bit corrections and double bit detections performed at the end of each Read.

For functional safety coverage, Flash performs an EDC post ECC operation at every Read to detect mismatches in the ECC logic detection and correction. The result is output as sideband. Also active is a monitor circuit that checks the reference current in the sense amplifiers, including control of the internally regulated Read voltage, to prevent anomalous Read operation.

ECC circuitry provides single-bit fault correction used to achieve automotive reliability targets. Some units experience single-bit corrections throughout the life of the product with no impact on product reliability.

32.4.4.1.1 EEPROM emulation

The chosen ECC algorithm allows some bit manipulation so that a doubleword can be rewritten several times without needing an erase of the block. This allows the use of a doubleword to store flags useful for the EEPROM emulation.

As an example, the chosen ECC algorithm allows starting from an all 1's doubleword value and rewriting whichever of its four 16-bit halfwords to all 0's content by keeping the same ECC value.

Table 417 shows a set of doublewords sharing the same ECC value:

Table 417. Bits Manipulation: doublewords with the same ECC value

Doubleword	ECC parity bits
0xFFFF_FFFF_FFFF_FFFF	0xFF
0xFFFF_FFFF_FFFF_0000	0xFF
0xFFFF_FFFF_0000_FFFF	0xFF
0xFFFF_0000_FFFF_FFFF	0xFF
0x0000_FFFF_FFFF_FFFF	0xFF
0xFFFF_FFFF_0000_0000	0xFF
0xFFFF_0000_FFFF_0000	0xFF
0x0000_FFFF_FFFF_0000	0xFF
0xFFFF_0000_0000_FFFF	0xFF
0x0000_FFFF_0000_FFFF	0xFF
0x0000_0000_FFFF_FFFF	0xFF

Table 417. Bits Manipulation: doublewords with the same ECC value (continued)

Doubleword	ECC parity bits
0xFFFF_0000_0000_0000	0xFF
0x0000_FFFF_0000_0000	0xFF
0x0000_0000_0000_0000	0xFF

Note: When some Flash blocks are used for EEPROM emulation, reserve at least three blocks for this purpose.

32.4.4.2 Flash address generation check

For functional safety coverage, Flash provides output sidebands reflecting the re-encoded address used to perform the actual Read operation. In the re-encoding process both wordline and column within the selected block are evaluated. For the completion of the re-encoding operation, the addresses related to block selections are also computed. Re-encoding circuitry is able to detect both no-selection as well as multi-selection. Flash provides a self-check of address consistency comparing at each Read the sampling of the current Read address with the re-encoded one, and in case of mismatch an error sideband is set.

32.4.5 Modify mode

All the Modify operations of the Flash memory module are managed through the Flash User Registers interface.

When a Modify operation is active on some blocks, no Read access is possible on any other block of the same Flash partition. The Read-While-Modify is supported only among different Flash partitions.

During a Flash Modify operation any attempt to read any Flash location in the same Flash partition outputs invalid data and MCR.RWE is automatically set.

If a reset occurs during a Modify operation, the operation is immediately terminated and the module is reset to Read mode. The data integrity of the Flash section where the Modify operation was terminated is not guaranteed: the interrupted Flash Modify operation must be repeated.

In general each Modify operation is started through a sequence of 3 steps:

1. The first instruction is used to select the desired operation by setting its corresponding selection bit in MCR.PGM or MCR.ERS or UT0.MRE.
2. The second step is the definition of the operands: the address and the data for programming, or the blocks for Array Integrity Check or Margin Read.
3. The third instruction is used to start the Modify operation by setting MCR.EHV or UT0.AIE.

Once selected, but not yet initiated, one operation can be canceled by resetting the operation selection bit.

A summary of the available Flash Modify operations are shown in *Table 418*.

Table 418. Flash Modify operations

Operation	Select bit	Operands	Start bit
Doubleword Program	MCR.PGM	Address and Data by Interlock Writes	MCR.EHV
Block Erase	MCR.ERS	SEL0, SEL1 and SEL2	MCR.EHV
Array Integrity Check	None	SEL0, SEL1 and SEL2	UT0.AIE
Margin Read	UT0.MRE	UT0.MRV + SEL0, SEL1 and SEL2	UT0.AIE

Once MCR.EHV (or UT0.AIE) is set, all the operands can no longer be modified until MCR.DONE (or UT0.AID) is high.

In general each Modify operation is completed through a sequence of 4 steps:

1. Wait for operation completion: wait for MCR.DONE (or UT0.AID) to go high.
2. Check operation result: check bit MCR.PEG (or compare UM0–UM9 with expected value).
3. Switch-off FPEC by resetting MCR.EHV (or UT0.AIE).
4. Deselect current operation by clearing MCR.PGM/ERS (or UT0.MRE).

Note: Do not initiate a Modify operation on a Flash memory module while another Modify operation is in progress on a separate module.

In the following sections, all the possible Modify operations are described with accompanying examples of the sequences needed to activate them.

32.4.5.1 Program

A Flash Program sequence operates on any doubleword within the Flash core. Up to two words within the doubleword may be altered in a single doubleword Program operation. A single Program operation can also alter one page (4 doublewords, 32 bytes), or 4 pages max (128 bytes in total). It is possible to program only certain doublewords in the page or the quad-page, leaving the others with their original content. Whenever the array is programmed, the ECC bits are also programmed.

ECC is handled on a 64-bit boundary. Whenever you program, ECC bits are also programmed. Thus, if only one word in any given 64-bit ECC segment is programmed, the adjoining word (in that segment) should not be programmed, since ECC calculation has already completed for that 64-bit segment. Attempts to program the adjoining word will probably result in an operation failure. All programming operations should be from 64 bits to 1024 bits, and should be 64-bit aligned. The programming operation should completely fill selected ECC segments within the doubleword. Over programming of a 64-bit ECC segment may be done only in case of EEPROM emulation byte zeroing.

Due to the fact that Flash is supposed to function in an SoC with an end2end ECC scheme, ECC parity bit are also input during an Interlock Write. After setting MCR.EHV, the FPEC checks the integrity of the data received in terms of ECC. If any doubleword in the doubleword program, the page program or the quad-page program contains an inconsistency between data and parity, the whole operation is rejected and a MCR.PEG=0 is set. The doublewords without data and with parity inconsistency are thus not programmed.

Programming changes the value stored in an array bit from logic ‘1’ to logic ‘0’ only. Programming cannot change a stored logic ‘0’ to a logic ‘1’. If a logic ‘0’ is attempted to be

over programmed by a logic ‘1’, the resulting operation fails (MCR.PEG=0), and the 0s that are interlocked are merged (ORed) with 0s that are already present in the 64-bit ECC segment. Extreme care has to be taken when doing such an operation due to the high risk of producing ECC errors.

Addresses in locked blocks cannot be programmed. Register locked blocks cannot be programmed. Hardware locking is also available which only affects the Program operation. Hardware locking is enabled by sideband signals, and if enabled, programs are prevented from accessing hardware locked blocks in the same way as program software locking.

The user may program the values in any or all of eight the words in a page with a single program sequence. Page-bound words have addresses which differ only in address bits [4:2]. Up to four pages can be programmed at once on a quad-page boundary, which differ only in address bits [6:5]. Only certain doublewords may be programmed within the page or the quad-page, without altering the content of the ones unselected via interlock.

The Program operation consists of the following sequence of events:

1. Change the value of MCR.PGM from ‘0’ to ‘1’.
2. Ensure the block that contains the address to be programmed is unlocked. Write the first address to be programmed with the program data. The Flash memory module latches address bits (23:7) at this time, as well as written data. This Write is referred to as a Program Data Interlock Write. An Interlock Write may be as large as 64 bits, and as small as 32 bits (depending on the CPU bus).
3. If more than 1 word is to be programmed, write the additional address in the doubleword with data to be programmed. This is referred to as a Program Data Write. The Flash memory modules ignore address bits (23:7) for Program Data Writes. The eventual unwritten data word defaults to 0xFFFFFFFF.
4. Write a logic ‘1’ to the MCR.EHV to start the internal program sequence or terminate via step 9.
5. Wait until the MCR.DONE goes high.
6. Confirm MCR.PEG=1.
7. Write a logic ‘0’ to the MCR.EHV.
8. If more addresses are to be programmed, return to step 2.
9. Write a logic ‘0’ to the MCR.PGM to terminate the Program operation.

Programs may be initiated with the 0–1 transition of MCR.PGM or by clearing MCR.EHV at the end of a previous program. The first Write after a program is initiated determines the page address to be programmed. This first Write is referred to as an Interlock Write. The Interlock Write determines if the UTEST or normal array space will be programmed via MCR.PEAS.

In the case of an erase-suspended program, the values in MCR.PEAS may be modified via the Program Interlock Write, enabling erase-suspended programs to UTEST nonvolatile memory space and revert back to their original state (related to erase-suspended operation) once the erase-suspended program is completed.

An Interlock Write must be performed before setting MCR.EHV. An Interlock Write performed after setting MCR.EHV is forbidden and data and address passed through interlock is lost.The user may terminate a program sequence by clearing MCR.PGM prior to setting MCR.EHV.

After the Interlock Write, additional writes affect the data to be programmed at the word location determined by address bits [4:2], as well as the page location within a 1024 bit

segment (determined by address bits [6:5]). Unwritten locations default to a data value of 0xFFFF_FFFF. If multiple writes are performed on the same location, the data for the last Write is used in programming.

While MCR.DONE is low and MCR.EHV is high, the user may clear EHV, resulting in a Program Abort.

A Program Abort forces the module to step 8 of the program sequence.

An aborted program results in MCR.PEG being set low, indicating a failed operation. MCR.DONE must be checked to know when the aborting command has completed.

The data space being operated on before the Abort contains indeterminate data. This may be recovered by repeating the same program instruction or executing an Erase of the affected blocks.

The user may not abort a program sequence while in Program Suspend.

Figure 317. Doubleword program of data 0x55AA_55AA at address 0x00_AAA8 and data 0xAA55_AA55 at address 0x00_AAAC

```
MCR = 0x0000_0010;           /* Set PGM in MCR: Select Operation */
(0x00_AAA8) = 0x55AA_55AA;   /* Latch Address and 32 LSB data */
(0x00_AAAC) = 0xAA55_AA55;   /* Latch 32 MSB data */
MCR = 0x0000_0011;           /* Set EHV in MCR: Operation Start */
do                           /* Loop to wait for DONE=1 */
{ tmp = MCR;                 /* Read MCR */
} while ( !(tmp & 0x0000_0400) );
status = MCR & 0x0000_0200;  /* Check PEG flag */
MCR = 0x0000_0010;           /* Reset EHV in MCR: Operation End */
MCR = 0x0000_0000;           /* Reset PGM: Deselect Operation */
```

32.4.5.2 Program Suspend/Resume

The program sequence may be suspended to allow Read access to the Flash array. It is not possible to program or to erase during a Program Suspend. Read-While-Write operations may also be used to read the array during a program sequence providing the Read is to a different partition.

During Program Suspend, all reads to the doubleword targeted for program return indeterminate data.

A Program Suspend can be initiated by changing the value of MCR.PSUS from ‘0’ to ‘1’. MCR.PSUS can be set to ‘1’ at any time when MCR.PGM and MCR.EHV are high and MCR.ERS is low. 0–1 transition of MCR.PSUS causes the module to start the sequence which places it in Program Suspend.

It is possible to suspend a program performed during an Erase Suspend. The user must wait until MCR.DONE=1 before the module is suspended and further actions are attempted. MCR.DONE takes no more time than t_{PSUS} to go high after MCR.PSUS is set to ‘1’.

Once suspended, the array may be read. Flash memory reads while MCR.PSUS=1 from the doubleword being programmed return indeterminate data.

Figure 318. Doubleword Program Suspend

```
MCR = 0x0000_0019; /* Set PSUS in MCR: Program Suspend */
do                 /* Loop to wait for DONE=1 */
{ tmp = MCR;       /* Read MCR */
} while ( !(tmp & 0x0000_0400) );
```

Note that there is no need to clear MCR.EHV and MCR.PGM in order to perform reads during Program Suspend.

The program sequence is resumed by writing a logic ‘0’ to MCR.PSUS.

MCR.EHV must be set to ‘1’ before MCR.PSUS can be cleared to resume the operation.

The module continues the program sequence from one of a set of predefined points. This may extend the time required for the program operation.

Figure 319. Doubleword Program Resume

```
MCR = 0x0000_0011;/* Reset PSUS in MCR: Program Resume */
```

Repeated Suspends at a high frequency may result in the operation timing out, and the embedded Flash memory responds by completing the operation with a fail code (MCR.PEG=0). The minimum time between Program Suspends to ensure this does not occur is t_{PSRT}.

32.4.5.3 Over programming protection (OPP) enable

Over programming protection can be enabled on a block-by-block basis. In an OPP enabled block, any doubleword which has already been programmed cannot be programmed again. The OPP enable does not affect the Erase operation. It is possible to independently set OPP and (hardware) erase protection in order to avoid any kind of overwrite. Using these protection mechanisms in conjunction with the Test mode disable feature gives even greater overwrite protection. Attempts to over-program result in MCR.PEG being cleared.

OPP can be enabled for 16 KB, 32 KB, 64 KB and 256 KB blocks.

OPP is enabled by sideband signals and, if enabled, the Program operation is modified to check for programmed bits prior to undertaking a high voltage program event.

32.4.5.4 Successful program address

To enable the SoC to create logic to authenticate operations with a program (like in a diary operation), the Flash memory module provides the last successful address programmed as a sideband signal. This value is held until the next successful program. A successful program is defined as a program operation of data that is not all 1s, to an unlocked block and the operation results in MCR.PEG returning a ‘1’. An all 1’s value is the default (and invalid) address after reset.

In case of page programming, the last successful address returned is the beginning of the page itself regardless of whether that specific address has not been passed through interlock. In case of quad-page programming, the last successful address returned is the beginning of the quad-page itself regardless of whether that specific address has not been passed through interlock.

32.4.5.5 Erase

Erase changes the value stored in all bits of the selected block(s) to logic ‘1’. An Erase sequence operates on any combination of blocks in the low, mid, high or 256K address space. The Test block cannot be erased.

The Erase sequence is fully automated within the Flash. The user only needs to select the blocks to be erased and initiate the Erase sequence.

Register locked blocks cannot be erased. Hardware locking, which only affects the Erase operation is also available. Hardware locking is enabled by sideband signals and, if enabled, erases are prohibited on hardware locked blocks in the same way as erase software locking. If multiple blocks are selected for erase during an Erase sequence, no specific operation order can be assumed.

The Erase operation consists of the following sequence of events:

1. Change the value in the MCR.ERS bit from ‘0’ to ‘1’.
2. Select the block(s) to be erased by writing 1’s to the appropriate register(s) in SEL0, SEL1 and SEL2 registers. Note that Lock and Select are independent. If a block is selected and locked, no erase occurs.
3. Write to any address in Flash. This is referred to as an Erase Interlock Write.
4. Write a logic ‘1’ to MCR.EHV to start the internal Erase sequence or skip to step 9 to terminate.
5. Wait until MCR.DONE goes high.
6. Confirm MCR.PEG=1.
7. Write a logic ‘0’ to MCR.EHV.
8. If more blocks are to be erased, return to step 2.
9. Write a logic ‘0’ to the MCR.ERS bit to terminate the Erase operation.

After setting MCR.ERS, one Write (referred to as an Interlock Write) must be performed before MCR.EHV can be set to ‘1’.

Data words written during Erase sequence Interlock Writes are ignored. As Flash is supposed to function in an SoC with an end2end ECC scheme, ECC parity bits are also input during an Interlock Write. After setting MCR.EHV, the FPEC checks the integrity of the data received in terms of ECC even if the value of the data is not meaningful for the Erase operation. If an inconsistency between data and parity is found, the whole operation is rejected and MCR.PEG=0 is set.

The user may terminate the Erase sequence by clearing ERS before setting EHV.

An Erase operation may be aborted by clearing MCR.EHV assuming MCR.DONE is low, MCR.EHV is high and MCR.ESUS is low. An Erase Abort forces the module to step 8 of the Erase sequence. An aborted Erase results in MCR.PEG being set low, indicating a failed operation. MCR.DONE must be checked to know when the aborting command has completed.

The block(s) being operated on before the Abort contain indeterminate data. This may be recovered by executing an Erase on the affected blocks. The user may not abort an Erase sequence while in Erase Suspend.

Figure 320. Erase of blocks B0F1 and B0F2

```
MCR = 0x0000_0004;            /* Set ERS in MCR: Select Operation */
SEL0 = 0x0006_0000;           /* Set LSL2-1 in SEL0: Select Blocks */
(0x00_0000) = 0xFFFF_FFFF;    /* Latch a Flash Address */
MCR = 0x0000_0005;            /* Set EHV in MCR: Operation Start */
do                            /* Loop to wait for DONE=1 */
{ tmp = MCR;                  /* Read MCR */
} while ( !(tmp & 0x0000_0400) );
status = MCR & 0x0000_0200;   /* Check PEG flag */
MCR = 0x0000_0004;            /* Reset EHV in MCR: Operation End */
MCR = 0x0000_0000;            /* Reset ERS: Deselect Operation */
```

32.4.5.5.1 Erase Suspend/Resume

The Erase sequence may be suspended to allow Read access to the Flash array. It is possible to program another block in any Flash memory module during Erase Suspend. Programming a location subject to suspended Erase results in the program not being executed and PEG being set to 0.

It is not possible to erase any block during an Erase Suspend.

During Erase Suspend, all Reads to blocks targeted for erase return indeterminate data.

An Erase Suspend can be initiated by changing the value of MCR.ESUS from ‘0’ to ‘1’. MCR.ESUS can be set to 1 at any time when MCR.ERS and MCR.EHV are high and MCR.PGM is low. A 0–1 transition of MCR.ESUS causes the module to start the sequence which places it in Erase Suspend. The user must wait until MCR.DONE=1 before the module is suspended and further actions are attempted.

MCR.DONE takes no longer than t_{ESUS} to go high after MCR.ESUS is set to ‘1’.

Once suspended, it is not possible to alter the content of SEL0, SEL1 and SEL2 registers. Once suspended, the array may be read. Flash memory Reads while MCR.ESUS=1 from the block(s) being erased return indeterminate data.

Figure 321. Block Erase Suspend

```
MCR = 0x0000_0007; /* Set ESUS in MCR: Erase Suspend */
do                 /* Loop to wait for DONE=1 */
{ tmp = MCR;       /* Read MCR */
} while ( !(tmp & 0x0000_0400) );
```

Note that there is no need to clear MCR.EHV and MCR.ERS in order to perform Reads during Erase Suspend.

The Erase sequence is resumed by writing a logic ‘0’ to MCR.ESUS.

MCR.EHV must be set to ‘1’ before MCR.ESUS can be cleared to resume the operation. The module continues the Erase sequence from one of a set of predefined points and this may extend the time required for the Erase operation.

Figure 322. Block Erase Resume

```
MCR= 0x0000_0005;/* Reset ESUS in MCR: Erase Resume */
```

Repeated Suspends at a high frequency may result in the operation timing out and the embedded Flash memory responds by completing the operation with a fail code (MCR.PEG=0). The minimum time between Erase Suspends to ensure this does not occur is t_{ESRT}.

32.4.5.5.2 Factory Erase

The Factory Erase feature can be used only once through a diary location mechanism. If location UTEST at offset +0x0020 was virgin at the moment of the reset, MCR.FERS can be set, after rising MCR.ERS and before MCR.EHV. After setting MCR.EHV the FPEC recognizes the value of FERS and checks if the diary location has been programmed by the user (having changed from being virgin at reset). Under those conditions a Factory Erase can commence and no Read-While-Write is allowed (all the f90_busy sideband signals are set high).

32.4.5.6 User Test mode

Customers can put the Flash memory module in User Test mode to perform specific tests for the integrity of the Flash array.

Two kinds of test can be performed:

- Array Integrity Self Check
- Margin Read

User Test mode is an exclusive operation: it cannot be run in conjunction with any other Flash mode (Read, Write, low-power and power-down), and Read-While-Write functionality is inapplicable. Read accesses attempted by the user during User Test mode generates a Read-While-Write error (RWE of MCR set).

It is not permitted to perform User Test operations on the Test blocks (including the UTEST section).

32.4.5.6.1 Array Integrity Self Check

Array Integrity is checked using a predefined address sequence (proprietary), and this operation is executed on selected (SEL0, SEL1 and SEL2) blocks. Blocks marked by TMDBS are automatically excluded from MISR signature computation. Any random or non-random code is valid. Once the operation is completed, the results of the reads can be checked by reading the MISR value (stored in UM0–UM9), to determine if an incorrect Read, or ECC detection was noted. Array Integrity requires that the Read Wait States and Address Pipelined control registers in the BIU be set to match the frequency being used.

The internal MISR calculator is a 64 + 8 + 1-bit register.

The 256-bit data, the respective ECC parity data, and the double ECC (ERR) errors of the four doublewords are therefore captured by the MISR through 4 different Read accesses at the same location.

The whole check is done through 4 complete scans of the memory address space:

1. The first pass scans only bits 63-0 + 8 ecc + ERR of each doubleword.
2. The second pass scans only bits 128-64 + 8 ecc + ERR of each doubleword.
3. The third pass scans only bits 191-129 + 8 ecc + ERR of each doubleword.
4. The fourth pass scans only bits 255-192 + 8 ecc + ERR of each doubleword.

The 256-bit data and the respective ECC parity data are sampled after any single-bit ECC correction and the double error flag is set after any ECC detection. Single-bit correction is always active and logging may be enabled (UT0.SBCE), resulting in the MCR.SBC flag and related address to be captured in ADR register, but this does not change the final signature in the UM0–UM9.

Only data from existing locations are captured by the MISR. The MISR can be seeded to any value by writing the UM0–UM9 registers.

The Array Integrity Self Check consists of the following sequence of events:

1. Set UTE in UT0 by writing the respective password in UT0.
2. Select the block(s) to be checked by writing 1's to the appropriate register(s) in SEL0, SEL1 and/or SEL2 registers. Blocks selected for Array Integrity Check do not need to be unlocked. It is not possible to perform Array Integrity operations on the UTEST nonvolatile memory block.
3. If desired, UT0.AIS can be set for sequential addressing only.
4. Seed the MISR UM0–UM9 with desired values.
5. If breakpoints are desired, set UT0.AIBPE to '1', and ensure that MCR.EER and MCR.SBC are cleared. If breaking on single-bit correction is required, ensure that UT0.SBCE is set.
6. Write a logic '1' to UT0.AIE to start the Array Integrity Check. Array Integrity operations may be aborted prior to UT0.AID going high. This may be done by clearing UT0.AIE. It should be noted that in the event of an aborted Array Integrity check, the MISR registers contain a signature for the portion of the operation that was completed prior to the Abort, and is not definitive. Prior to performing further Array Integrity operations, the UM0–UM9 registers may require initialization to the desired seed value by performing register writes. Array Integrity operations may be suspended while UT0.AID is low. UT0.AISUS may be set to request an Array Integrity Suspend. Once UT0.AID has been set high, UT0.AISUS may be cleared to resume the Array Integrity sequence.
7. Wait until UT0.AID goes high.
8. Check that UT0.NAIBP is set to '1' if breakpoints were previously enabled so that the MCR.EER, MCR.SBC and ADR registers may be checked to determine the cause and address of the break. Prior to resuming the operation, it necessary to clear MCR.SBC or MCR.EER in order to preserve coherency of MCR.SBC, MCR.EER and ADR updates with respect to any subsequent breakpoints. Operation may then be resumed by clearing UT0.NAIBP. Continue waiting until the UT0.AID bit goes high. If breakpoints were not enabled, or if UT0.NAIBP is low when UT0.AID goes high, then the operation is complete.
9. Compare UM0–UM9 content with the expected result.
10. Write a logic '0' to the UT0.AIE bit.

Use sequential addressing for normal integrity checks of the Flash memory. If a more detailed check of the Read path is required (in diagnostic mode), leave UT0.AIS at '0' and use the proprietary address sequence that checks the Read path more completely, however this sequence takes more time.

Caution: Do not modify the content of Block Select (SEL0, SEL1 and SEL2) and Lock (LOCK0, LOCK1 and LOCK2) registers during execution of the Array Integrity Check operation or the MISR value may vary unpredictably.

User mode array Reads requested during the Array Integrity test are ignored to ensure that the Array Integrity operation is not corrupted. The memory array does not respond to array

Read requests during this time. User mode array Reads may be executed if suspended or at a breakpoint.

While UT0.AID is low and UT0.AIE is high, the user may clear AIE, resulting in an Array Integrity Check Abort. UT0.AID must be checked to know when the aborting command has completed.

Array Integrity Check can be suspended by setting UT0.AISUS while Array Integrity Check is running (UT0.AID=0). Once suspended (UT0.AID=1) Array Integrity Check cannot be aborted and clearing UT0.AISUS starts the Resume.

Figure 323. Array Integrity Check of blocks B0F1 and B0F2

```
UM0/9 = 0x0000_0000; /* Reset UM0/9 content */
UT0 = 0xF9F9_9999;   /* Set UTE in UT0: Enable User Test */
SEL0 = 0x0006_0000;  /* Set LSL2-1 in SEL0: Select Blocks */
SEL2 = 0x0000_0006;  /* Set A256KSL2-1 in SEL2: Select Blocks */
UT0 = 0x8000_0002;   /* Set AIE in UT0: Operation Start */
do                   /* Loop to wait for AID=1 */
{ tmp = UT0;         /* Read UT0 */
} while ( !(tmp & 0x0000_0001) );
data0/9 = UM0/9;     /* Read UM0/9 content*/
UT0 = 0x0000_0000;   /* Reset UTE, AIE: Operation End */
```

32.4.5.6.2 Margin Read

User Margin Read may be achieved using the Array Integrity interface and has all the associated features of the Array Integrity interface (MISR calculation, suspend capability and Breakpoints).

Margin Read procedure (either Margin 0 or Margin 1) can be run on selected blocks in order to unbalance the Sense Amplifiers with respect to standard Read conditions, so that all the Read accesses reduce the margin vs ‘0’ (UT0.MRV=0) or vs ‘1’ (UT0.MRV=1).

User Margin Read is a self timed event and is independent of system clocks or wait states selected. Margin Read can be run on selected (SEL0, SEL1 and SEL2) blocks, generating checksum calculations on MISR and ECC flagging. Blocks marked by TMDBS are automatically excluded from MISR signature computation. Once the operations are completed, the results of the Margin Reads can be checked, comparing checksum values in UM0–UM9 and reading MCR.EER and MCR.SBC to determine if zero, one or two bits are being detected by the Margin Read.

The MISR can be seeded to any value by writing the UM0–UM9 registers.

As with Array Integrity Check, the internal MISR calculator is a 64 + 8 + 1-bit register. The 256-bit data, the corresponding ECC parity data and the double ECC (ERR) errors of the four doublewords are therefore captured by the MISR through 4 different Read accesses at the same location. The whole check is accomplished through 4 complete scans of the memory address space:

1. The first pass scans only bits 63-0 + 8 ecc + ERR of each doubleword.
2. The second pass scans only bits 128-64 + 8 ecc + ERR of each doubleword.
3. The third pass scans only bits 191-129 + 8 ecc + ERR of each doubleword.
4. The fourth pass scans only bits 255-192 + 8 ecc + ERR of each doubleword.

The 256-bit data and the respective ECC parity data are sampled after any single-bit ECC correction, and the double error flags are set after any ECC detection. Single-bit logging is always enabled regardless of the value of UT0.SBCE, resulting in the MCR.SBC flag and related address to be captured in ADR register, but this does not alter the final signature in the UM0–UM9.

In any case, the charge losses detected through the Margin Read cannot be considered device failures and no Failure Analysis is initiated on them.

The Margin Read setup operation consists of the following sequence of events:

1. Set UTE in UT0 by writing the respective password in UT0.
2. Select the block(s) to be checked by writing 1's to the appropriate register(s) in SEL0, SEL1 and SEL2 registers. Blocks selected for Margin Read do not need to be unlocked. It is not possible to do Margin Read operations on the UTEST nonvolatile memory block.
3. Seed the MISR UM0–UM9 with desired values.
4. Ensure that MCR.EER and MCR.SBC are cleared.
5. Set UT0.SBCE to '1' to detect single bits during Margin Read.
6. Change the value of UT0.MRE from '0' to '1'.
7. Select the Margin level: UT0.MRV=0 for 0's margin, UT0.MRV=1 for 1's margin.
8. Write a logic '1' to UT0.AIE to start the Margin Read.
9. Wait until UT0.AID goes high.
10. Check that UT0.NAIBP is set to '1' if breakpoints were previously enabled so that MCR.EER, MCR.SBC and ADR.ADDR may be checked to determine the cause and address of the break. Prior to resuming operation, clear MCR.SBC or MCR.EER. Operation may then be resumed by clearing UT0.NAIBP. Continue to wait until UT0.AID goes high. If breakpoints were not enabled, or if UT0.NAIBP is low when UT0.AID goes high, then the operation is complete.
11. Compare UM0–UM9 content with the expected results.
12. Check MCR.EER and MCR.SBC.
13. Write a logic '0' to UT0.AIE, UT0.MRE and UT0.MRV.
14. If more blocks are to be checked, return to step 2.

The linear address sequence is used regardless of the value of UT0.AIS.

Caution: Do not modify the content of Block Select (SEL0, SEL1 and SEL2) and Lock (LOCK0, LOCK1 and LOCK2) registers during execution of the Margin Read operation, otherwise the MISR value may vary unpredictably.

User mode array Reads requested during the Margin Read are ignored to ensure that the Margin Read operation is not corrupted. The memory array does not respond to array Read requests during this time. User mode array Reads may be executed if suspended or at a breakpoint.

While UT0.AID is low and UT0.AIE is high, the user may abort Margin Read by clearing AIE. UT0.AID must be checked to know when the aborting command has completed. Margin Read can be suspended by setting UT0.AISUS while Margin Read is running (UT0.AID=0). Once suspended (UT0.AID=1), Margin Read cannot be aborted and clearing UT0.AISUS starts the Resume.

Figure 324. Margin Read Check versus 1's

```
UM0/9 = 0x0000_0000; /* Reset UM0/9 content */
UT0 = 0xF9F9_9999;   /* Set UTE in UT0: Enable User Test */
SEL0 = 0x0006_0000;  /* Set LSL2-1 in SEL0: Select Blocks */
UT0 = 0x8000_0020;   /* Set MRE in UT0: Select Operation */
UT0 = 0x8000_0030;   /* Set MRV: Select Margin versus 1's */
UT0 = 0x8000_0032;   /* Set AIE in UT0: Operation Start */
do                   /* Loop to wait for AID=1 */
{ tmp = UT0;         /* Read UT0 */
} while ( !(tmp & 0x0000_0001) );
data0/9 = UM0/9;     /* Read UM0/9 content*/
UT0 = 0x8000_0030;   /* Reset AIE in UT0: Operation End */
UT0 = 0x0000_0000;   /* Reset UTE, MRE, MRV, AIS in UT0:
                        Deselect Operation */
```

32.4.6 Protection strategy

Two kinds of protection are available: Modify Protection to avoid unwanted program/erase in Flash blocks and Test Mode Disable to avoid piracy.

32.4.6.1 Modify Protection

The Flash Modify Protection information is stored in nonvolatile Flash cells located in the UTEST region. This information is read once during the Flash Initialization phase following exit from reset and is stored in volatile registers that act as actuators at SoC level. The information is passed to the Flash memory module through erase_lock and program_lock sidebands, and allows independent protection of each block of Low, Mid, High and 256K Address Space Block.

There are also Address Space Block Locking registers provided to independently lock/unlock each block of Low, Mid, High and 256K Address Space Block against program and erase. Locking is done through the LOCK0, LOCK1 and LOCK2 registers. These registers do not have a nonvolatile image stored in the UTEST area of TestFlash, so the locking information is kept on reset: volatile content is all 1's, meaning all blocks are locked.

All the volatile registers can be written to '0' or '1' at any time, therefore the user application can lock and unlock blocks when desired. Note, however, that overall protection is achieved by the combination of sidebands with Low, Mid, High and 256K Address Space Block Locking registers in such a way that a block can be programmed/erased only if both (sidebands and Locking registers) indicate an unlocked state. Sidebands independently control program protection, erase protection and over programming protection (OPP) on a block-by-block basis.

As UTEST and BAF (each one with its own lock bit, LOCK0.TSLOCK and LOCK0.LOWLOCK respectively) are two sections of the same physical block (TestFlash), effective locking is achieved via the 'AND' operand of the lock bits. For effective locking, overall protection is obtained in conjunction with sidebands: UTEST and BAF have dedicated program protection sidebands (SoC specific).

32.4.6.2 Test mode disable

In the TestFlash area a mechanism is also available to disable entry into Test mode. Extreme care must be taken when using this feature, as blocks that are selected for protection in this way cannot be analyzed for possible failures by manufacture failure analysts.

Protection of this sort prevents all high voltage operations to the Flash executed by the internal FPEC as well as reads from FPEC, including Array Integrity Check or Margin Read, when using Test mode interfaces. Protection through other interfaces is managed through normal User mode protection mechanisms, that are SoC specific.

To disable manufacturer entry into Test mode, first program the Test Mode Disable Seal location to 0x5A4B_3C2D and once the next reset is asserted, Test mode is disabled.

It is possible to create a password to enable manufacturer entry into Test mode. This can be programmed into the Test Mode Disable Override Passcode. It is safest to do this prior to censoring the part to avoid unintended lockout. The passcode may not be 0x0000_0000, 0xFFFF_FFFF, or 0x5555_5555. These are all invalid passcodes and are not accepted for override. If the customer desires that override never be possible, one of the three invalid passcodes should be put into this location. Passcodes may be entered to authenticate entry (if enabled) by performing a register Write to address 0x90.

Only blocks selected in the Test Mode Disable Block Select field are controlled by the Test Mode Disable feature. Thus is it possible for customers to select blocks for this type of protection, and render them ineligible for manufacturer failure analysis. Bits programmed to ‘0’ in the Test Mode Disable Block Select field designate blocks that are controlled by Test Mode Disable feature. The TestFlash block, and thus the UTEST and BAF sections, is always protected once the Test Mode Disable Seal password is written. In order to permit two opportunities to select blocks for Test Mode disable, two regions are available: one at address 0x804010, and the other at address 0x804030; both 128-bit sized (only 80 may be effective, depending on actual sectorization). The bits of these two regions are logically ANDed in order to define which blocks are effectively selected for the Test Mode Disable feature.

The 80-bit Block Select field is organized as follows, and aligned in the way blocks are defined in the LOCK registers:

Table 419. Test Mode Disable block select

Block	Block selection bits
Blocks in Mid space	Data[15:0]
Blocks in Low space	Data[29:16]
Blocks in High space	Data[47:32]
Blocks in 256K space	Data[78:48]

33 Decorated Storage Memory Controller (DSMC)

33.1 Introduction

This section details hardware support for atomic read-modify-write memory operations. In the Power Architecture, these capabilities are called “*decorated storage*”. It is supported by capabilities in the processor cores plus instantiations of a Decorated Storage Memory Controller (DSMC).

The Decorated Storage APU defines instructions for providing load and store operations to memory addresses that require *additional semantics* beyond just the reading and writing of data values to the addressed memory locations. Decorated storage operations are intended to be used for specific devices or memory targets that require these additional semantics. A “decoration” is the additional semantic information to be applied to the decorated storage operation by the DSMC.

Consider the basic mnemonics, syntax and formats for the decorated load instructions.

For loads, the syntax is `l[b,h,w]d{cb}x rT,rA,rB` where the data size specifier is defined as 8-bit (`b` = byte), 16-bit (`h` = halfword) or 32-bit (`w` = word), and the three register specifiers include `rT` as the destination target register, `rB` as the effective address and `rA` as the decoration. The optional `{cb}` specifier defines a version of the instruction which is treated as a cache bypass operation.

The syntax for store instruction is `st[b,h,w]d{cb}x rS,rA,rB` where the same data size specifiers are used, and the three register specifiers are `rS` as the source data register, `rB` as the effective address and `rA` as the decoration. The cache bypass attribute is again specified with the use of the optional `{cb}` in the instruction mnemonic.

For all decorated loads and stores, the memory effective address is simply defined by the rB register (hence the `“x”` mnemonic suffix, signaling an “indexed” addressing mode) and the decoration is specified by the rA register. The core transmits the contents of the rA register as the decoration value along with the access address (register rB) to the DSMC.

The decorated load and store instruction formats are shown in *Figure 325*.

Figure 325. Power Architecture Decorated Load and Store Instruction Formats

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
lbdx	0	1	1	1	1	1	**r**T						**r**A				**r**B				1	0	0	0	0	0	0	0	1	1	/
lhdx	0	1	1	1	1	1	**r**T						**r**A				**r**B				1	0	0	0	1	0	0	0	1	1	/
lwdx	0	1	1	1	1	1	**r**T						**r**A				**r**B				1	0	0	1	0	0	0	0	1	1	/
lbdcbx	0	1	1	1	1	1	**r**T						**r**A				**r**B				1	0	0	0	0	0	0	0	1	0	/
lhdcbx	0	1	1	1	1	1	**r**T						**r**A				**r**B				1	0	0	0	1	0	0	0	1	0	/
lwdcbx	0	1	1	1	1	1	**r**T						**r**A				**r**B				1	0	0	1	0	0	0	0	1	0	/
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
stbdx	0	1	1	1	1	1	**r**S						**r**A				**r**B				1	0	1	0	0	0	0	0	1	1	/
sthdx	0	1	1	1	1	1	**r**S						**r**A				**r**B				1	0	1	0	1	0	0	0	1	1	/
stwdx	0	1	1	1	1	1	**r**S						**r**A				**r**B				1	0	1	1	0	0	0	0	1	1	/
stbdcbx	0	1	1	1	1	1	**r**S						**r**A				**r**B				1	0	1	0	0	0	0	0	1	0	/
sthdcbx	0	1	1	1	1	1	**r**S						**r**A				**r**B				1	0	1	0	1	0	0	0	1	0	/
stwdcbx	0	1	1	1	1	1	**r**S						**r**A				**r**B				1	0	1	1	0	0	0	0	1	0	/

The basic decorated load or store memory reference supplements the access address and attribute information with a 32-bit “decoration”. The DSMC decoration defines the special memory operation to be performed and optionally includes bit specifiers and operand data for the command or both.

The default format for the 32-bit decoration field defines a 4-bit command in the most significant bits (decoration[0:3]).

The cache bypass decoration load (`l[b,h,w]dcbx`) and store (`st[b,h,w]dcbx`) instructions do not distinguish between cacheable and cache-inhibited storage attributes, and instead have semantics to effectively emulate a cache-inhibited operation to storage regardless of the actual storage attributes. This is desired in order to allow a small portion of an otherwise cacheable storage area to be treated as non-cacheable using instruction supplied semantics, regardless of storage attributes provided by a memory management, protection or other memory control scheme. This allows decoration or other accesses to be applied to a portion of memory that is normally treated as cacheable according to its storage attributes, as well as to ensure that any stale copies of storage locations present in the cache are not used.

The combination of the decorated load and store instruction support in the e200zX cores plus the decorated storage memory controller in the core platform adds a robust atomic read-modify-write capability to the Power Architecture. The architecture capability defined by these core and platform functions is targeted at manipulation of *n*-bit fields in peripheral registers (and is consistent with I/O hardware addressing in the Embedded C standard) as well as software synchronization data structures needed in multi-core systems (mutexes, semaphores, test-and-set and compare-and-swap structures).

For additional information on the decorated load and store instructions from the core's perspective, refer to the appropriate e200zX core reference manual.

33.2 Decorated Stores: st[b,h,w]d{cb}x rS,rB,rA

The next sections present descriptions of the specific operations, based on the 4-bit command field defined in decoration[0:3]. These descriptions include the bit pattern definitions for the 32-bit decoration value and include pseudo-code detailing the sequence of operations. The decoration formats are defined using a <command>.<size> syntax where the operand size specifier is b (byte, 8-bit), h (halfword, 16-bit) or w (word, 32-bit). The operand size is specified directly in the decorated instruction executed in the core, and this attribute is driven to the system bus as part of the decorated data transfer. Additionally, the write data is taken from the right-justified 8, 16 or 32 bits of the rS register, that is:

```
 8-bit wdata = rS[24:31]
16-bit wdata = rS[16:31]
32-bit wdata = rS[ 0:31]
```

Likewise, read data is loaded into the right-justified 8, 16 or 32 bits of the rT register:

```
 8-bit rdata = rT[24:31]
16-bit rdata = rT[16:31]
32-bit rdata = rT[ 0:31]
```

For the byte and halfword decorated load instructions, the upper bits of the rT register are zero filled.

The starting bit (SRTBIT) position follows the Power Architecture convention where the MSB is bit 0 and the LSB is bit 31. The bit field width (BFW) value defines the width and BFW = 0 specifies the maximum width, that is, the container size.

The basic decorated store includes three data transfer fields sourced from the core: the access address (rB), the decoration (rA) and the write data (wdata) operand (rS). There are five operations defined and most of these transactions convert a single core AHB write bus cycle into an atomic read-modify-write, that is, an indivisible read followed by write sequence. Support for three basic boolean logic functions (AND, OR, XOR) is provided along with a compare-and-store operation plus a bit field insert operation. These operations do not support any type of bit field wrapping.

In the next sections detailing the decorated store operations, the following associations between the pseudocode variables and the core registers apply.

- decoration = rA
- accessAddress = rB
- wdata = rS

33.2.1 Bit Field Insert (BFINS) into an 8, 16 or 32-bit Memory Container

Figure 326. Decoration Format: BFINS

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
bfins.b	0	0	0	0	0	0	0	SRTBIT			0	0	0	BFW			-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
bfins.h	0	0	0	0	0	0	SRTBIT				0	0	BFW				-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
bfins.w	0	0	0	0	0	SRTBIT					0	BFW					-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

This command inserts a bit field defined by SRTBIT and BFW into the memory “container” defined by the access size associated with the decorated store instruction using an atomic read-modify-write sequence.

```
tmp = mem[accessAddress, size] // memory read

if bfw == 0
   then bfw = container

if ((srtbit + bfw) <= container)// generate bit mask
    mask = ((1 << bfw) - 1) << (container - (srtbit + bfw))
else
    mask = ((1 << bfw) - 1) >> ((srtbit + bfw) - container)

tmp = tmp   & ~mask// modify
    | wdata &  mask

mem[accessAddress, size] = tmp// memory write
```

The write data operand (wdata) associated with the decorated store instruction contains the bit field to be inserted. *It must be properly aligned within a right-justified container in the source register (rS)*, that is, within the lower 8 bits for a byte operation, the lower 16 bits for a halfword or the entire 32 bits for a word operation.

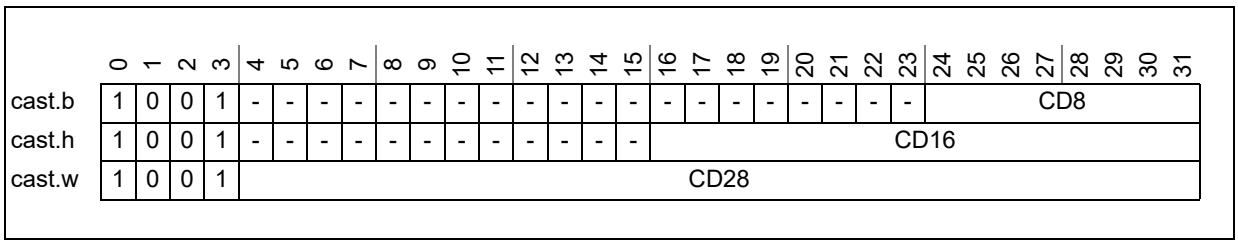
To illustrate, consider the following example of the insertion of the 3-bit field “xyz” into an 8-bit memory container, initially set to “abcd_efgh”. For all cases, the bit field width (BFW) is 3.

```
if SRTBIT = 0 and the decorated store rS register[24:31] = xyz-_----,
   then destination is “xyzd_efgh”
if SRTBIT = 1 and the decorated store rS register[24:31] = -xyz_----,
   then destination is “axyz_efgh”
if SRTBIT = 2 and the decorated store rS register[24:31] = --xy_z---,
   then destination is “abxy_zfgh”
if SRTBIT = 3 and the decorated store rS register[24:31] = ---x_yz--,
   then destination is “abcx_yzgh”
if SRTBIT = 4 and the decorated store rS register[24:31] = ---_xyz-,
   then destination is “abcd_xyzh”
if SRTBIT = 5 and the decorated store rS register[24:31] = ---_-xyz,
   then destination is “abcd_exyz”
if SRTBIT = 6 and the decorated store rS register[24:31] = ---_--xy,
   then destination is “abcd_efxy”
if SRTBIT = 7 and the decorated store rS register[24:31] = ---_---x,
   then destination is “abcd_efgx”
```

If the bit field insert operation specifies SRTBIT as 0 and BFW as 0 (indicating the width matches the container width), then the operation is logically equivalent to a simple register store operation.

33.2.2 Compare-and-Store (CAST)

Figure 327. Decoration Format: CAST



This command begins by performing a read of the referenced memory address and comparing that data with the operand included in the decoration word. If the read data is equal to the compare operand, the write data associated with the decorated store instruction is placed into the memory location, else the original read data is rewritten into the memory location. The write operand is selected from the appropriate data byte lanes in the same manner as any memory store operation; this implies the write data operand is right justified in the rS source register.

```
tmp = mem[accessAddress, size] // memory read

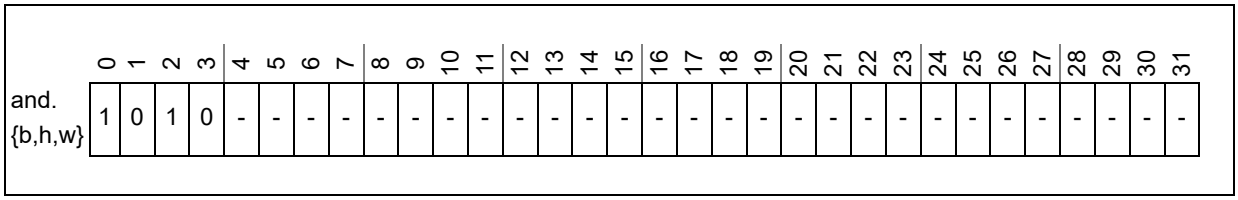
if (size == 8)// define compare_operand
    compare_operand = CD8
else if (size == 16)
          compare_operand = CD16
     else compare_operand = {0x0, CD28}// zero-filled data

if (tmp == compare_operand)// compare - "modify"
     mem[accessAddress, size] = wdata// memory write
else mem[accessAddress, size] = tmp
```

Note for the word size operation (cast.w), the data value specified in the low-order 28 bits of the decoration is zero filled in the most significant bits to create the required 32 bit compare operand.

33.2.3 Logical AND (AND)

Figure 328. Decoration Format: AND

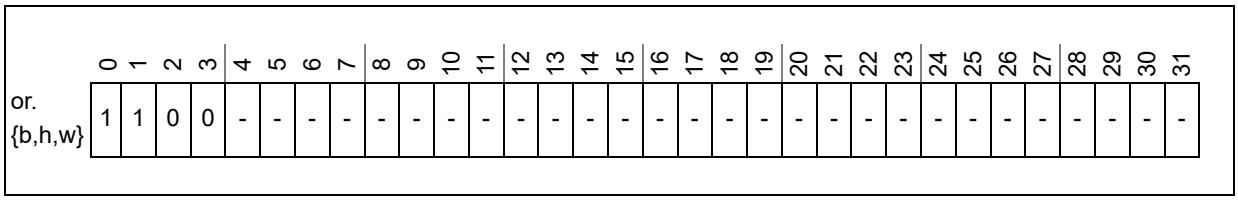


This command performs an atomic read-modify-write of the referenced memory location. First, the location is read; it is then modified by performing a logical AND operation using the write operand defined by the rS register; finally, the result of the AND operation is written back into the referenced memory location.

```
tmp = mem[accessAddress, size] // memory read
tmp = tmp & wdata// modify
mem[accessAddress, size] = tmp// memory write
```

33.2.4 Logical OR (OR)

Figure 329. Decoration Format: OR

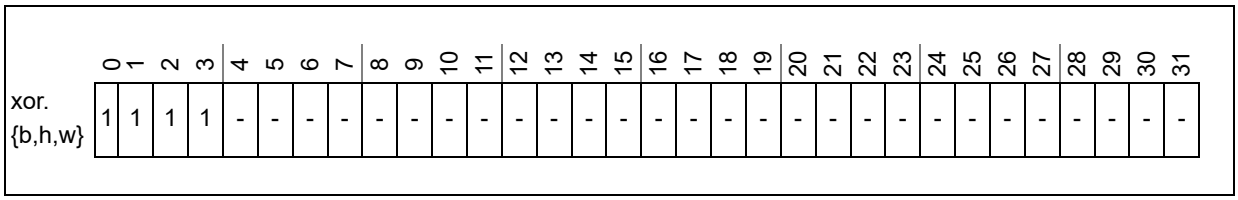


This command performs an atomic read-modify-write of the referenced memory location. First, the location is read; it is then modified by performing a logical OR operation using the write operand defined by the rS register; finally, the result of the OR operation is written back into the referenced memory location.

```
tmp = mem[accessAddress, size] // memory read
tmp = tmp | wdata// modify
mem[accessAddress, size] = tmp// memory write
```

33.2.5 Logical Exclusive-OR (XOR)

Figure 330. Decoration Format: XOR



This command performs an atomic read-modify-write of the referenced memory location. First, the location is read; it is then modified by performing a logical XOR operation using the write operand defined by the rS register; finally, the result of the XOR operation is written back into the referenced memory location.

```
tmp = mem[accessAddress, size] // memory read
tmp = tmp ^ wdata// modify
mem[accessAddress, size] = tmp// memory write
```

33.3 Decorated Loads: `l[b,h,w]d{cb}x rT,rB,rA`

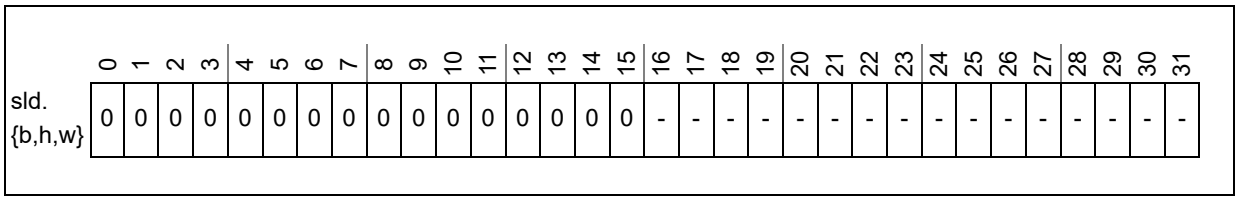
The basic decorated load includes two data transfer fields sourced from the core: the access address (rB), the decoration (rA) plus the read data (rdata) operand returned from DSMC to the core and loaded into the destination register (rT). There are 3 operations defined and two of these transactions convert a single core AHB read bus cycle into an atomic read-modify-write. Support for a swap function, a load-and-set-1-bit functions are provided along with a simple load (aka an “extract”) operation.

In the next sections detailing the decorated load operations, the following associations between the pseudocode variables and the core registers apply.

- decoration = rA
- accessAddress = rB
- rdata = rT

33.3.1 Simple Load (SLD)

Figure 331. Decoration Format: SLD



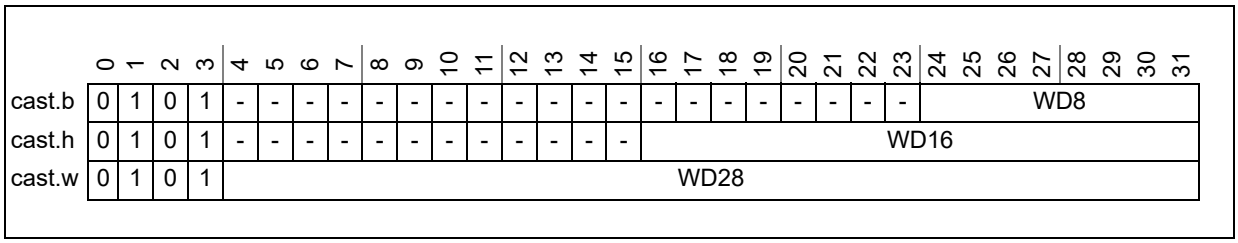
This command performs a simple memory load of the reference size from the access address. For this operation, the 12-bit field defined in decoration[4:15] must be zeroes else the transfer is not performed and error terminated.

```
rdata = mem[accessAddress, size] // memory read
```

Support for generic bit field extract operations is best handled using existing core and compiler capabilities involving standard load instructions followed by left and right shift operations to emulate these functions.

33.3.2 Registers-and-Memory Exchange (SWAP)

Figure 332. Decoration Format: SWAP



This command loads the contents of the referenced memory location into the core's rT register and stores the source operand from the core's rA register (the decoration) into the memory location.

```
tmp   = mem[accessAddress, size] // memory read
rdata = tmp// rT = rdata = tmp

if (size == 8)// store_operand - "modify"
    store_operand = WD8
else if (size == 16)
          store_operand = WD16
     else store_operand = {0x0, WD28}// zero-filled write data

mem[accessAddress, size] = store_operand// memory write
```

Note for the word size operation (swap.w), the data value specified in the low-order 28 bits of the decoration is zero filled in the most significant bits to create the required 32 bit store operand.

33.3.3 Load-and-Set-1(Bit) (LAS1)

Figure 333. Decoration Format: LAS1

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
las1.b	0	1	1	0	0	0	0	BIT			-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
las1.h	0	1	1	0	0	0	BIT				-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
las1.w	0	1	1	0	0	BIT					-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

This command first reads the referenced memory location; it then modifies the read operand by setting the single bit position defined in the decoration (BIT); the modified data is then written back to the referenced memory location and the original read data returned to the initiating processor core.

```
tmp   = mem[accessAddress, size] // memory read
rdata = tmp// rT = rdata = tmp
mask  = 1 << (container - bit - 1)// generate bit mask
tmp   = tmp | mask// modify
mem[accessAddress, size] = tmp// memory write
```

33.4 DSMC Instantiations

The decorated storage memory controller which performs these operations is instantiated multiple times within the core platform and physically resides between the data port of the cores and the master ports of the crossbar.

The module includes error checking logic that validates the decoration field is properly defined (all illegal commands are rejected and the transfer error terminated). Additionally, the decorated memory controller only operates on aligned, single data transfers (misalignment and bursts are not supported and error terminated if attempted).

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

34 Successive Approximation Register Analog-to-Digital Converter (SARADC)

34.1 Introduction

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

34.2 Overview

The SARADC digital interface contains advanced features for normal or injected conversion modes of operation. The conversion can be triggered by software or hardware. For advanced cross triggering functions, it provides interface with an on-chip Crosstriggering Unit (CTU) which can support two different CTU Modes: CTU trigger mode and CTU Control Mode (see details in *Section 34.4.5: CrossTriggering Unit (CTU) interface*).

The digital interface can be configured to control up to 256 multiplexed analog input channels. There are three types of input channels:

- Internal channels, mapped to index ranging from 0 to 95.
- Test channels, mapped to index ranging from 96 to 127.

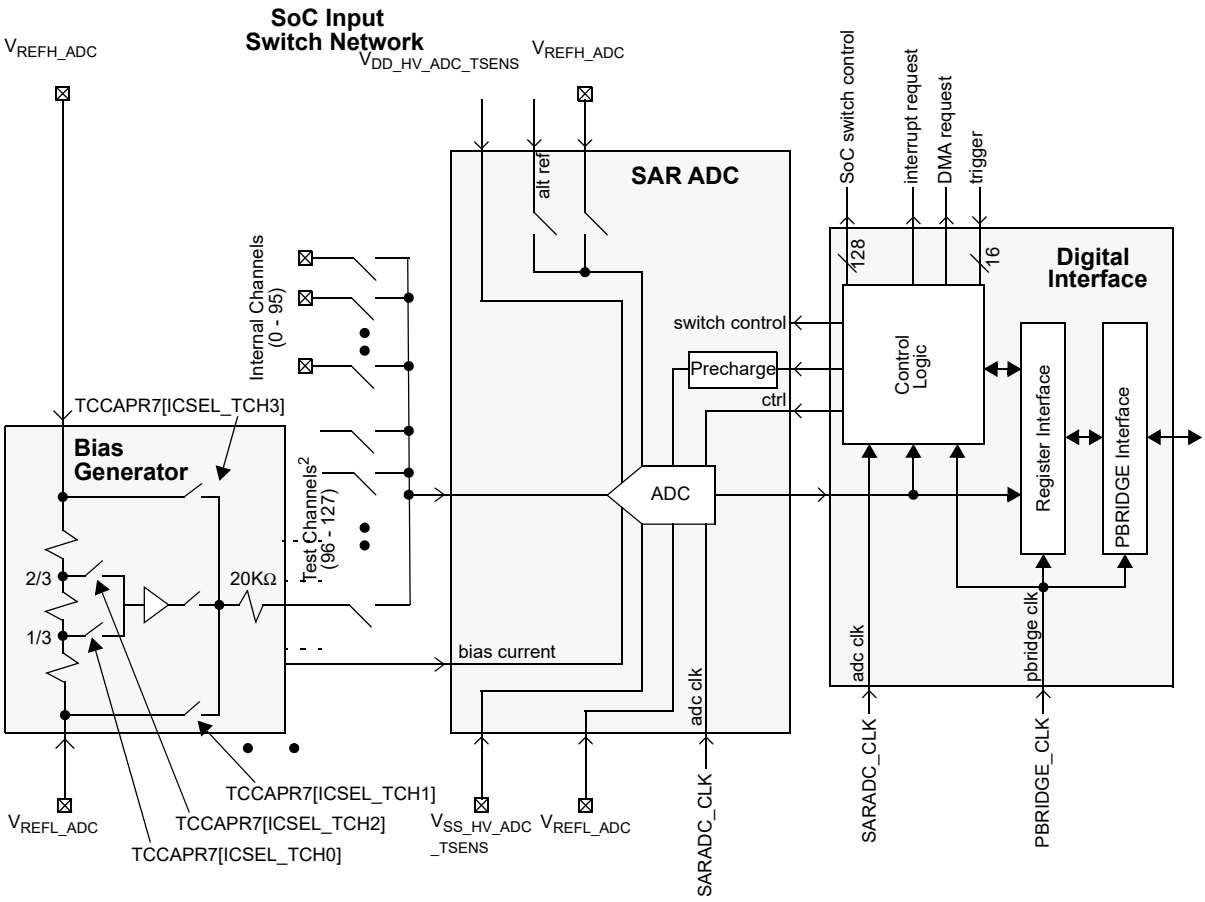
Note: The number of input channels and their mapping depends on the device configuration. See the device-specific Analog-to-Digital Converters (ADC) Configuration section (analog input pin multiplexing and SAR ADC channel assignment) for a channel mapping table.

The mask registers present within the digital interface can be properly programmed to configure which channel has to be converted.

Four conversion timing registers allow configuration of different precharging and sampling durations, one of which can then be assigned to each channel. For each channel, it is possible to choose the voltage reference through software. Analog Watchdogs allow continuous hardware monitoring of analog input channels. The digital interface also provides for interrupt/DMA support for various conditions related to end of channel conversions.

Figure 334 shows the SARADC block diagram.

Figure 334. SARADC block diagram



Note: Test channels are mapped to internal analog channels via the SARADCB_TCCAPRx registers.

34.3 Feature description

34.3.1 Main features

- Selectable 10-bit or 12-bit data resolution output for 12-bit SARADC.
- Selectable 8-bit or 10-bit data resolution output for 10-bit SARADC.
- Up to 96 internal channels, 32 test channels; variable number of analog channels of each type controlled by parameters
- 4 different conversion timing registers selectable for any channel
- Mapping of external channel to any internal channel through static programming by software
- Shorting of test channel with internal channel through static programming by software
- Normal conversion with One Shot/Scan modes
- Injected conversion with One Shot mode
- 2 different abort features that allow to abort either a single channel conversion or chain conversion
- Reference selection for each channel
- Power Down Mode
- Dedicated data register for each channel, containing the following information:
  - Conversion result in one half-word:
    10-bit or 12-bit for 12-bit SARADC.
    8-bit or 10-bit for 10-bit SARADC.
  - Status byte which provides some conversion information such as mode of operation (Normal, Injected, or CTU), data valid, data overwritten status
  - Control byte for reference selection, conversion timing parameter selection
- Configurable number of analog watchdogs.
- 2 different CTU modes (CTU Control mode and CTU Trigger mode) available when CTU feature is present in SARADC instance.
- Interrupt/DMA support for the following conditions
  - End of conversion of single channel for both normal, injected conversions
  - End of conversion chain for both normal, injected conversions
  - End of CTU conversion
  - Watchdog thresholds crossover

34.4 Functional description

Two main conversion types are available within the SARADC digital interface:

- Normal conversion
- Injected conversion

Note: "precharge" option on this device means "de-charge" to Reference ground VSS_HV_ADR_S .

34.4.1 Normal channel conversion

This is the normal conversion that the user programs by configuring the normal conversion mask registers (ICNCMR0–2, TCNCMR). Each channel can be individually enabled by setting '1' in the corresponding field of these registers. Mask registers must be programmed before starting the conversion and cannot be changed until the conversion of all the selected channels ends (MSR[NSTART] bit is reset).

34.4.1.1 Start of normal conversion

The normal conversion can be started as follows:

- By software
  - The normal conversion chain starts when the MCR[NSTART] bit is set. It is recommended that the normal trigger enable MCR[NTRGEN] bit is reset during conversions started by software. Once normal conversion has started, any further write operation of MCR[NSTART] bit has no effect during ongoing conversion, but the conversion is enqueued after completion of current conversion in oneshot mode.

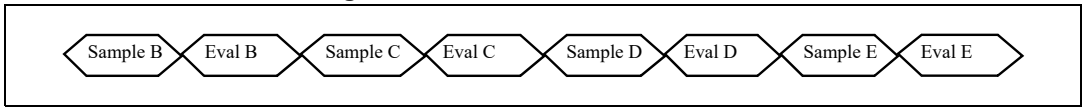
34.4.1.2 Normal conversion operating modes

Two operating modes are available for the normal conversion:

- One Shot Mode
- Scan Mode

To enter one of these modes, it is necessary to program the MCR[MODE] bit. The first phase of the conversion process involves sampling the analog channel and the next phase involves the conversion phase when the sampled analog value is converted to digital as shown in *Figure 335*.

Figure 335. Normal conversion flow



In **One Shot Mode** (MODE = 0), a sequential conversion specified in the ICNCMR0–2, TCNCMR, ECNCMR0–3 mask registers is performed only once. At the end of each conversion, the digital result of the conversion is stored into the corresponding data register.

For Example, channels A-B-C-D-E-F-G-H are present in the device where channels B-D-E are to be converted in the One Shot Mode. MODE = 0 is set for One Shot mode. Conversion starts from the channel B followed by conversion of channels D-E. At the end of conversion of channel E, the scanning of channels stops.

The MSR[NSTART] status bit is automatically set when the normal conversion starts. At the same time MCR[NSTART] bit is reset if the conversion is started by software.

Once the normal conversion operation starts in oneshot mode and another software conversion request is made writing the MCR[NSTART] bit and before MSR[NSTART] is cleared, one additional conversion is initiated with the same channel mask register settings after the current conversion is finished. This is called **enqueuing**. Enqueuing can help in removing the delay between two consecutive one shot chains and uncertainty caused by metastability between MCR[NSTART] bit written at system clock and generation of start pulse on ADC clock before every chain execution.

In **Scan Mode** (MODE = 1), a sequential conversion of N channels specified in the ICNCMR0–2, TCNCMR, ECNCMR0–3 registers is continuously performed. As in the previous case, at the end of each conversion the digital result of the conversion is stored into the corresponding data register.

In this mode, the software can start conversion by writing '1' to MCR[NSTART] bit or a valid normal trigger edge is detected which also sets MCR[NSTART] bit. The MSR[NSTART] status bit is automatically set when the normal conversion starts. Unlike One Shot Mode, the MCR[NSTART] bit is not reset. It can be reset by software when the user needs to stop scan mode. In this case, the current scan conversion is completed and MSR[NSTART] bit is also reset after the last conversion of the chain.

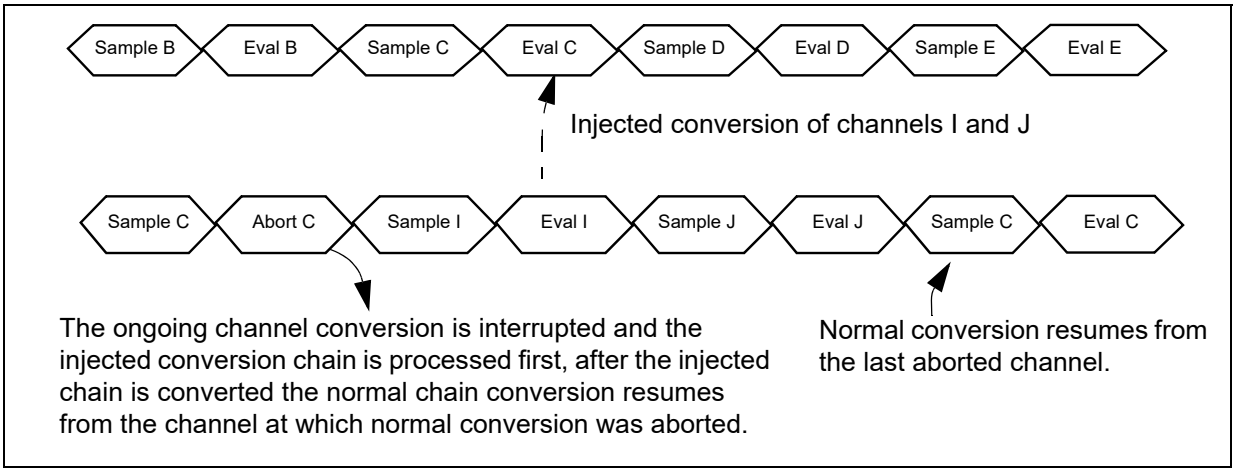
For Example, channels A-B-C-D-E-F-G-H are present in the device where channels B-D-E are to be converted in the *Scan Mode*. MODE = 1 is set for Scan Mode. Conversion starts from the channel B followed by conversion of the channels D-E. At the end of conversion of channel E the scanning of channel B starts followed by conversion of the channels D-E. This sequence repeats itself until the MCR[NSTART] bit is reset by software.

In both modes, at the end of each channel conversion an End Of Conversion interrupt is issued (if enabled by the corresponding mask bit), and at the end of the conversion sequence an End Of Chain interrupt is issued (if enabled by the corresponding mask bit). The End Of Chain interrupt is issued for each pass of scan mode.

34.4.2 Injected channel conversion

A conversion chain can be injected into the current normal conversion by configuring the injected mask registers (ICJCMR0–2, TCJCMR). As in normal conversion, each internal, test channel can be individually selected. This injected conversion can only be in one shot mode and interrupts the normal conversion. When an injected conversion is inserted, ongoing channel conversion is aborted and the injected channel request is processed. After the last channel in the injected chain is converted, normal conversion resumes from the channel at which the normal conversion was stopped, as shown in *Figure 336*.

Figure 336. Injected sample/conversion sequence



The injected conversion can be started as follows:

- By software
  - The injected conversion chain starts when the MCR[JSTART] bit is set. The current conversion is suspended and the injected chain is converted. At the end of the chain, the MSR[JSTART] bit is reset and the normal chain conversion is resumed. It is recommended that the injection trigger enable MCR[JTRGEN] bit is reset during conversions started by software. Once injected conversion has started, any further write operation of MCR[JSTART] bit will not have any effect during ongoing conversion and the new request will get enqueued and executed after current injected chain finishes.

The MSR[JSTART] status bit is automatically set when the injected conversion starts. At the same time MSR[JSTART] is reset if the conversion is started by software.

At the end of each injected conversion, an End Of Injected Conversion (JEOC) interrupt is issued (if enabled by the corresponding mask bit) and at the end of the sequence an End Of Injected Chain (JECH) interrupt is issued (if enabled by the corresponding mask bit).

34.4.3 Abort conversion

Two different abort functions are provided:

- The user can abort the ongoing conversion by setting the ABORT bit in the MCR. The current conversion is aborted and the conversion of the next channel of the chain is immediately started (generating a new start pulse to the SARADC). In the case of an abort operation, the NSTART/JSTART bit remains set and the ABORT bit is reset after the conversion of the next channel starts. The EOC corresponding to the aborted channel is not generated. This behavior is true for normal or triggered/Injected conversion modes. If the last channel of a chain is aborted, the end of chain is reported generating an ECH interrupt. ABORT function should not be used when only single channel is selected in a conversion chain, instead ABORTCHAIN can be used.
- It is also possible to abort the current chain conversion by setting the ABORTCHAIN bit in the MCR. In that case the behavior of the ADC depends on the MODE bit. In fact, if scan mode is disabled, the NSTART bit is automatically reset together with the ABORTCHAIN bit. Otherwise, if the MODE bit is set to ‘1’, a new chain conversion is started. The EOC of the current aborted conversion is not generated but an ECH interrupt is generated to signal the end of the chain.

  When a chain conversion abort is requested (ABORTCHAIN bit is set) while an injected conversion is running over a suspended Normal conversion, there are 2 cases. If One Shot Mode is enabled (MODE=0 in MCR register) then both injected chain and Normal conversion chain are aborted (both the NSTART and JSTART bits are also reset). If the Scan Mode is enabled (MODE=1 in MCR register) theninjected conversion is stopped and the suspended Normal conversion is restarting. (only JSTART bit is reset).

Note: ABORT/ABORTCHAIN functions are not supported during CTU initiated conversions.

34.4.4 Analog conversion timings

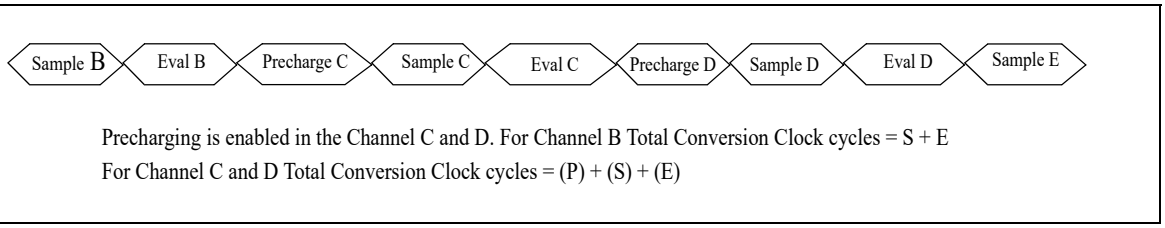
34.4.4.1 Conversion timings

In order to support different loadings and switching times (in particular for the external channel types) four different Conversion Timing registers are present (CTR0–3). Each conversion timing register contains PRECHG, INPSAMP bitfields to program the required

duration for precharging and sampling phases. The selection of these registers for each channel is done by the SAMPSEL bitfield of corresponding channel data register.

Precharging phase allows to precharge or discharge the internal capacitor to a fixed internal voltage for duration determined by PRECHG bitfield prior to actual analog channel sampling start. This is useful for resetting information regarding the last converted data. Precharging feature can be enabled or disabled for each channel by programming the PCE bit of corresponding channel data register. If precharging is enabled for a given channel, the normal sequence of operation is precharging, then sampling, then evaluation, as shown in *Figure 337*. Please refer to device datasheet for the minimum precharge and sample durations required.

Figure 337. Analog conversion sequence



Bitfields PRECHG, INPSAMP are used to define the total conversion duration (t_{conv}) and in particular the partition among precharge duration (t_{prechg}), sampling phase duration (t_{sample}) and evaluation phase duration (t_{eval}).

The precharging phase duration is given by

$$t_{prechg} = PRECHG * t_{ck}$$

where PRECHG must be greater than or equal to 2 (hardware requirement) and PCE bit of channel data register is ‘1’. In case the value of PRECHG is found to be less than 2, it is automatically set to 2 inside SARADC. If the PCE bit is ‘0’, the precharging phase is skipped and conversion starts with sampling phase directly.

The sampling phase duration is given by the following equation:

$$t_{sample} = INPSAMP * t_{ck}$$

where INPSAMP must be greater than or equal to 5 (hardware requirement). In case the value of INPSAMP is found to be less than 5, it is automatically set to 5 inside SARADC. The total evaluation phase duration is given by the following equations:

$$t_{eval} = EVAL * t_{ck}$$

In this phase, the internal sampling capacitor is disconnected from the analog channel and ADC estimates the digitized value of the sampled input using successive approximation algorithm. In the evaluation phase, all the bits are estimated sequentially to provide the conversion result.

The total conversion duration is (not including external multiplexing) is given by the following:

$$t_{conv} = t_{prechg} + t_{sample} + t_{eval}$$

The timings refer to the unit t_{ck} which is the inverse of f_{ck}, where f_{ck} = SARADC peripheral clock. Refer to the device data sheet for the total conversion time.

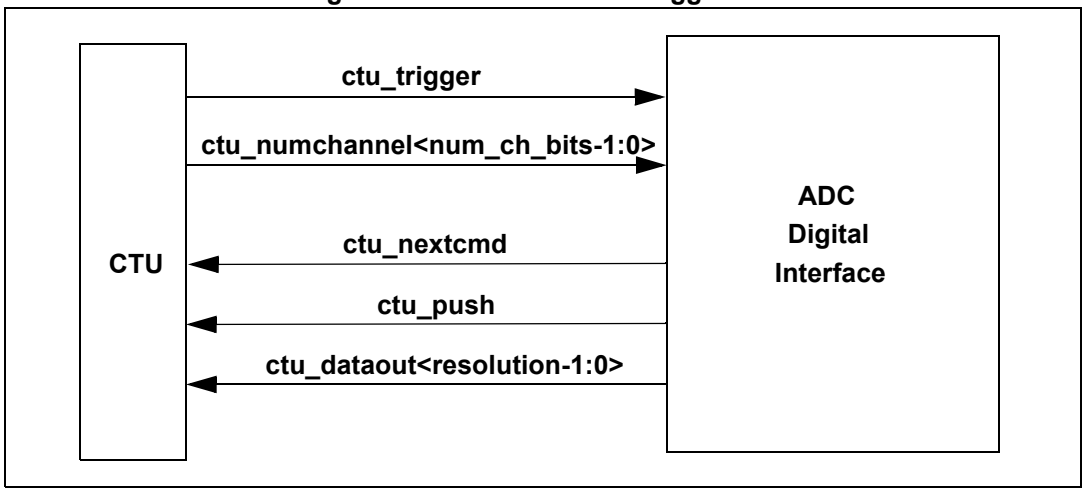
Note: The current implementation of ADCDIG does not support hardware restriction over values of PRECHG and INPSAMP fields. As minimum values for these fields are product specific and need to be managed through software.

34.4.5 CrossTriggering Unit (CTU) interface

The Cross Triggering Unit (CTU) is added to enhance the injected conversion capability of the SARADC. The CTU contains multiple event inputs that can be used to select the channels to be converted from the appropriate event configuration register. In *Figure 338* the CTU/ADC interface is shown.The CTU generates a trigger output pulse of one clock cycle and outputs onto a data bus which channel has to be converted. A single channel is converted for each request. After performing the conversion, the digital interface returns the conversion result to CTU. The conversion result is also saved in the corresponding channel data register and it is compared with watchdog thresholds if requested. Together with the converted data, it also sends a request for next command and a control which can be used to push the conversion result into CTU data storage registers/fifo.

These two pulses named *ctu_nextcmd* and *ctu_push* spaced of one or two clock cycle (of analog ADC clock). The ctu_nextcmd is set to '1' during the last clock cycle or last but one of ADC evaluation phase (depending on ADC type used). The ctu_push is asserted when the conversion is finished meaning that the ctu_dataout is valid.

Figure 338. SARADC crosstrigger unit



Different CTU modes can be selected programming the CTUEN and CTU_MODE fields of the MCR with respect of CTSEN SARADC parameter as described in the following table.

Table 420. CTU modes

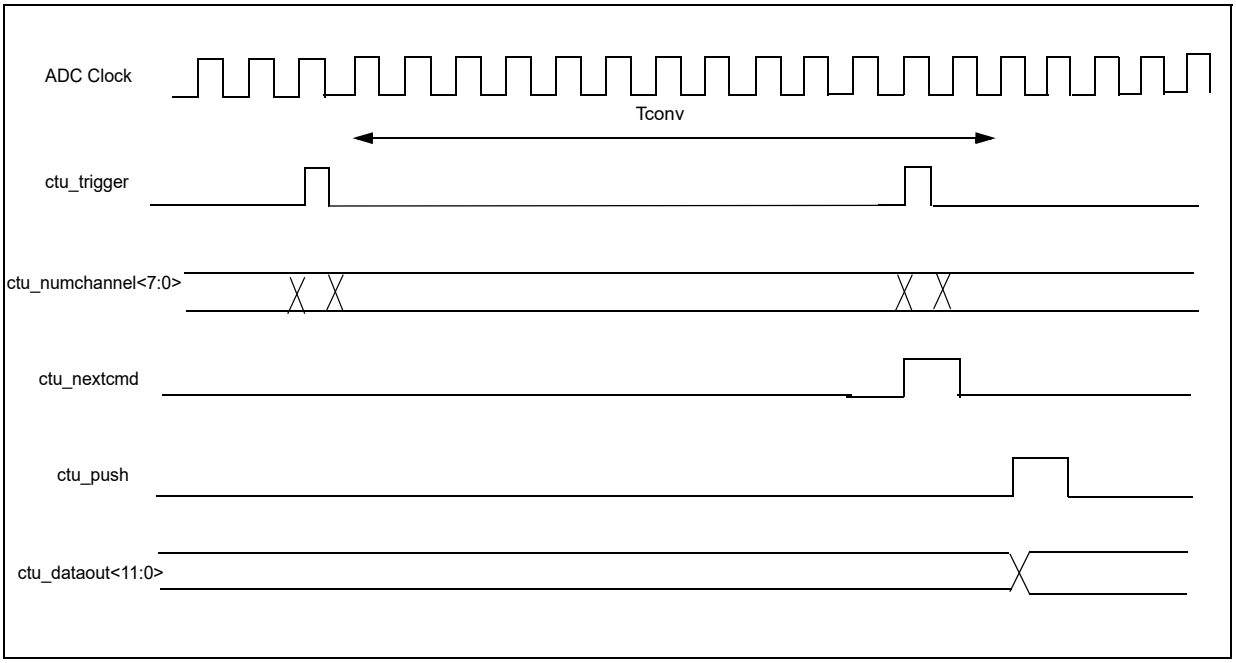
CTSEN(1)	CTUEN	CTU_MODE	CTU mode
—	0	—	CTU is disabled
0	1	—	CTU Trigger Mode is enabled
1	1	—	CTU Control Mode is enabled
2	1	1	CTU Trigger Mode is selected
2	1	0	CTU Control Mode is selected

1. Refer to SAR ADC triggering section of chapter Analog-to-Digital Converters (ADC) Configuration for CTSEN value applicable on each SAR_ADC instance.

The CTU, SARADC analog block and digital Interface are synchronous to each other, the CTU clock and ADC clock post division should be synchronous and phase aligned to ensure

jitter free start pulses to ADC analog. Timings of CTU Interface signals are described in *Figure 339*.

Figure 339. CTU interface timings



34.4.5.1 CTU trigger mode

In CTU trigger mode, normal and injected conversions triggered by the CPU are still enabled. When a trigger pulse and the injected channel are received internally by the ADCDig, the conversion starts. The CTUSTART bit in the MSR is set automatically at this point and it is also automatically reset when the triggered injected conversion is completed.

If an injected conversion (programmed by the user by setting the JSTART bit) is ongoing and a pulse is received on the CTU trigger input line, then the injected channel conversion chain is aborted and only the triggered CTU injected conversion proceeds. Aborting the injected conversion, the JSTART bit in the MSR is reset to ‘0’. That abort is signalled through the status bit JABORTCHAIN in the MSR.

If normal conversion is ongoing and a pulse is received on the CTU trigger input line, then ongoing channel conversion is aborted and the triggered injected conversion is processed. When it is finished, normal conversion resumes from the channel at which the normal conversion was aborted.

If another CTU trigger pulse is received before the end of conversion, that request is discarded and this error condition is reported by CTUTRGERR bit in ISR.

When a normal conversion is requested during CTU conversion (CTUSTART bit = ‘1’), the normal conversion starts when CTU conversion is completed (CTUSTART reset). Otherwise, when an Injected conversion is requested during CTU conversion, that conversion is discarded and the JSTART bit in the MCR is immediately reset.

CTU trigger mode can be entered by setting CTUMODE=’1’ during powerdown and CTUEN=’1’ any time after the powerdown is exited. CTU trigger mode can be exited by clearing the CTUEN bit, But care must be taken not to exit the CTU mode when some CTU

initiated conversion is ongoing. Also power down request (by setting MCR[PWDN] bit) should not be made while CTU transactions are ongoing.

34.4.5.2 CTU control mode

When CTU Control mode is enabled, the CPU is able to write in the ADC registers but it cannot start any conversion. Conversion requests can be generated only by the CTU trigger pulse. If a normal or injected conversion is requested, it is automatically discarded.

When a CTU sends trigger pulse and channel number to be converted to SARADC digital interface, triggered injected conversion starts. The MSR[CTUSTART] bit is set automatically on the first received trigger pulse and it can be reset only when CTU is disabled (CTUEN bit to ‘0’).

CTU control mode can be entered by setting CTUMODE=’0’ during powerdown and CTUEN=’1’ any time after the powerdown is exited. CTU control mode can be exited by clearing the CTUEN bit, But care should be taken not to enter or exit the CTU mode when some conversion is ongoing. Also power down request (by setting MCR[PWDN] bit) should not be made while CTU transactions are ongoing.

34.4.6 Test channel connection with internal analog channel

The test channels are meant for monitoring various on-chip analog signals coming from temperature sensor, bandgap, voltage regulator and different voltage levels available on the chip. These analog signals can be interfaced to SARADC through test channels for testing purpose. Test channels are mapped from 96 to 127. Each test channel is controlled in the same way as normal internal channel.

SAR ADC self-testing method: the test channel[127:124] is also mapped to various V_ref voltages (4 voltages) selection switches in the ADCBIAS block. The decoding of these channels to switch on and select a particular V_ref voltage is done in SoC level muxing logic.

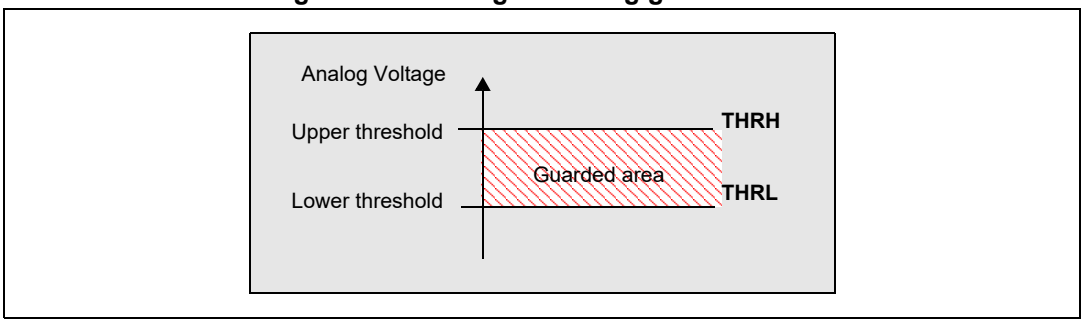
It is also possible to perform test channel conversions with and without shorting the normal analog channel pin with the internal test voltage.

After the conversion is performed on a test channel the converted data is stored at two CDR locations at the end of conversion, first location is the CDR belonging to the test channel on which conversion is performed and second location is the CDR belonging to internal channel on which test channel is mapped.

34.4.7 Programmable analog watchdog

Analog watchdogs are used for determining whether the result of the conversion of a channel lies within a given guard area (as shown in the *Figure 340*) bounded by upper and a lower threshold values named THRH and THRL respectively.

Figure 340. Analog watchdog guarded area



Following conversion of the selected channel, a comparison is performed between the converted value and the threshold values. If the converted value lies outside that guard area, then corresponding threshold violation interrupts are generated. The comparison result is stored as WDGxH and WDGxL bits in the WTISR as explained in *Table 421*. Depending on the MSKWDGxL and MSKWDGxH mask bits in the WTIMR, an interrupt is generated on threshold violation.

Table 421. Values of WDGxH and WDGxL fields

WDGxH	WDGxL	Converted data
1	0	converted data > THRH
0	1	converted data < THRL
0	0	THRH ≤ converted data ≤ THRL

The lower and higher threshold values for the analog watchdog are programmed using THRHLR0–15 registers. Analog watchdog functionality for each channel can be enabled independently from ICWENR0–2, TCWENR, ECWEN0–3 registers and can select the watchdog threshold registers (THRHLR0–15) to be used by programming ICWSELR0–11, TCWSELR0–3, ECWSELR0–15 registers. The threshold registers selected by WSEL_CHx field of watchdog threshold selection registers will provide the threshold values.

For example, if channel 30 is to be monitored with the threshold values in register THRHLR12, then WSEL_CH30 is programmed to select THRHLR12 register to provide threshold values. Enabling is done by setting the bit corresponding to channel 30 in ICWENR0 register.

If the converted value for a particular channel lies outside the range specified by threshold values, then the corresponding bit is set in ICAWORR0–2, TCAWORR, ECAWORR0–3 registers. In this case, a set of threshold registers (THRHLR0–15) can be linked to several analog channels. The threshold values to be selected for a channel needs be programmed only once in ICWSELR, TCWSELR, ECWSELR registers.

Note: If the higher threshold for the analog watchdog is programmed lower than the lower threshold and the converted value is less than the lower threshold, then the WDGxL interrupt for the low threshold violation is set, else if the converted value is greater than the lower threshold (consequently also greater than the higher threshold) then the interrupt WDGxH for high threshold violation is set. The user should therefore take care to avoid this situation as it could lead to misinterpretation of the watchdog interrupts.

34.4.8 DMA functionality

A Direct Memory Access (DMA) request can be programmed after the conversion of every channel by setting the respective masking bit in ICDSR0–2, TCDSR and ECDSR0–3 registers. The DMA masking registers must be programmed before starting any conversion. The DMA transfers can be enabled using the DMAE[DMAEN] bit. When the DMAE[DCLR] bit is set, the DMA request is cleared on the reading of the register for which DMA transfer has been enabled.

34.4.9 Interrupts

The SARADC digital interface generates the following maskable interrupt signals:

- End Of Conversion interrupts
  - NEOC (end of normal conversion of each channel)
  - NECH (end of normal chain)
  - JEOC (end of injected conversion of each channel)
  - JECH (end of injected chain)
  - EOCTU (End of CTU Conversion)

The EOC_CH[x] bit of interrupt pending registers (ICIPR0–2, TCIPR) is set when channel x completes the conversion (normal or injection). This pending status is qualified when the corresponding mask bit IM_CH[x] is also set in the interrupt mask registers (ICIMR0–2, TCIMR).

The NEOC interrupt is generated only if the following conditions are met

- ISR[NEOC] bit is set after the normal conversion completion for any channel x.
- IMR[MSKNEOC] bit is set
- EOC_CH[x], IM_CH[x] bits are set at least for one channel x

The JEOC interrupt is generated only if the following conditions are met

- ISR[JEOC] bit is set after the injected conversion completion for any channel x.
- IMR[MSKJEOC] bit is set
- EOC_CH[x], IM_CH[x] bits are set at least for one channel x

The EOCTU interrupt is generated only if the following conditions are met

- ISR[EOCTU] bit is set after the CTU triggered conversion completion for any channel x.
- IMR[MSKEOCTU] bit is set
- EOC_CH[x], IM_CH[x] bits are set at least for one channel x

It is recommended to clear the NEOC/JEOC/EOCTU bits of ISR and all the individual channel pending bits which are not masked while servicing the interrupt.

The ISR[NECH] bit is set when a normal channel conversion chain is completed. This pending status is qualified when the corresponding mask bit IMR[MSKNECH] is also set.

The ISR[JECH] bit is set when an injected channel conversion chain is completed. This pending status is qualified when the corresponding mask bit IMR[MSKNECH] is also set.

- WDGxL and WDGxH (Watchdog threshold) interrupt requests

The interrupts generated due to the analog watchdog are handled by registers WTISR and WTIMR in order to check and enable the interrupt requests to external interrupt controller module. The watchdog interrupt source sets two pending bits WDGxH and WDGxL for each

channel being monitored in the WTISR. The interrupt request is generated if at least one of the following conditions are met:

- WTISR[WDGxL] bit is set and the corresponding mask bit MSKWDGxL is also set in WTIMR for at least one channel.
- WTISR[WDGxH] bit is set and the corresponding mask bit MSKWDGxH is also set in WTIMR for at least one channel.

The ISR and WTISR contain the interrupt pending request status flags. In case the user wants to clear a particular interrupt event status, writing a '1' to the corresponding status bit clears the pending interrupt flag. (At this write operation, all other bits of the ISR must be maintained at '0'.)

The ADC module provides dedicated interrupt outputs for all interrupt sources. Also, all interrupt lines are OR-ed together to provide a single output to the external interrupt controller if a combined interrupt is required.

34.4.10 Power down mode

The SARADC analog block can be put in low current consumption mode by setting the PWDN bit in MCR. This is the default mode after exiting from reset. This state must be exited before starting any operation by resetting the PWDN bit in the MCR. When in powerdown mode, no conversion should be started. Bit ADCSTATUS[0] in the MSR is set only when ADC enters Powerdown mode.

The power-down mode can be requested by setting the PWDN bit in the MCR. If a one-shot conversion is ongoing, the SARADC cannot immediately enter the power-down mode. In fact, the SARADC enters power-down mode only after completing the ongoing conversion. If a scan conversion is ongoing, the ongoing operation shall be first aborted manually by resetting the NSTART bit and using the ABORTCHAIN bit. A power Down request while scan mode conversion is ongoing is ignored.

After the power down phase is completed, the process ongoing before the power down phase must be restarted manually (by setting the appropriate START bit). Resetting PWDN bit and setting NSTART or JSTART bit during the same cycle is forbidden.

CTU triggered conversion should not be initiated during power down mode. In both CTU modes, to request power down mode the application has to wait for the end of the current CTU conversion (CTUSTART bit automatically reset when conversion finishes). Additionally, when SARADC is in CTU control mode the application needs to exit the CTU control mode (by resetting the CTUEN bit) before requesting the power down mode.

34.4.11 Stop Mode

The SARADC can be request to enter this low power mode anytime during the operation. If stop mode is requested while some Normal/Injected mode chained conversion are ongoing, the current channel conversion will be completed and upon stop mode exit the conversion will be resumed from next channel. However, stop mode should not be requested while CTU triggered conversions are ongoing.

34.5 Memory map and registers

This section provides memory map for the SARADC digital interface.

Note:

Table 422. SARADC digital interface register map

Offset	Register	Register classification(1)	Location
0x000	Main Configuration Register (MCR)	Saf-Relv	*Section 34.5.1.1*
0x004	Main Status Register (MSR)	No-Saf	*Section 34.5.1.2*
0x008–0x00C	Reserved		
0x010	Interrupt Status Register (ISR)	No-Saf	*Section 34.5.1.3*
0x014–0x01C	Internal channel Interrupt Pending Registers 0–2 (ICIPR0–ICIPR2)	Saf-Relv	*Section 34.5.1.4*
0x020	Interrupt Mask Register (IMR)	Saf-Relv	*Section 34.5.1.5*
0x024–0x02C	Internal Channel Interrupt Mask Registers 0–2 (ICIMR0–ICIMR2)	Saf-Relv	*Section 34.5.1.6*
0x030	Watchdog Threshold Interrupt Status Register (WTISR)	Saf-Relv	*Section 34.5.1.7*
0x034	Watchdog Threshold Interrupt Mask Register (WTIMR)	Saf-Relv	*Section 34.5.1.8*
0x038–0x03C	Reserved		
0x040	DMA Enable Register (DMAE)	Saf-Relv	*Section 34.5.1.9*
0x044–0x04C	Internal Channel DMA Select Registers 0–2 (ICDSR0–ICDSR2)	Saf-Relv	*Section 34.5.1.10*
0x050–0x05C	Reserved		
0x060–0x06C	Watchdog Threshold Registers 0–3 (WTHRHLR0–WTHRHLR3)	Saf-Relv	*Section 34.5.1.11*
0x070–0x090	Reserved		
0x094–0x0A0	Conversion Timing Registers 0–3 (CTR0–3)	Saf-Relv	*Section 34.5.1.12*
0x0A4–0x0AC	Internal Channel Normal Conversion Mask Registers 0–2 (ICNCMR0–ICNCMR2)	Saf-Relv	*Section 34.5.1.13*
0x0B0	Reserved		
0x0B4–0x0BC	Internal Channel Injected Conversion Mask Registers 0–2 (ICJCMR0–ICJCMR2)	Saf-Relv	*Section 34.5.1.14*
0x0C0–0x0C4	Reserved		
0x0C8	Power Down Exit Delay Register (PDEDR)	Saf-Relv	*Section 34.5.1.15*
0x0CC–0x0FC	Reserved		
0x100–0x27C	Internal Channel Data Registers 0–95 (ICDR0–ICDR95)	Saf-Relv	*Section 34.5.1.16*
0x280–0x2AF	Reserved		
0x2B0–0x2DC	Internal Channel Watchdog Selection Registers 0–11 (ICWSELR0–ICWSELR11)	Saf-Relv	*Section 34.5.1.17*
0x2E0–0x2E8	Internal Channel Watchdog Enable Registers 0–2 (ICWENR0–ICWENR2)	Saf-Relv	*Section 34.5.1.18*
0x2EC–0x2EF	Reserved		
0x2F0–0x2F8	Internal Channel Analog Watchdog Out of Range Registers 0–2 (ICAWORR0–ICAWORR2)	No-Saf	*Section 34.5.1.19*

Table 422. SARADC digital interface register map (continued)

Offset	Register	Register classification(1)	Location
0x2FC–0x3FC	Reserved		
0x400	Test Channel Interrupt Pending Register (TCIPR)	Saf-Relv	*Section 34.5.1.20*
0x404	Test Channel Interrupt Mask Register (TCIMR)	Saf-Lat	*Section 34.5.1.21*
0x408	Test Channel DMA Select Register (TCDSR)	Saf-Lat	*Section 34.5.1.22*
0x40C	Test Channel Normal Conversion Mask Register (TCNCMR)	Saf-Lat	*Section 34.5.1.23*
0x410	Test Channel Injected Conversion Mask Register (TCJCMR)	Saf-Lat	*Section 34.5.1.24*
0x414–0x420	Test Channel Watchdog Selection Registers 0–3 (TCWSELR0–TCWSELR3)	Saf-Lat	*Section 34.5.1.25*
0x424	Test Channel Watchdog Enable Register (TCWENR)	Saf-Relv	*Section 34.5.1.26*
0x428	Test Channel Analog Watchdog Out of Range Register (TCAWORR)	Saf-Relv	*Section 34.5.1.27*
0x42C	Reserved		
0x430–0x44C	Test Channel Connection with Analog Pin Registers 0–7 (TCCAPR0–TCCAPR7)	Saf-Relv	*Section 34.5.1.28*
0x450–0x4CC	Test Channel Data Registers 96-127 (TCDR96-TCDR127)	Saf-Relv	*Section 34.5.1.29*
0x4D0–0x4FC	Reserved		

1. No-Saf = Not Safety Relevant, Saf-Relv = Safety Relevant, Saf-Lat = Safety Latent

34.5.1 Register descriptions

34.5.1.1 Main Configuration Register (MCR)

Offset: 0x000 Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	OWREN	WLSIDE	MODE	Reserved	NSTART	Reserved	Reserved		JSTART	Reserved	Reserved		Reserved	0	CTUEN(1)	CTU_MODE(1)
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	Reserved	0	0	0	Reserved				ABORTCHAIN	ABORT	0	FRZ	0	Reserved	Reserved	PWDN
W																
Reset	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

1. Only available with cross-triggering unit (CTU) feature.
2.

Figure 341. Main Configuration Register (MCR)

Table 423. MCR field descriptions

Field	Description
0 OWREN	Overwrite enable This bit enables or disables the functionality to overwrite unread converted data in ICDR, TCDR, ECDR registers. 0 Prevents overwrite of unread converted data, new result is discarded. 1 Enables converted data to be overwritten by a new conversion.
1 WLSIDE	Write left/right-aligned 0 The conversion data in ICDR, TCDR, ECDR registers is written right-aligned. 1 Conversion data is left-aligned (from 15 to (15 – resolution + 1))
2 MODE	One Shot/Scan 0 One Shot Mode: Configures the normal conversion of one chain. 1 Scan Mode: Configures continuous chain conversion mode; when the programmed chain conversion is finished it restarts immediately
4 NSTART	Normal Start conversion Setting this bit starts the chain or scan conversion. Resetting this bit during scan mode causes the current chain conversion to finish, then stops the operation. 0 Causes the current chain conversion to finish and stops the operation. 1 Starts the chain or scan conversion

Table 423. MCR field descriptions (continued)

Field	Description
8 JSTART	Injected Start conversion Setting this bit will start the configured injected analog channels to be converted by software. Resetting this bit has no effect, as the injected chain conversion cannot be interrupted. 0 Writing '0' has no effect 1 Starts the injected chain conversion
14 CTUEN	Cross Trigger Unit Enable 0 The cross triggering unit is disabled and the triggered injected conversion cannot take place 1 The cross triggering unit is enabled and the triggered injected conversion can take place **Note:** Can be set or cleared anytime with some restriction mentioned in CTU trigger/control mode section.
15 CTU_MODE	Cross Trigger Unit Mode This bit is present only if generic parameter CTSEN = 2, otherwise this bit is a reserved bit and always read as 0. This should be set only when ADC Digital is in powerdown mode. 0 CTU control mode is selected 1 CTU trigger mode is selected
24 ABORTCHAIN	Abort Chain When this bit is set, the ongoing Chain Conversion is aborted. This bit is reset by hardware as soon as a new conversion is requested. 0 Conversion is not affected 1 Aborts the ongoing chain conversion **Note:** Behavior of this request during CTU mode is mentioned in CTU trigger/control mode section.
25 ABORT	Abort Conversion When this bit is set, the ongoing conversion is aborted and a new conversion is invoked. This bit is reset by hardware as soon as a new conversion is invoked. 0 Conversion is not affected 1 Aborts the ongoing conversion **Note:** Behavior of this request during CTU mode is mentioned in CTU trigger/control mode section.

Table 423. MCR field descriptions (continued)

Field	Description
27 FRZ	Freeze This bit enables to stop the SARADC conversions at the end of current channel conversion when SoC enters debug mode. 0 Conversions are not stopped 1 When SoC enters debug mode, further conversions by SARADC analog block will be stopped. The ongoing channel conversion will be completed and the converted data is stored. When the SoC exits debug mode, SARADC resumes the job that was left before halting the conversions. Changing configuration registers during halted condition may cause a pending operation to fail when normal operation is resumed
31 PWDN	Power-down enable When this bit is set, the analog module is requested to enter Power Down mode. When SARADC status is PWDN, resetting this bit starts SARADC transition to IDLE mode. 0 SARADC is in normal mode 1 SARADC has been requested to power down

34.5.1.2 Main Status Register (MSR)

Offset: 0x004 Access: User Read Only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	NSTART	0	0	0	JSTART	0	0	0	0	JABORTCHAIN	0	CTUSTART(1)
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	CHADDR								0	0	0	0	0	ADCSTATUS		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

1. Only available with cross-triggering unit (CTU) feature.

Figure 342. Main Status Register (MSR)

Table 424. MSR field descriptions

Field	Description
4 NSTART	This status bit is used to signal that a normal conversion is ongoing. 0 Normal conversion is not taking place. 1 Normal conversion is ongoing or the normal conversion is pending due to injected conversion or CTU triggered injected conversion.
8 JSTART	This status bit is used to signal that an injected conversion is ongoing. 0 Injected conversion is not taking place. 1 Injected conversion is ongoing.
13 JABORTCHAIN	This status bit is used to signal that an Injected conversion chain has been aborted. This bit is reset when a new conversion starts. 0 Last injected conversion chain has not been aborted. 1 Last injected conversion chain has been aborted.
15 CTUSTART	This status bit is used to signal that a CTU conversion is ongoing. This bit is set when a CTU trigger pulse is received and the CTU conversion starts. When CTU trigger mode is enabled this bit is automatically reset when the conversion is completed. Otherwise, if Control Mode is enabled this bit can be reset only when the CTU is disabled (CTUEN set to ‘0’). 0 CTU triggered injected conversion is not taking place. 1 CTU triggered injected conversion is ongoing.
16:23 CHADDR	Channel under measure address This status bit is used to signal which channel is under measure.
29:31 ADCSTATUS	The value of this parameter depends on ADC status. While requesting an external channel conversion, SARADC digital interface needs to wait for a fixed time determined by DSD bitfield of DSDR before proceeding to actual sampling phase. This is known as wait state. 000 IDLE 001 Power-down 010 Wait state 011 — 100 Sample 101 — 110 Conversion 111 —

34.5.1.3 Interrupt Status Register (ISR)

Offset: 0x010 Access: User Read/Write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	CTUTRGERR(1)	EOCTU(1)	JEOC	JECH	NEOC	NECH
W											w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. Only available with cross-triggering unit (CTU) feature.

Figure 343. Interrupt Status Register (ISR)

Table 425. ISR field descriptions

Field	Description
26 CTUTRGERR	CTU Trigger Error Flag This Error Flag indicates that the next CTU trigger has been discarded as it overlapped during execution of current CTU conversion. This indicated that CTU sampling duration is less than programmed conversion time of ADC. 0 No CTU trigger overlap error condition since last CTU conversion. 1 CTU trigger overlap error has occurred.
27 EOCTU	End of CTU channel conversion interrupt flag This bit indicates the end of conversion for a CTU injected channel. 0 End of conversion of CTU triggered channel has not occurred since the last time the flag is cleared. 1 End of conversion of CTU triggered channel has occurred or the flag has not been cleared since the last end of conversion occurrence.
28 JEOC	End of injected channel conversion interrupt flag This bit indicates the end of conversion for an injected channel. 0 End of conversion of injected channel has not occurred since the last time the flag is cleared. 1 End of conversion of injected channel has occurred or the flag has not been cleared since the last end of conversion occurrence.
29 JECH	End of injected chain conversion interrupt flag This bit indicates the end of conversion for an injected chain. 0 End of conversion of injected chain has not occurred since the last time the flag is cleared. 1 End of conversion of injected chain has occurred or the flag has not been cleared since the last end of conversion occurrence.

Table 425. ISR field descriptions (continued)

Field	Description
30 NEOC	End of normal channel conversion interrupt flag This bit indicates the end of conversion for a normal channel. 0 End of conversion of normal channel has not occurred since the last time the flag is cleared. 1 End of conversion of normal channel has occurred or the flag has not been cleared since the last end of conversion occurrence.
31 NECH	End of normal chain conversion interrupt flag This bit indicates the end of conversion for an normal chain. 0 End of conversion of normal chain has not occurred since the last time the flag is cleared. 1 End of conversion of normal chain has occurred or the flag has not been cleared since the last end of conversion occurrence.

34.5.1.4 Internal Channel Interrupt Pending Register *n* (ICIPR*n*)

The interrupt channel register to channel association is described in *Table 427*.

Offset: 0x014 + *n**0x4 (*n* = 0 to 2) Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	EOC_CH[x]															
W	w1c															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	EOC_CH[x]															
W	w1c															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 344. Internal Channel Interrupt Pending Register *n* (ICIPR*n*)

Table 426. ICIPR*n* field descriptions

Field	Description
0:31 EOC_CH[x]	End of conversion interrupt pending bit for channel x 0 End of conversion for CH[x] has not occurred. 1 End of conversion for CH[x] has occurred.

Table 427. ICIPR0–2 Registers to channel association

Register	Register bits 0:31
ICIPR0	EOC_CH[31:0]
ICIPR1	EOC_CH[63:32]
ICIPR2	EOC_CH[95:64]

Note: If channel CH[x] is not existing then the corresponding bit field EOC_CH[x] is reserved and returns 0.

If all channels CH[x] are not existing, the full register is not existing and any attempt to access this space generates a transfer error.

34.5.1.5 Interrupt Mask Register (IMR)

The description of the Interrupt Mask Register is given below.

Offset: 0x020 Access: User Read/Write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	MSKEOCTU(1)	MSKJEOC	MSKJECH	MSKNEOC	MSKNECH
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. Only available with cross-triggering unit (CTU) feature.

Figure 345. Interrupt Mask Register (IMR)

Table 428. IMR field descriptions

Field	Description
27 MSKEOCTU	Mask bit for EOCTU 0 EOCTU interrupt is disabled. 1 EOCTU interrupt is enabled.
28 MSKJEOC	Mask bit for JEOC 0 JEOC interrupt is disabled. 1 JEOC interrupt is enabled.
29 MSKJECH	Mask bit for JECH 0 JECH interrupt is disabled. 1 JECH interrupt is enabled.
30 MSKNEOC	Mask bit for NEOC 0 NEOC interrupt is disabled. 1 NEOC interrupt is enabled.
31 MSKNECH	Mask bit for NECH 0 NECH interrupt is disabled. 1 NECH interrupt is enabled.

34.5.1.6 Internal Channel Interrupt Mask Register *n* (ICIMR*n*)

The interrupt mask register to channel association is described in *Table 430*.

Offset: 0x024 + *n**0x4 (*n* = 0 to 2) Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	IM_CH[x]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	IM_CH[x]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 346. Internal Channel Interrupt Mask Register *n* (ICIMR*n*)

Table 429. ICIMR*n* field descriptions

Field	Description
0:31 IM_CH[x]	IM_CH[x]: Interrupt mask bit for channel x 0 Interrupt for CH[x] is disabled. 1 Interrupt for CH[x] is enabled.

Table 430. Interrupt Mask Registers to Channel Association

Register	Register bits 0:31
ICIMR0	IM_CH[31:0]
ICIMR1	IM_CH[63:32]
ICIMR2	IM_CH[95:64]

Note: If channel CH[x] is not existing then the corresponding bit field IM_CH[x] is reserved and returns 0.

If all channels CH[x] are not existing, the full register is not existing and any attempt to access this space generates a transfer error.

34.5.1.7 Watchdog Threshold Interrupt Status Register (WTISR)

This register gives the interrupt status information for the 16 possible upper and 16 possible lower threshold limits monitored by up to 16 watchdog monitors which can be selected for each channel.

Offset: 0x030 Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	WDG15H	WDG15L	WDG14H	WDG14L	WDG13H	WDG13L	WDG12H	WDG12L	WDG11H	WDG11L	WDG10H	WDG10L	WDG9H	WDG9L	WDG8H	WDG8L
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	WDG7H	WDG7L	WDG6H	WDG6L	WDG5H	WDG5L	WDG4H	WDG4L	WDG3H	WDG3L	WDG2H	WDG2L	WDG1H	WDG1L	WDG0H	WDG0L
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 347. Watchdog Threshold Interrupt Status Register (WTISR)

Table 431. WTISR field descriptions

Field	Description
30-2x WDGxH [x = 0–15]	This corresponds to the interrupt generated on the converted value being higher than the programmed higher threshold as reported by WDG monitor ‘x’. 0 Converted data is not higher than the programmed higher threshold. 1 Converted data is higher than the programmed higher threshold.
31-2x WDGxL [x = 0–15]	This corresponds to the interrupt generated on the converted value being lower than the programmed lower threshold as reported by WDG monitor ‘x’. 0 Converted data is not lower than the programmed lower threshold. 1 Converted data is lower than the programmed lower threshold.

34.5.1.8 Watchdog Threshold Interrupt Mask Register (WTIMR)

This register gives the interrupt mask information for the 16 possible upper and 16 possible lower threshold limits which can be selected for each channel.

Offset: 0x034 Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	MSKWDG15H	MSKWDG15L	MSKWDG14H	MSKWDG14L	MSKWDG13H	MSKWDG13L	MSKWDG12H	MSKWDG12L	MSKWDG11H	MSKWDG11L	MSKWDG10H	MSKWDG10L	MSKWDG9H	MSKWDG9L	MSKWDG8H	MSKWDG8L
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	MSKWDG7H	MSKWDG7L	MSKWDG6H	MSKWDG6L	MSKWDG5H	MSKWDG5L	MSKWDG4H	MSKWDG4L	MSKWDG3H	MSKWDG3L	MSKWDG2H	MSKWDG2L	MSKWDG1H	MSKWDG1L	MSKWDG0H	MSKWDG0L
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 348. Watchdog Threshold Interrupt Mask Register (WTIMR)

Table 432. WTIMR field descriptions

Field	Description
30-2x MSKWDGxH [x = 0–15]	This corresponds to the mask bit for the interrupt generated on the converted value being higher than the programmed higher threshold. 0 Interrupt for WDGxH is disabled. 1 Interrupt for WDGxH is enabled.
31-2x MSKWDGxL [x = 0–15]	This corresponds to the mask bit for the interrupt generated on the converted value being lower than the programmed lower threshold. 0 Interrupt for WDGxL is disabled. 1 Interrupt for WDGxL is enabled.

34.5.1.9 DMA Enable Register (DMAE)

The DMA Enable (DMAE) register sets up the DMA for use with the SARADC.

Offset: 0x040 Access: User Read/Write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	DCLR	DMAEN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 349. DMA Enable Register (DMAE)

Table 433. DMAE register field descriptions

Field	Description
30 DCLR	DMA clear sequence enable 0 DMA request cleared by Acknowledge from DMA controller. 1 DMA request cleared on read of data registers.
31 DMAEN	DMA global enable 0 DMA feature disabled. 1 DMA feature enabled.

34.5.1.10 Internal Channel DMA Select Register *n* (ICDSR*n*)

The DMA select registers to channel association is described in *Table 435*.

Offset: 0x044 + *n**0x4 (*n* = 0 to 2) Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	DS_CH[x]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	DS_CH[x]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 350. Internal Channel DMA Select Register *n* (ICDSR*n*)

Table 434. ICDSR*n* field descriptions

Field	Description
0:31 DS_CH[x]	DMA select for channel x 0 CH[x] is disabled to transfer data in DMA mode. 1 CH[x] is enabled to transfer data in DMA mode.

Table 435. Internal Channel DMA Select Registers to Channel Association

Register	Register bits 0:31
ICDSR0	DS_CH[31:0]
ICDSR1	DS_CH[63:32]
ICDSR2	DS_CH[95:64]

Note: If channel CH[x] is not existing then the corresponding bit field DS_CH[x] is reserved and returns 0.

If all channels CH[x] are not existing, the full register is not existing and any attempt to access this space generates a transfer error.

34.5.1.11 Watchdog Threshold Register *n* (WTHRHLR*n*)

Note: The number of WTHRHLR registers corresponds to the number of analogs watchdogs implemented in the ADC module.

The length of the threshold field depends on the SARADC resolution. For 10-bit SARADC, the bits 4, 5, 20 and 21 are reserved.

Note: The reset value of the bits 4 and 5 is 0 instead of 1 in case SARADC resolution is 10-bit.

Offset: 0x060 + *n**0x4 (*n* = 0 to 3) Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	THRH											
W																
Reset	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	THRL											
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 351. Watchdog Threshold Register *n* (WTHRHLR*n*)

Table 436. WTHRHLR*n* field descriptions

Field	Description
4:15 THRH	High threshold value for channel x
20:31 THRL	Low threshold value for channel x

34.5.1.12 Conversion Timing Register *n* (CTR*n*)

Offset: 0x094 + *n**0x4 (*n* = 0 to 3) Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	CRES	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	PRECHG				INPSAMP							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 352. Conversion Timing Register *n* (CTR*n*)

Table 437. CTR*n* field descriptions

Field	Description
0 CRES	Conversion Resolution select This bit selects the conversion resolution for the conversion. 0 12-bit resolution in case of 12-bit SARADC. 10-bit resolution in case of 10-bit SARADC 1 10-bit resolution in case of 12-bit SARADC. 8-bit resolution in case of 10-bit SARADC
20:23 PRECHG	Precharging phase duration This bitfield defines the precharging phase duration to be applied when this phase is enabled by PCE bit of channel data register. This duration is calculated as (PRECHG X 1/Frequency of SARADC clock). Please refer to device datasheet for the minimum duration required for precharging phase.
24:31 INPSAMP	Sampling phase duration This bitfield defines the sampling phase duration to be applied. This duration is calculated as (INPSAMP X 1/Frequency of SARADC clock). Please refer to device datasheet for the minimum duration required for sampling phase.

34.5.1.13 Internal Channel Normal Conversion Mask Register *n* (ICNCMR*n*)

The normal conversion mask registers to channel association is described in *Table 439*.

Offset: 0x0A4 + *n**0x4 (*n* = 0 to 2) Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	NCE_CH[x]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	NCE_CH[x]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 353. Internal Channel Normal Conversion Mask Register *n* (ICNCMR*n*)

Table 438. ICNCMR*n* field descriptions

Field	Description
0:31 NCE_CH[x]	Normal conversion enable for channel x 0 Normal conversion is disabled for CH[x]. 1 Normal conversion is enabled for CH[x].

Table 439. Internal Channel Normal Conversion Mask Registers to Channel Association

Register	Register bits 0:31
ICNCMR0	NCE_CH[31:0]

Table 439. Internal Channel Normal Conversion Mask Registers to Channel Association (continued)

Register	Register bits 0:31
ICNCMR1	NCE_CH[63:32]
ICNCMR2	NCE_CH[95:64]

Note: If channel CH[x] is not existing then the corresponding bit field NCE_CH[x] is reserved and returns 0.

If all channels CH[x] are not existing, the full register is not existing and any attempt to access this space generates a transfer error.

34.5.1.14 Internal Channel Injected Conversion Mask Register *n* (ICJCMR*n*)

The injected conversion mask registers to channel association is described in *Table 441*.

Offset: 0x0B4 + n*0x4 (n = 0 to 2) Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	JCE_CH[x]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	JCE_CH[x]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 354. Internal Channel Injected Conversion Mask Register *n* (ICJCMR*n*)

Table 440. ICJCMR*n* field descriptions

Field	Description
0:31 JCE_CH[x]	Injected conversion enable for channel x 0 Injected conversion is disabled for CH[x]. 1 Injected conversion is enabled for CH[x].

Table 441. Internal Channel Normal Conversion Mask Registers to Channel Association

Register	Register bits 0:31
ICJCMR0	JCE_CH[31:0]
ICJCMR1	JCE_CH[63:32]
ICJCMR2	JCE_CH[95:64]

Note: If channel CH[x] is not existing then the corresponding bit field JCE_CH[x] is reserved and returns 0.

If all channels CH[x] are not existing, the full register is not existing and any attempt to access this space generates a transfer error.

34.5.1.15 Power Down Exit Delay Register (PDEDR)

Offset: 0x0C8 Access: User Read/Write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	PDED							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 355. Power Down Exit Delay Register (PDEDR)

Table 442. PDEDR field descriptions

Field	Description
24:31 PDED	Power down exit delay duration Defines the delay between the power-down bit reset and the starting of conversion. The power-down delay is calculated as [PDED × 1/(SARADC clock frequency)].

34.5.1.16 Internal Channel Data Register n (ICDR*n*)

The conversion results for each channel is loaded into data registers. Each data register will contain the conversion result, status information related to ADC mode, data valid, timing parameter selection for each channel.

This register should be accessed via 32-bit R/W.

Offset: 0x100 + *n**0x4 (*n* = 0 to 95) Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	PCE	0	CTSEL		0	0	0	0	VALID	OVERW	RESULT	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	CDATA															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 356. Internal Channel Data Register *n* (ICDR*n*)

Table 443. ICDR*n* field descriptions

Field	Description
4 PCE	Precharge Enable This bit enables the precharging phase during channel conversion. 0 Precharge phase is disabled. 1 Precharge phase is enabled.
6:7 CTSEL	This bitfield selects the conversion timing register for each channel to select different precharge and sampling phase durations. 00 CTR0 is selected. 01 CTR1 is selected. 10 CTR2 is selected. 11 CTR3 is selected.
11	Reserved
12 VALID	Data valid flag This bit is used to notify when the data is valid (a new value has been written). It is automatically cleared when data is read. 0 Converted data has been read by software. 1 Converted data is valid and has not been read yet.
13 OVERW	Data overwritten flag This bit signals that the previous converted data has been overwritten by a new conversion. This functionality depends on the value of MCR[OWREN]: – When OWREN = 0, OVERW is frozen to 0 and CDATA field is protected against being overwritten until being read. – When OWREN = 1, OVERW flags the CDATA field overwrite status. 0 Converted data has not been overwritten. 1 Previous converted data has been overwritten before having been read.
14:15 RESULT	Conversion result mode status This bit reflects the mode of conversion for the corresponding channel. 00 Data is a result of Normal conversion mode. 01 Data is a result of Injected conversion mode. 10 Data is a result of CTU conversion mode. 11 Reserved
16:31 CDATA	Channel converted data **Note:** If MCR[WLSIDE] is set to ‘0’ than the converted data is read right aligned with lower 12/10 bits representing the actual converted data(width of actual converted data depends on selected resolution of current conversion, selectable by CTRx[CRES] bit) and remaining upper 4/6 bits are read as zeros. Whereas if the MCR[WLSIDE] is set as ‘1’ than the converted data is read left aligned with upper 12/10 bits representing the actual converted data(width of actual converted data depends on selected resolution of current conversion, selectable by CTRx[CRES] bit) and remaining lower 4/6 bits are read as zeros.

34.5.1.17 Internal Channel Watchdog Select Register *n* (ICWSELR*n*)

These registers contain WSEL_CHx[0:3] bitfields to select the threshold register which provides the values to be used for upper and lower bounds for channel x.

The Watchdog select registers to channel association is described in *Table 445*.

Offset: 0x2B0 + n*0x4 (*n* = 0 to 11) Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	WSEL_CH[(8*n)+7]				WSEL_CH[(8*n)+6]				WSEL_CH[(8*n)+5]				WSEL_CH[(8*n)+4]			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	WSEL_CH[(8*n)+3]				WSEL_CH[(8*n)+2]				WSEL_CH[(8*n)+1]				WSEL_CH[8*n]			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 357. Internal Channel Watchdog Select Register *n* (ICWSELR*n*)

Table 444. ICWSELR*n* field descriptions

Field	Description
0:3 WSEL_CH[(8*n)+7]	Watchdog select for channel x 0000 THRHLR0 register is selected. 0001 THRHLR1 register is selected. 0010 THRHLR2 register is selected. … 1110 THRHLR14 register is selected. 1111 THRHLR15 register is selected.
4:7 WSEL_CH[(8*n)+6]	Watchdog select for channel x 0000 THRHLR0 register is selected. 0001 THRHLR1 register is selected. 0010 THRHLR2 register is selected. … 1110 THRHLR14 register is selected. 1111 THRHLR15 register is selected.
8:11 WSEL_CH[(8*n)+5]	Watchdog select for channel x 0000 THRHLR0 register is selected. 0001 THRHLR1 register is selected. 0010 THRHLR2 register is selected. … 1110 THRHLR14 register is selected. 1111 THRHLR15 register is selected.
12:15 WSEL_CH[(8*n)+4]	Watchdog select for channel x 0000 THRHLR0 register is selected. 0001 THRHLR1 register is selected. 0010 THRHLR2 register is selected. … 1110 THRHLR14 register is selected. 1111 THRHLR15 register is selected.
16:19 WSEL_CH[(8*n)+3]	Watchdog select for channel x 0000 THRHLR0 register is selected. 0001 THRHLR1 register is selected. 0010 THRHLR2 register is selected. … 1110 THRHLR14 register is selected. 1111 THRHLR15 register is selected.

Table 444. ICWSELR*n* field descriptions (continued)

Field	Description
20:23 WSEL_CH[(8*n)+2]	Watchdog select for channel x 0000 THRHLR0 register is selected. 0001 THRHLR1 register is selected. 0010 THRHLR2 register is selected. … 1110 THRHLR14 register is selected. 1111 THRHLR15 register is selected.
24:27 WSEL_CH[(8*n)+1]	Watchdog select for channel x 0000 THRHLR0 register is selected. 0001 THRHLR1 register is selected. 0010 THRHLR2 register is selected. … 1110 THRHLR14 register is selected. 1111 THRHLR15 register is selected.
28:31 WSEL_CH[8*n]	Watchdog select for channel x 0000 THRHLR0 register is selected. 0001 THRHLR1 register is selected. 0010 THRHLR2 register is selected. … 1110 THRHLR14 register is selected. 1111 THRHLR15 register is selected.

Table 445. Internal Channel Watchdog Enable Registers to Channel Association

Register	Register bits 0:31
ICWSELRn (n = 0 to 11)	WSEL_CH[(8*n)+7] … WSEL_CH[8*n]

The number of bits implemented for each WSEL_CHx depends on Watchdog number:

- If Watchdog number > 8, the number of bits implemented is 4 (WSEL_CHx[3:0])
- If 4 < Watchdog number <= 8, the number of bits implemented is 3 (WSEL_CHx[3] is reserved).
- If 1 < Watchdog number <= 4, the number of bits implemented is 2 (WSEL_CHx[3:2] is reserved).
- If Watchdog number = 1, only 1 bit is present.

Each such nibble (WSEL_CHx) has positional correspondence to Internal Channel number (each channel has a 1:1 nibble correspondence), so each word ICWSELRx corresponds to eight internal Channels for a total of twelve 32-bit registers to covering all 96 test channels.

If an Internal channel is not defined, the corresponding WSEL_CHx nibble is not implemented. For the implemented nibbles, width of WSEL_CHx field within the nibble is dependent on the Watchdog number. For example, if Watchdog number is 4 than only two LSBs of each WSEL_CHx are implemented and remaining two will remain unimplemented. If both nibbles in a byte-access, all four nibbles in a halfword-access, and all eight nibbles in a word-access are unimplemented, a transfer error is generated.

34.5.1.18 Internal Channel Watchdog Enable Register *n* (ICWENR*n*)

The Watchdog enable registers to channel association is described in *Table 447*.

Offset: 0x2E0 + *n**0x4 (*n* = 0 to 2) Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	WEN_CH[x]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	WEN_CH[x]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 358. Internal Channel Watchdog Enable Register *n* (ICWENR*n*)

Table 446. ICWENRn field descriptions

Field	Description
0:31 WEN_CH[x]	Watchdog enable for channel x 0 Watchdog feature is disabled for CH[x]. 1 Watchdog feature is enabled for CH[x].

Table 447. Internal Channel Watchdog Enable Registers to Channel Association

Register	Register bits 0:31
ICWER0	WEN_CH[31:0]
ICWER1	WEN_CH[63:32]
ICWER2	WEN_CH[95:64]

Note: If channel CH[x] is not existing then the corresponding bit field WEN_CH[x] is reserved and returns 0.

If all channels CH[x] are not existing, the full register is not existing and any attempt to access this space generates a transfer error.

34.5.1.19 Internal Channel Analog Watchdog Out of Range Register *n* (ICAWORR*n*)

The analog watchdog registers to channel association is described in *Table 449*.

Offset: 0x2F0 + n*0x4 (*n* = 0 to 2) Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	AWOR_CH[x]															
W	w1c															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	AWOR_CH[x]															
W	w1c															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 359. Internal Channel Analog Watchdog Out of Range Register *n* (ICAWORR*n*)

Table 448. ICAWORR*n* field descriptions

Field	Description
0:31 AWOR_CH[x]	Analog watchdog out of range status for channel x, provided corresponding WEN_CHx bit is set. 0 CH[x] converted data is not out of range determined by its thresholds. 1 CH[x] converted data is out of range determined by its thresholds.

Table 449. Internal Channel Analog Watchdog Registers to Channel Association

Register	Register bits 0:31
ICAWORR0	AWOR_CH[31:0]
ICAWORR1	AWOR_CH[63:32]
ICAWORR2	AWOR_CH[95:64]

Note: If channel CH[x] is not existing then the corresponding bit field AWOR_CH[x] is reserved and returns 0.

If all channels CH[x] are not existing, the full register is not existing and any attempt to access this space generates a transfer error.

34.5.1.20 Test Channel Interrupt Pending Register (TCIPR)

The interrupt channel register to channel association is described in *Table 451*.

Offset: 0x400 Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	EOC_CH[x]															
W	w1c															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	EOC_CH[x]															
W	w1c															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 360. Test Channel Interrupt Pending Register (TCIPR)

Table 450. TCIPR field descriptions

Field	Description
0:31 EOC_CH[x]	End of conversion interrupt pending bit for channel x 0 End of conversion for CH[x] has not occurred. 1 End of conversion for CH[x] has occurred.

Table 451. TCIPR Register to Channel Association

Register	Register bits 0:31
TCIPR	EOC_CH[127:96]

Note: If channel CH[x] is not existing then the corresponding bit field EOC_CH[x] is reserved and returns 0.

If all channels CH[x] are not existing, the full register is not existing and any attempt to access this space generates a transfer error.

34.5.1.21 Test Channel Interrupt Mask Register (TCIMR)

The interrupt mask register to channel association is described in *Table 453*.

Offset: 0x404 Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	IM_CH[x]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	IM_CH[x]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 361. Test Channel Interrupt Mask Register (TCIMR)

Table 452. TCIMR field descriptions

Field	Description
0:31 IM_CH[x]	Interrupt mask bit for channel x 0 Interrupt for CH[x] is disabled. 1 Interrupt for CH[x] is enabled.

Table 453. Test Channel Interrupt Mask Register to Channel Association

Register	Register bits 0:31
TCIMR	IM_CH[127:96]

Note: If channel CH[x] is not existing then the corresponding bit field IM_CH[x] is reserved and returns 0.

If all channels CH[x] are not existing, the full register is not existing and any attempt to access this space generates a transfer error.

34.5.1.22 Test Channel DMA Select Register (TCDSR)

The DMA select registers to channel association is described in *Table 455*.

Offset: 0x408 Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	DS_CH[x]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	DS_CH[x]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 362. Test Channel DMA Select Register (TCDSR)

Table 454. TCDSR field descriptions

Field	Description
0:31 DS_CH[x]	DMA select for channel x 0 CH[x] is disabled to transfer data in DMA mode. 1 CH[x] is enabled to transfer data in DMA mode.

Table 455. Test Channel DMA Select Registers to Channel Association

Register	Register bits 0:31
TCDSR	DS_CH[127:96]

Note: If channel CH[x] is not existing then the corresponding bit field DS_CH[x] is reserved and returns 0.

If all channels CH[x] are not existing, the full register is not existing and any attempt to access this space generates a transfer error.

34.5.1.23 Test Channel Normal Conversion Mask Register (TCNCMR)

The normal conversion mask register to channel association is described in *Table 457*.

Offset: 0x40C Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	NCE_CH[x]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	NCE_CH[x]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 363. Test Channel Normal Conversion Mask Register (TCNCMR)

Table 456. TCNCMR field descriptions

Field	Description
0:31 NCE_CH[x]	Normal conversion enable for channel x 0 Normal conversion is disabled for CH[x]. 1 Normal conversion is enabled for CH[x].

Table 457. Test Channel Normal Conversion Mask Register to Channel Association

Register	Register bits 0:31
TCNCMR	NCE_CH[127:96]

Note: If channel CH[x] is not existing then the corresponding bit field NCE_CH[x] is reserved and returns 0.

If all channels CH[x] are not existing, the full register is not existing and any attempt to access this space generates a transfer error.

34.5.1.24 Test Channel Injected Conversion Mask Register (TCJCMR)

The injected conversion mask register to channel association is described in *Table 459*.

Offset: 0x410 Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	JCE_CH[x]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	JCE_CH[x]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 364. Test Channel Injected Conversion Mask Register (TCJCMR)

Table 458. TCJCMR field descriptions

Field	Description
0:31 JCE_CH[x]	JCE_CH[x]: Injected conversion enable for channel x 0 Injected conversion is disabled for CH[x]. 1 Injected conversion is enabled for CH[x].

Table 459. Test Channel Normal Conversion Mask Register to Channel Association

Register	Register bits 0:31
TCJCMR	JCE_CH[127:96]

Note: If channel CH[x] is not existing then the corresponding bit field JCE_CH[x] is reserved and returns 0.

If all channels CH[x] are not existing, the full register is not existing and any attempt to access this space generates a transfer error.

34.5.1.25 Test Channel Watchdog Select Register *n* (TCWSELR*n*)

This register contains WSEL_CHx[0:3] bitfields to select the threshold register which provides the values to be used for upper and lower bounds for channel x.

The Watchdog select register to channel association is described in *Table 461*.

Offset: 0x414 + *n**0x4 (*n* = 0 to 3) Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	WSEL_CH[(8*n)+103]				WSEL_CH[(8*n)+102]				WSEL_CH[(8*n)+101]				WSEL_CH[(8*n)+100]			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	WSEL_CH[(8*n)+99]				WSEL_CH[(8*n)+98]				WSEL_CH[(8*n)+97]				WSEL_CH[(8*n)+96]			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 365. Test Channels Watchdog Select Register *n* (TCWSELR*n*)

Table 460. TCWSELR*n* field descriptions

Field	Description
4:7 WSEL_CH[(8*n)+103]	Watchdog select for channel x 0000 THRHLR0 register is selected. 0001 THRHLR1 register is selected. 0010 THRHLR2 register is selected. … 1110 THRHLR14 register is selected. 1111 THRHLR15 register is selected.
0:3 WSEL_CH[(8*n)+102]	Watchdog select for channel x 0000 THRHLR0 register is selected. 0001 THRHLR1 register is selected. 0010 THRHLR2 register is selected. … 1110 THRHLR14 register is selected. 1111 THRHLR15 register is selected.
8:11 WSEL_CH[(8*n)+101]	Watchdog select for channel x 0000 THRHLR0 register is selected. 0001 THRHLR1 register is selected. 0010 THRHLR2 register is selected. … 1110 THRHLR14 register is selected. 1111 THRHLR15 register is selected.
12:15 WSEL_CH[(8*n)+100]	Watchdog select for channel x 0000 THRHLR0 register is selected. 0001 THRHLR1 register is selected. 0010 THRHLR2 register is selected. … 1110 THRHLR14 register is selected. 1111 THRHLR15 register is selected.
16:19 WSEL_CH[(8*n)+99]	Watchdog select for channel x 0000 THRHLR0 register is selected. 0001 THRHLR1 register is selected. 0010 THRHLR2 register is selected. … 1110 THRHLR14 register is selected. 1111 THRHLR15 register is selected.
20:23 WSEL_CH[(8*n)+98]	Watchdog select for channel x 0000 THRHLR0 register is selected. 0001 THRHLR1 register is selected. 0010 THRHLR2 register is selected. … 1110 THRHLR14 register is selected. 1111 THRHLR15 register is selected.

Table 460. TCWSELR*n* field descriptions (continued)

Field	Description
24:27 WSEL_CH[(8*n)+97]	Watchdog select for channel x 0000 THRHLR0 register is selected. 0001 THRHLR1 register is selected. 0010 THRHLR2 register is selected. … 1110 THRHLR14 register is selected. 1111 THRHLR15 register is selected.
28:31 WSEL_CH[(8*n)+96]	Watchdog select for channel x 0000 THRHLR0 register is selected. 0001 THRHLR1 register is selected. 0010 THRHLR2 register is selected. … 1110 THRHLR14 register is selected. 1111 THRHLR15 register is selected.

Table 461. Test Channel Watchdog Select Registers to Channel Association

Register	Register bits 0:31
TCWSELR0	WSEL_CH[103] … WSEL_CH[96]
TCWSELR1	WSEL_CH[111] … WSEL_CH[104]
TCWSELR2	WSEL_CH[119] … WSEL_CH[112]
TCWSELR3	WSEL_CH[127] … WSEL_CH[120]

The number of bits implemented for each WSEL_CHx depends on Watchdog number:

- If Watchdog number > 8, the number of bits implemented is 4 (WSEL_CHx[3:0])
- If 4 < Watchdog number <= 8, the number of bits implemented is 3 (WSEL_CHx[3] is reserved).
- If 1 < Watchdog number <= 4, the number of bits implemented is 2 (WSEL_CHx[3:2] is reserved).
- If Watchdog number = 1, only 1 bit is present.

Each such nibble (WSEL_CHx) has positional correspondence to Test Channel number (each channel has a 1:1 nibble correspondence), so each word TCWSELRx corresponds to eight test Channels, thus requiring total four 32-bit registers to cover all 32 test channels.

If a Test channel is not defined, the corresponding WSEL_CHx nibble is not implemented. For the implemented nibbles, width of WSEL_CHx field within the nibble is dependent on the Watchdog number. For example, if Watchdog number is 4 than only two LSBs of each WSEL_CHx are implemented and remaining two will remain unimplemented. If both nibbles in a byte-access, all four nibbles in a halfword-access, and all eight nibbles in a word-access are unimplemented, a transfer error is generated.

34.5.1.26 Test Channel Watchdog Enable Register (TCWENR)

The Watchdog enable registers to channel association is described in *Table 463*.

Offset: 0x424 Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	WEN_CH[x]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	WEN_CH[x]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 366. Test Channel Watchdog Enable Register (TCWENR)

Table 462. TCWENR field descriptions

Field	Description
0:31 WEN_CH[x]	Watchdog enable for channel x 0 Watchdog feature is disabled for CH[x]. 1 Watchdog feature is enabled for CH[x].

Table 463. Test Channel Watchdog Enable Registers to Channel Association

Register	Register bits 0:31
TCWENR	WEN_CH[127:96]

Note: If channel CH[x] is not existing then the corresponding bit field WEN_CH[x] is reserved and returns 0.

If all channels CH[x] are not existing, the full register is not existing and any attempt to access this space generates a transfer error.

34.5.1.27 Test Channel Analog Watchdog Out of Range Register (TCAWORR)

The analog watchdog registers to channel association is described in *Table 465*.

Offset: 0x0428 Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	AWOR_CH[x]															
W	w1c															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	AWOR_CH[x]															
W	w1c															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 367. Test Channel Analog Watchdog Out of Range Register (TCAWORR)

Table 464. TCAWORR field descriptions

Field	Description
0:31 AWOR_CH[x]	Analog watchdog out of range status for channel x, provided corresponding WEN_CHx bit is set. 0 CH[x] converted data is not out of range determined by its thresholds. 1 CH[x] converted data is out of range determined by its thresholds.

Table 465. Analog Watchdog Register to Channel Association

Register	Register bits 0:31
TCAWORR	AWOR_CH[127:96]

Note: If channel CH[x] is not existing then the corresponding bit field AWOR_CH[x] is reserved and returns 0.

If all channels CH[x] are not existing, the full register is not existing and any attempt to access this space generates a transfer error.

34.5.1.28 Test Channel Connection with Analog Pin Register *n* (TCCAPR*n*)

Test channels connected through a 20 kΩ source impedance can be shorted with any internal analog channel using the configuration programmed in these registers.

Test channel short registers to channel association is described in *Table 466*.

Offset: 0x430 + *n**0x4 (*n* = 0 to 7) Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	ESIC_TCH[4*n+3]	ICSEL_TCH[4*n+3]							ESIC_TCH[4*n+2]	ICSEL_TCH[4*n+2]						
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	ESIC_TCH[4*n+1]	ICSEL_TCH[4*n+1]							ESIC_TCH[4*n]	ICSEL_TCH[4*n]						
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 368. Test Channel Connection with Analog Pin Register *n* (TCCAPR*n*)

Table 466. TCCAPR*n* field descriptions

Field	Description
0 ESIC_TCH[4*n+3]	Enable short with internal channel for test channel x=4*n+3 0 Test channel x cannot be shorted with internal channel. 1 Test channel x can be shorted with internal channel.
1:7 ICSEL_TCH[4*n+3]	Internal channel selection for short with test channel x=4*n+3
8 ESIC_TCH[4*n+2]	Enable short with internal channel for test channel x=4*n+2 0 Test channel x cannot be shorted with internal channel. 1 Test channel x can be shorted with internal channel.
9:15 ICSEL_TCH[4*n+2]	Internal channel selection for short with test channel x=4*n+2
16 ESIC_TCH[4*n+1]	Enable short with internal channel for test channel x=4*n+1 0 Test channel x cannot be shorted with internal channel. 1 Test channel x can be shorted with internal channel.
17:23 ICSEL_TCH[4*n+1]	Internal channel selection for short with test channel x=4*n+1
24 ESIC_TCH[4*n]	Enable short with internal channel for test channel x=4*n 0 Test channel x cannot be shorted with internal channel. 1 Test channel x can be shorted with internal channel.
25:31 ICSEL_TCH[4*n]	Internal channel selection for short with test channel x=4*n

Table 467. Test Channel Short Registers to Channel Association

Register	Register bits 0:31 (arrangement of bit fields left to right)
TCCAPR0	ESIC_TCH3, ICSEL_TCH3, ESIC_TCH2, ICSEL_TCH2, ESIC_TCH1, ICSEL_TCH1, ESIC_TCH0, ICSEL_TCH0
TCCAPR1	ESIC_TCH7, ICSEL_TCH7, ESIC_TCH6, ICSEL_TCH6, ESIC_TCH5, ICSEL_TCH5, ESIC_TCH4, ICSEL_TCH4
TCCAPR2	ESIC_TCH11, ICSEL_TCH11, ESIC_TCH10, ICSEL_TCH10, ESIC_TCH9, ICSEL_TCH9, ESIC_TCH8, ICSEL_TCH8
TCCAPR3	ESIC_TCH15, ICSEL_TCH15, ESIC_TCH14, ICSEL_TCH14, ESIC_TCH13, ICSEL_TCH13, ESIC_TCH12, ICSEL_TCH12
TCCAPR4	ESIC_TCH19, ICSEL_TCH19, ESIC_TCH18, ICSEL_TCH18, ESIC_TCH17, ICSEL_TCH17, ESIC_TCH16, ICSEL_TCH16
TCCAPR5	ESIC_TCH23, ICSEL_TCH23, ESIC_TCH22, ICSEL_TCH22, ESIC_TCH21, ICSEL_TCH21, ESIC_TCH20, ICSEL_TCH20
TCCAPR6	ESIC_TCH27, ICSEL_TCH27, ESIC_TCH26, ICSEL_TCH26, ESIC_TCH25, ICSEL_TCH25, ESIC_TCH24, ICSEL_TCH24
TCCAPR7	ESIC_TCH31, ICSEL_TCH31, ESIC_TCH30, ICSEL_TCH30, ESIC_TCH29, ICSEL_TCH29, ESIC_TCH28, ICSEL_TCH28

34.5.1.29 Test Channel Data Register *n* (TCDR*n*)

The conversion results for each channel is loaded into data registers. Each data register contains:

- The conversion result
- Status information related to ADC mode,
- Data valid
- Control information to select the required reference voltage
- Timing parameter selection for each channel.

These registers are accessed 32-bit R/W.

Offset: 0x2D0 + *n**0x4 (*n* = 96 to 127) Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	PCE	0	CTSEL		0	0	0	FCERR	VALID	OVERW	RESULT	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	CDATA															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 369. Test Channel Data Register *n* (TCDR*n*)

Table 468. TCDR*n* field descriptions

Field	Description
4 PCE	Precharge Enable This bit enables the precharging phase during channel conversion. 0 Precharge phase is disabled 1 Precharge phase is enabled
6:7 CTSEL	This bitfield selects the conversion timing register for each channel to select different precharge and sampling phase durations. 00 CTR0 is selected 01 CTR1 is selected 10 CTR2 is selected 11 CTR3 is selected
11 FCERR	Fast Comparator Error Flag This bit is used to notify error condition on current fast comparator operation inside ADC analog block. It is automatically cleared when data is read. 0 Cleared by reading the Converted by software 1 Fast Comparator Error Condition has occured in current conversion

Table 468. TCDR*n* field descriptions (continued)

Field	Description
12 VALID	Data valid flag This bit is used to notify when the data is valid (a new value has been written). It is automatically cleared when data is read. 0 Converted data has been read by software 1 Converted data is valid and has not been read yet
13 OVERW	Data overwritten flag This bit signals that the previous converted data has been overwritten by a new conversion. This functionality depends on the value of MCR[OWREN]: – When OWREN = 0, OVERW is frozen to 0 and CDATA field is protected against being overwritten until being read. – When OWREN = 1, OVERW flags the CDATA field overwrite status. 0 Converted data has not been overwritten 1 Previous converted data has been overwritten before having been read
14:15 RESULT	Conversion result mode status This bit reflects the mode of conversion for the corresponding channel. 00 Data is a result of Normal conversion mode 01 Data is a result of Injected conversion mode 10 Data is a result of CTU conversion mode 11 Reserved
16:31 CDATA	Channel converted data **Note:** If MCR[WLSIDE] is set to ‘0’ than the converted data is read right aligned with lower 12/10 bits representing the actual converted data(width of actual converted data depends on selected resolution of current conversion, selectable by CTRx[CRES] bit) and remaining upper 4/6 bits are read as zeros. Whereas if the MCR[WLSIDE] is set as ‘1’ than the converted data is read left aligned with upper 12/10 bits representing the actual converted data(width of actual converted data depends on selected resolution of current conversion, selectable by CTRx[CRES] bit) and remaining lower 4/6 bits are read as zeros.

34.6 Start of conversion pulse delay

The following table lists the delays between various trigger points (initiation request for conversion as registered at system clock) and the actual issue of start pulse (adc_start[l]) to the ADC hard-macrocell. Here, each possible trigger source is described with the number of clock cycles required for different operations until the trigger of adc_start. This table does not take into account any software delay involved in programming or polling of any bit in order to measure conversion timing.

l. adc_start is the start of conversion indication to ADC hard-macrocell to start the conversion.

Table 469. Start of conversion pulse delay table

Trigger source	Register control (PBRIDGE_ CLK)	Inter domain synchronization (PBRIDGE_CLK)	ADC state machine (SARADC_ CLK)	Inter domain synchronization (SARADC_CLK)	Comments
Normal conversion triggered by software (setting of NSTART of MCR)	1	—	2	1-2[3]	Cumulative delay from wen(byte_en) assertion to actual issue of adc_start pulse = 3-4 SARADC_CLK cycles + 1 PBRIDGE_CLK cycle.
Normal conversion triggered by external trigger (HW trigger through)	—	—	2	1–2[3]	Cumulative delay from the triggering edge (rising or falling) to the actual issue of adc_start pulse = 3–4 SARADC_CLK cycles.
Normal conversion within a chain	—	—	0 (from EOC[1] of previous conversion)	—	Delay from the EOC[1] of last conversion, The start of next conversion is issued exactly on the same cycle when EOC[1] is expected for previous conversion.
Injected conversion triggered by software (setting of JSTART bit of MCR)	1	—	3	1–2[3]	Cumulative delay from wen(byte_en) assertion. Here delay for first start pulse of the injected chain execution when FSM was originally in IDLE state is considered = 4–5 SARADC_CLK cycles + 1 PBRIDGE_CLK cycle.
Injected conversion triggered by external Trigger (HW trigger)	—	—	3	1–2[3]	Cumulative delay from triggering edge (rising or falling) to actual issue of adc_start = 4–5 SARADC_CLK cycles.
Injected conversion within a chain (delay between consecutive conversion of injected chain)	—	—	0 (from EOC[1] of previous conversion)	—	Delay from EOC[1] of last injected channel and adc_start pulse of next injected channel. The start of next conversion is issued exactly on the same cycle when EOC[1] is expected for previous conversion

Table 469. Start of conversion pulse delay table (continued)

Trigger source	Register control (PBRIDGE_CLK)	Inter domain synchronization (PBRIDGE_CLK)	ADC state machine (SARADC_CLK)	Inter domain synchronization (SARADC_CLK)	Comments
Injected conversion over normal conversion (software injected conversion chain)	1	1–2(3)	3–5(2)	1–2(3)	Cumulative delay from wen(byte_en) assertion to actual trigger of first injected channel, when the FSM was in conversion cycle of previous normal conversion = 4–7 SARADC_CLK cycles + 2-3 PBRIDGE_CLK cycles.
Injection of conversion over normal conversion in JTRGSEQ mode	—	—	3	1–2(3)	Cumulative delay from triggering edge (rising or falling) to actual issue of adc_start = 4–5 SARADC_CLK cycles.
Abort of a Normal/Injected Channel conversion during Normal Chain/Injected Chain execution (setting of MCR[ABORT] bit)	1	1–2(3)	3–5(2)	1–2(3)	Delay from writing of ABORT bit (wen/byte_en assertion) to actual issue of adc_start pulse for the next channel conversion = 4–7 SARADC_CLK cycles + 2–3 PBRIDGE_CLK cycles.

1. EOC is the end of conversion from ADC hard macrocell
2. The above delay values are calculated over a range of system and ADC clock frequencies. With PBRIDGE_CLK(max) = 50 MHz to PBRIDGE_CLK(min) = 40 MHz and SARADC_CLK(max) = 16 MHz to SARADC_CLK(min) = 14 MHz. L.H.S value corresponds to max frequencies and RHS corresponds to min frequencies.
3. The above variation is owing to inter domain synchronization.

34.7 Initialization Information

To initialize the SARADC registers for data conversion, the following programming sequence is required:-

1. Program the PDEDR register with appropriate power down exit delay value and required CTU mode (MCR[CTU_MODE]) in case CTU mode has to be entered later on by setting CTUEN bit.
2. Exit the power down mode by clearing the MCR[PWDN] bit. After the power down mode is exited the SARADC enters the IDLE state, this can be checked by polling for MSR[ADCSTATUS] bits. Note that steps 3-7 can also be executed before step 2.
3. Program the required conversion mode and control options for example one-shot mode or scan mode, hardware trigger enable and selection for normal or injected conversions, hardware trigger edge select, owerwrite enable e.t.c. in MCR register.
4. Program the conversion mask registers({I,T}C{N,J}CMR) for channels to be converted. If required, enable and select watchdog monitors({I,T}CWENR, {I,T}CWSELR) as applicable for each channel, program the watchdog threshold values(WTHRHLR) for each watchdog monitor.
5. If required, Enable and select the , Test to internal channel mapping by programming the TCCAPR registers.
6. Program the required Interrupt/DMA mask registers(DMAE, {I,T}CDSR, IMR, {I,T}C{N,J}MR, WTIMR) for each of the selected channels.
7. Program the Conversion timing registers(CTR0-3) for the required conversion durations. Enable the Precharge phase, reference selection, CTR selection per channel basis by programming the {I,T}CDR registers.
8. To initiate CTU mode conversion enter the selected CTU mode by setting the MCR[CTUEN] bit and program the CTU module accordingly to start the conversions. To initiate Normal/Injected mode conversions by software set the MCR[NSTART/JSTART] bit. To initiate hardware triggered Normal/Injected conversions program the external trigger module to provide the required triggers.
9. Read the conversion data and status in {I,T}CDR registers.

35 Enhanced Cross Trigger Unit (ECTU)

35.1 Introduction

The ECTU (enhanced cross trigger unit) is used to provide commands to the ADCs, start/stop trigger signals to other IPs and external trigger to the pad. This IP, if correctly programmed, provide new commands to the ADC also if the CPU is involved in other operations. It contains a number of FIFOs inside that are used to store converted ADC data and guarantee that no data will be lost.

35.2 Main features

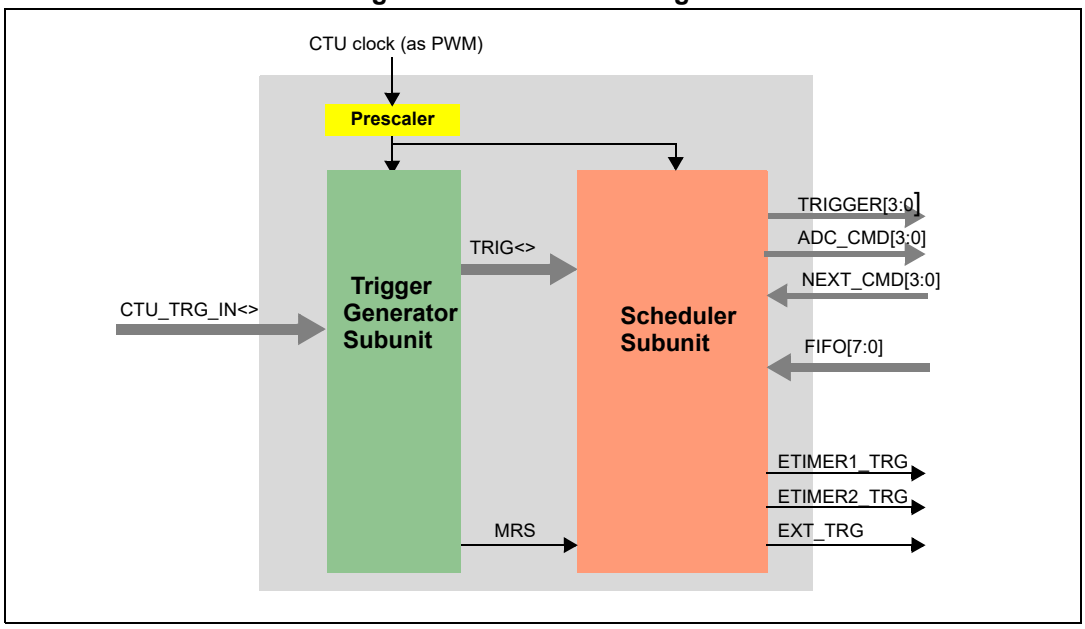
- Receives up to 8 input trigger events from other IPs such as PWM, Timers, external pins . For mapping details, refer to *Figure 33: Module to module connections* and *Table 13: Peripheral multiplexing table description*
- Generates up to 8 internal events to trigger ADC sequences
- Support of 4 ADCs
- Each internal trigger event can generate ADC commands to start conversion
- Support of single-mode conversion and multi-mode conversion
- 8 FIFOs to store converted ADC data
- Up to 64 ADC command registers to support various conversion sequences
- Motor function iMonitor using 32-bit CRC

35.3 Block diagram

The block diagram is composed of the following:

1. TGS (trigger generator subunit)
2. SU (scheduler subunit)

Figure 370. CTU block diagram



Input signal are latched inside TGS to generate up to 8 trigger events. These event go into SU and after a decode phase, generate the corresponding outputs for ADCs/external triggers.

35.4 Functional description

35.4.1 Trigger generator subunit (TGS)

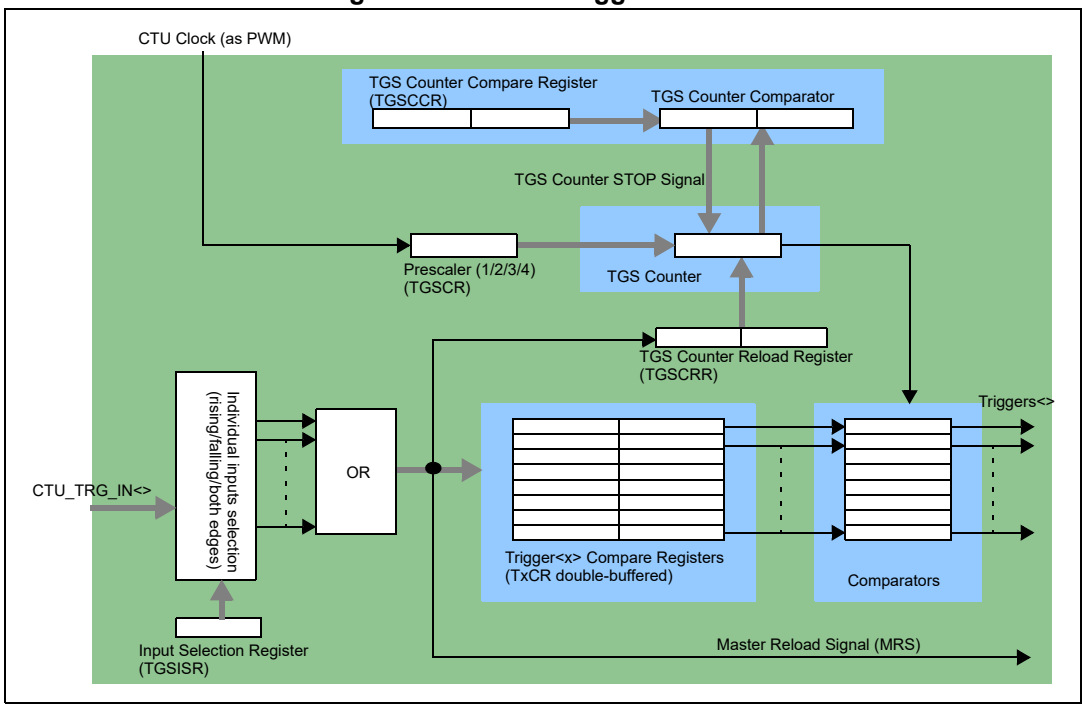
This part of the IP is used to generate up to 8 trigger events from a combination of input signals. The output could be generated in two different modes:

1. Triggered mode: each programmed event generated from the incoming signal can generate up to 8 trigger events.
2. Sequential mode: each trigger is generated from an incoming event. Each new event selects the respective TxCR register for the comparison.

35.4.1.1 TGS in triggered mode

The TGS mode is selected using TGSCR bit in the TGS control register.

Figure 371. TGS in triggered mode



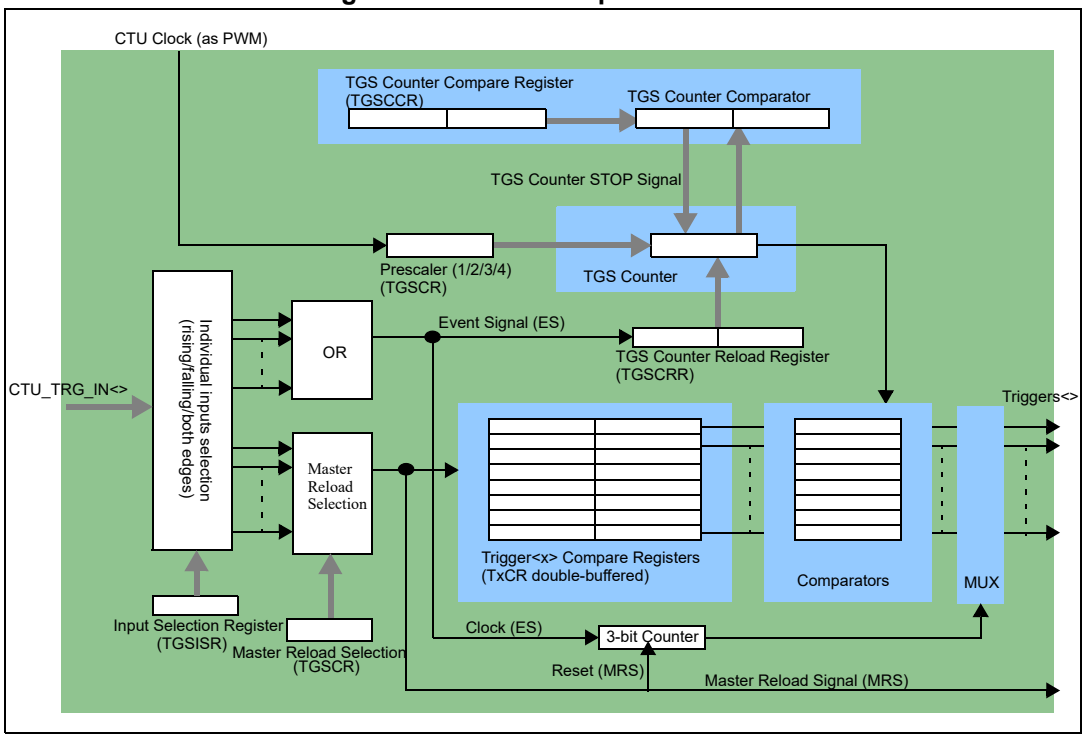
Each input signal is selected from the Input selection register (TGSISR) and is OR-ed with the others selected in order to generate the master reload signal (MRS). This signal is then phase-aligned with the ADC clock to ensure triggers to ADCs are generated always in the same manner. This signal is used to reload the triggers list registers and the TGS counter reload register (TGSCRR). In this way the values written in the trigger list registers are used only when a MRS occurs.

The TGS counter register is used to compare the counter value with the current value of trigger list register, and when the two value are the same an event is generated. A TGS counter STOP signal stops the counter if the value of TGS counter is equal to TGS counter comparator.

The master reload signal can also be set to ‘1’ by software setting the MRS_SG bit in the CTU control register.

A prescaler is used to divide the MC clock in order to increase the time before TGS counter STOP signal goes to one.

Figure 372. TGS in sequential mode



This mode of operation is different from the previous one because the output triggers are generated one at a time in a sequential mode addressed from the 3-bit counter.

The combination of the inputs is used to generate the event signal (ES). The ES signal reloads the TGS counter reload register (TGSCRR) and is used to increase the counter value of the 3-bit counter. The MRS signal otherwise is generated from a single active signal addressed from the Master reload selection (MRS_SM), resets the 3-bit counter and reloads the triggers list registers.

A signal is activated from the Input selection register, for this reason in sequential mode a MRS is a particular EV signal.

The meaning of the other registers are the same as in the previous mode.

35.4.2 Digital filter

External signals which can generate CTU triggers can be asynchronous with motor control clock. For this reason two flip-flops are provided to synchronize the signals. Additionally there is a digital filter that is programmable using the CTU_FILTER register. The external signal is considered at ‘1’ if it is latched <CTU_FILTER.Filter_value>+1 times as ‘1’ and is considered at ‘0’ if it is latched <CTU_FILTER.Filter_value>+1 times as ‘0’.

Due to the uncertainty of synchronizers, all pulses less than <CTU_FILTER.Filter_value> cycles of MT_clk will always be filtered, and all pulses higher than <CTU_FILTER.Filter_value> cycles of MT_clk will always be latched. The pulses in between remain in an uncertain domain that is they might or might not be filtered.

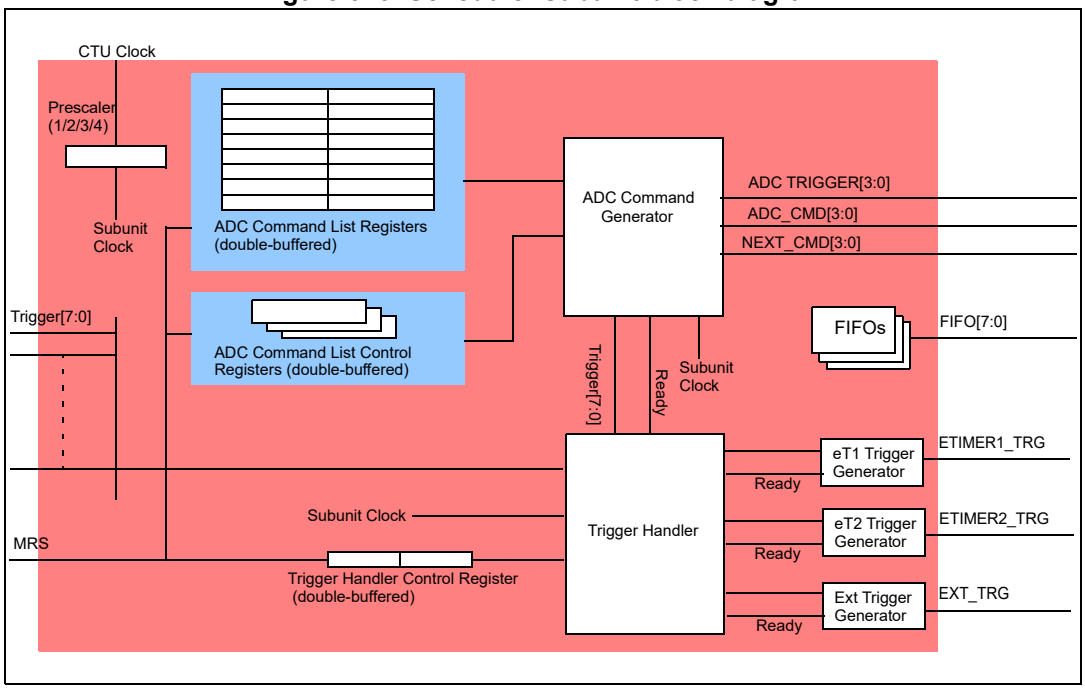
The register is correctly synchronized when the **Filter_en** bit in the control register is set. CTU_FILTER register must always be programmed before setting **Filter_en** bit.

The filter is present for all CTU_TRG_IN[15:0] and is controlled using a static design parameter CTU_TRG_FILTER_EN. For this module, the value of CTU_TRG_FILTER_EN[15:0] is 16'b1000_0000_0000_0000. Filters to all trigger inputs are enabled with a common control bit CTU_CR.Filter_en.

35.4.3 Scheduler subunit (SU)

The SU has the same functionality when TGS is in trigger or in sequential mode.

Figure 373. Scheduler subunit block diagram



Each input trigger event can be linked to any of 8 trigger event by the Trigger handler subunit. The outputs generated are:

1. ADC command
2. eTimer1
3. eTimer2
4. External trigger

The linking is programmed using THCRx registers in the Trigger handler control and it is possible to disable the output using the respective THCRx register bit.

If two events at the same time are linked to the same output only one output is generated and an error (TGS_OE) is provided. The output is generated using the trigger with the lowest index. For example if trigger 0 and trigger 1 both are linked to the ADC output and they occur together, an error is generated and the output linked with the trigger 0 is generated.

In case a trigger is linked to a timer or to the external trigger, a pulse with active rising edge is generated. Additional feature for external trigger is available: the external trigger output has the Pulse mode and the Toggle mode (1 bit for each trigger event in the TGSCR). In toggle mode, on each trigger event linked to the external trigger, the external trigger pin toggles.

The ON-Time of the triggers is defined from a COTR register (on time register). A guard time is also defined from the same register at the same value of the ON-Time. A new trigger will be generated only if the ON time + Guard time is past. The ON-Time and Guard time are used only for external trigger.

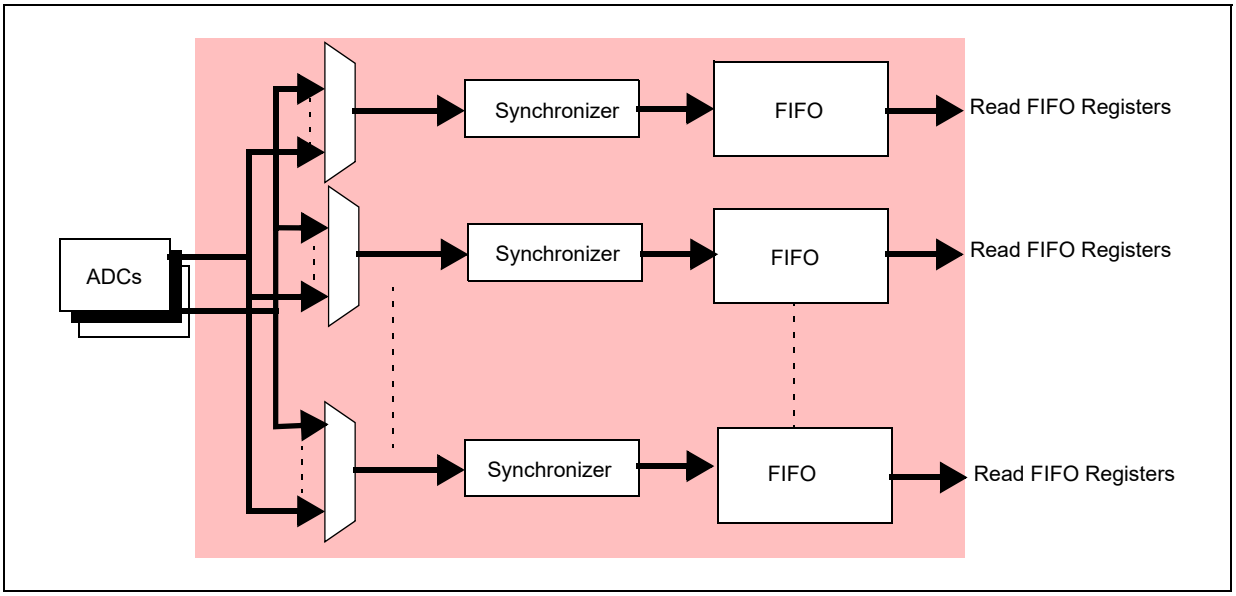
Each trigger index is mapped to ADC Commands (Command list register CLRx) using Command list control registers CLCRx. If a field of CLCRx contains the integer “n” the correspondent command register is the CLR(n+1).

The first time that an event linked to the ADC command is activated, the first command is generated according to the CLCRx register (*Figure 375*).

35.4.4 CTU FIFO

Below are the details about the FIFOs present in the CTU.

Figure 374. FIFO block diagram



FIFOs in the CTU receive data from ADCs at the end of conversion and put it into the right FIFO. The target FIFO information is embedded in the ADC command register. A FIFO bus select signal coming from the CTU command generator logic selects the right FIFO for the write operations. The same protocol is implemented for all ADCs. It is possible that more than one ADC command selects the same FIFO for storing converted data. In this case if ADC data are simultaneouly available then the writing order is selected with the lower indexed ADC command, that is the data of command<n> ADC will be written first then the data of command<n+1> and so on. As the ADC unit number is also stored in FIFOs along with data, the sequence of storing does not matter.

The data from ADC needs to be synchronized as FIFO read occurs at ipg_clk and ADC operates at MT_clk. The time for a correct synchronization is equal to 4 ipg_clk + 4 Motor control clock. If another data from the same ADC arrives into this time slices it is filtered. For this reason if for example the ipg_clk is at 64MHz the motor control clock cannot be less than about 8 Mhz. Otherwise another push that come before 650ns will be filtered.

When a read operation is started, from the CPU or from the DMA, the two fields of data are stored on the bus in a different format as per the description of registers FRx or FLx. Setting

the corresponding bits on the control register is possible to enable interrupt or enable the DMA interface. Three signals (DMA_ACK, DMA_REQ, DMA_DONE) define the DMA interface protocol. A DMA_REQ is activated if the number of elements in the FIFO is bigger than the threshold (overflow condition). When the request is accepted from the DMA (DMA_ACK = 1) the DMA_REQ signal is set to zero. A new request can be performed only after the DMA_done signal is set. Read operations are synchronous to the system clock, write operations instead are synchronous with the CTU clock.

Following flags report the status of the FIFOx

1. **Empty**. This bit is 1 if the FIFO is empty and is 0 if the FIFO is not empty.
2. **Full**. This bit is 1 if the FIFO is full and 0 if it is not full.
3. **Overflow**. This bit is 1 if the number of elements in the FIFO is bigger than the threshold.
4. **Overrun**. This is the only error bit and is 1 if a write in a FIFO occurred when FIFO is FULL after the last clear. It is 0 if, since last clear of this bit, never happens that a overrun condition occurred.

CTU IP contains 8 FIFOs. The depth of the FIFOs are as below:

Table 470. CTU FIFO depths

FIFO #	Depth (number of words)
0	8
1	8
2	8
3	8
4	8
5	8
6	8
7	8

35.5 CTU interrupt and DMA requests

35.5.1 DMA support

The DMA can be used to configure the CTU registers. One DMA channel is reserved for performing a block transfer and the MRS can be used as an optional DMA request signal (MRS_DMAE bit in the CTU Interrupt/DMA Register). Moreover, this CTU implementation requires DMA support for reading the data from the FIFOs. One DMA channel is available for each FIFO.

Each MRS, if MRS_DMAE bit is enabled, generates a DMA request. This request will be set to zero only if DMA acknowledge (ipd_ack) signal is one. An error is generated if the DMA is writing when another MRS occurs. To guarantee that the data will be stable the DMA must set GRE to ‘1’ (if a write operation occurred) and TGSISR_RE (only if TGSISR is changed) bit at the end of operations. A ipg_dma_done will be used like a set operation in the GRE bit if the DMA_DE bit in the CTUIR register is set. Like for any other access when the GRE is at one any other access generates an error (MRS_RE). A new access can be performed only

when a master reload occurs and reset the GRE bit. If an error occurs no DMA request will be provided.

35.5.2 CTU faults and errors

In this paragraph we talk about possible error and interrupt that can happen during the programming.

1. If a MRS occurs while user is updating the double-buffered registers the MRS_RE is set.
2. In sequential mode a 3-bit counter selects the right trigger event. Every time that ES has positive edge the counter is incremented by one. The MRS reloads this counter. If the number of positive edge of ES is bigger than eight an error occurs and SM_TO is set to ‘1’.
3. The ADC command is composed of six fields (ADC number, Channel number, FIFO number, Last command of sequence, Last command, Command interrupt request). If these fields are not correct for the ADC an error occurs and ICE flag is set to ‘1’. ICE flag is also set in following cases:
   a) When same ADC is used more than once in a concurrent command sequence. For example both CLR1 and CLR2 are programmed for ADC_0 conversion and only CLR2.EC bit is set. In such cases conversion takes place as according to lowest index of CLRx registers for example CLR1 will be used to select FIFO/CH_NUM etc for ADC_0.
   b) When LC bit is set but EC bit is not set (this situation can also result in MAX_CON interrupt). In this case conversion does not take place and this is considered as the last command.
4. If the EC bit is not found to be set within the maximum possible 4 commands then MAX_CON flag is set. This signifies that maximum concurrent conversion limit is not followed. When this error happens, no conversion takes place and this condition is considered as the last command. ECTU then waits for next trigger source.
5. If the time window defined from two consecutive MRS is not enough for the number of programmed events in the THCR registers, an error occurs and MRS_O flag is set.
6. TGS Overrun in Sequential Mode: new incoming EV occurs before that the trigger event selected by the previous EV occurs. The error is generated only if the trigger from the TGS is linked to some output of the SU.
7. A trigger event occurs during the time when the actions of the previous trigger event are not completed (user ensures no trigger event occurs during another one is processed, but if user mistakes and a trigger event occurs during another one is processed, the incoming trigger event will be lost and an error occurs). Another overrun error is possible if two trigger event coming from the TGS are linked to the same output trigger in the same time. In this case an overrun error is generated and the output related with the lowest trigger is generated. There are 4 overrun flags (T1_OE, T2_OE, ET_OE, ADC_OE). The Trigger Handler, when a trigger event occurs and the corresponding Ready signal is high, presents the respective trigger signal (one cycle high time + one cycle low time) to the respective generator sub-block (ADC Command Generator, or Ext. Trigger Generator). This generator sub-block then generates the requested signal. Until this real signal is generated (including guard time) the Ready signal is kept low. In case of ADC Command Generator, the Ready signal shall be kept low until last conversion in the batch is finished. The respective overrun flag is set at the following conditions: (a) Ready signal is low and (b) the rising edge of the respective trigger signal (from Trigger Handler to generator sub-block) occurs. This

architecture allows user to pre-set, for example, a trigger to the eTimer1 in the middle of an ADC conversion, that is the SU will be considered busy only if a request to perform the same action that the SU is already performing occurs.

8. The ADC commands has a bit for interrupt request. If this bit is set to 1 each new ADC command set the ADC_i flag at the start of the conversion. This flag is set only if CIR bit of the last command register is set for any concurrent conversion that is the EC bit must be 1 in that register.
9. If the internal counter (TGS counter) reaches the value inside the TGSCCR an error is generated (Error counter).
10. If the conversion time is not in the specified range an error is generated (SAF_ERRx bits in CTUEFR).
11. An interrupt could be generated also if MRS or a trigger occurs and the correspondent enable bit is set.
12. If the programmed CRC is not matching with the computed CRC at the MRS event, an error is generated (CRC_ERR bit in CTUEFR).
13. Others interrupt are due to the FIFO.

Table 471. CTU interrupt flags

Interrupt	Interrupt function
MRS_RE	Master Reload Signal Reload Error
SM_TO	Trigger Overrun (more than 8 EV) in TGS Sequential Mode
CRC_ERR	CRC comparison Error
ICE	Invalid Command Error
MAX_CON	Maximum concurrent command Error
MRS_O	Master Reload Signal Overrun
TGS_OSM	TGS Overrun in Sequential Mode
ADC_OE	ADC command generation Overrun Error
T1_OE	Timer 1 trigger generation Overrun Error
T2_OE	Timer 2 trigger generation Overrun Error
ET_OE	External Trigger generation Overrun Error
ADC_I	ADC command interrupt flag
MRS_I	MRS Interrupt flag
T0_I	Trigger 0 interrupt flag
T1_I	Trigger 1 interrupt flag
T2_I	Trigger 2 interrupt flag
T3_I	Trigger 3 interrupt flag
T4_I	Trigger 4 interrupt flag
T5_I	Trigger 5 interrupt flag
T6_I	Trigger 6 interrupt flag
T7_I	Trigger 7 interrupt flag
SAF_ERR4	Conversion time error flag for 4th concurrent conversion

Table 471. CTU interrupt flags (continued)

Interrupt	Interrupt function
SAF_ERR3	Conversion time error flag for 3th concurrent conversion
SAF_ERR2	Conversion time error flag for 2nd concurrent conversion
SAF_ERR1	Conversion time error flag for 1st concurrent conversion
FIFO interrupt flags (i = 1 to 8)	
FIFO_FULL[i]	This bit is set to 1 if the FIFO i is full
FIFO_EMPTY[i]	This bit is set to 1 if the FIFO i is empty
FIFO_OVERFLOW[i]	This bit is set to 1 if the number of words became higher than the value set in the threshold i
FIFO_OVERRUN[i]	This bit is set to 1 if a write operation occurs when corresponding FIFO_FULL[i] flag is set

In order to reduce the number of interrupt lines, some interrupts are combined together to generate single interrupt.

- IEE interrupt line is ORed of MRS_RE, SM_TO, ICE, MRS_O, TGS_OSM, ADC_OE, T1_OE, T2_OE, Error_compare, MAX_CON, SAF_ERRx and ET_OE.
- FIFOx_I interrupt line is ORed of FIFO_FULL[i], FIFO_EMPTY[i], FIFO_OVERFLOW[i] and FIFO_OVERRUN[i].

35.6 ADC commands list

The SU uses a commands list in order to select the command to send to the ADCs when a trigger event occurs. The commands list can hold 64 32-bits commands (see ADC Commands list format CLRx) and it is double-buffered, that is the commands list can be updated in any time between two consecutive MRS but the changes become visible only after that the next MRS occurs.

In order to manage the Commands List, $\log_2(64)$ bits for the position of the first command in the list of commands for each trigger event are necessary and are present in CLCRx registers. The number of commands piloted by the same trigger event is defined directly in the commands list (for each command there is a bit to say if it is the last command).

35.6.1 ADC commands list format

Each command list register (CLRx) contains command for a single ADC. The various fields of this register is as below:

Table 472. CLRx bit fields

Field	Description
CIR	Command Interrupt Request. If set, it generates an interrupt at the start of command execution. It must be set in the same command register which has EC bit set (end of concurrent command) else ADC_I flag will not be set.
LC	Last command sequence for the trigger.

Table 472. CLRx bit fields (continued)

Field	Description
EC	End of the concurrent command.
FIFO_CH	FIFO number for the conversion. This is to select the FIFO in which the converted ADC data would be stored. CTU can support max 8 FIFOs.
ADC_CH	ADC Channel number. To select the channel number of selected ADC. CTU can support maximum 2^7channels.
ADC_UNIT	ADC unit number. To select the external ADC. CTU can support maximum 4 ADCs.

CTU supports Single Conversion Mode and the Multiple Conversion Mode.

- **Single Conversion Mode:** Only 1 ADC would be selected to perform conversion at a time.
- **Multiple Conversion Mode:** Maximum 4 ADCs can be selected to perform sampling and conversion at the same time. The storage of results can be done in series (if same FIFO is selected for different ADCs) or in parallel (if different FIFOs are selected for different ADCs). If same FIFO is selected for multiple ADCs and converted data is available in the same cycle, then command register with least index will have higher priority, for example ADC number of CLR_1 will be having higher priority over ADC number of CLR_2 and so on. Otherwise, the first ADC which ends its conversion, will write its data first in the selected FIFO.

Each trigger source can generate a sequence of conversion windows. Each conversion window can be programmed to select 1 or more ADCs to start sampling and conversion together. The last of the conversion window is identified using EC bit of the CLRx register. If EC bit is ‘1’ it implies the end of present conversion window and CTU will then start generating ADC commands.

Similarly, last of the conversion sequences for a trigger event is identified using LC bit. If LC bit is ‘1’ it implies the end of the conversion sequence for the particular trigger source (*Figure 375*).

The number of maximum concurrent conversions is limited by a static design parameter (CTU_MAX_CONCURRENT=4). If CLRx are programmed in such a way that it exceeds the concurrent conversions from CTU_MAX_CONCURRENT then an error flag is generated (CTUIFR.MAXCON).

EC bit must be set for both conditions - ‘end of concurrent conversion’ and ‘end of conversion sequences’.

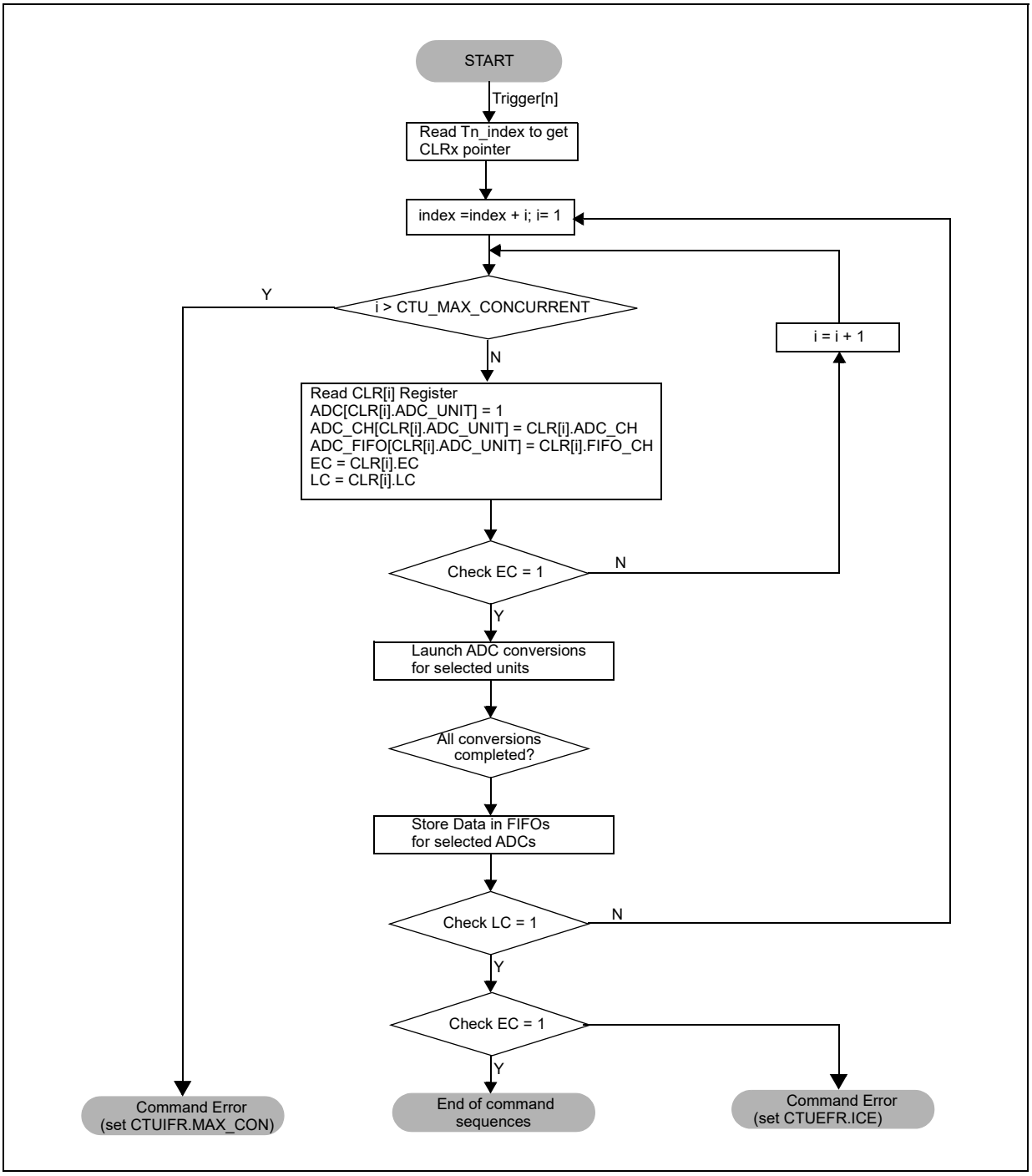
Note: If LC bit is 1 and EC bit is not set, then it implies an error condition and ICE error flag is set.

For multiple conversion mode, maximum 4 command registers are evaluated starting from the base command as is pointed by CLCRx register. And, for a series of concurrent conversions, the next command registers are taken as incremented from the previous concurrent conversion.

As there can be maximum 64 command registers, it is possible that a roll-over condition occurs for selecting next command register. In such case, roll-over happens and next command is taken from CLR1 register.

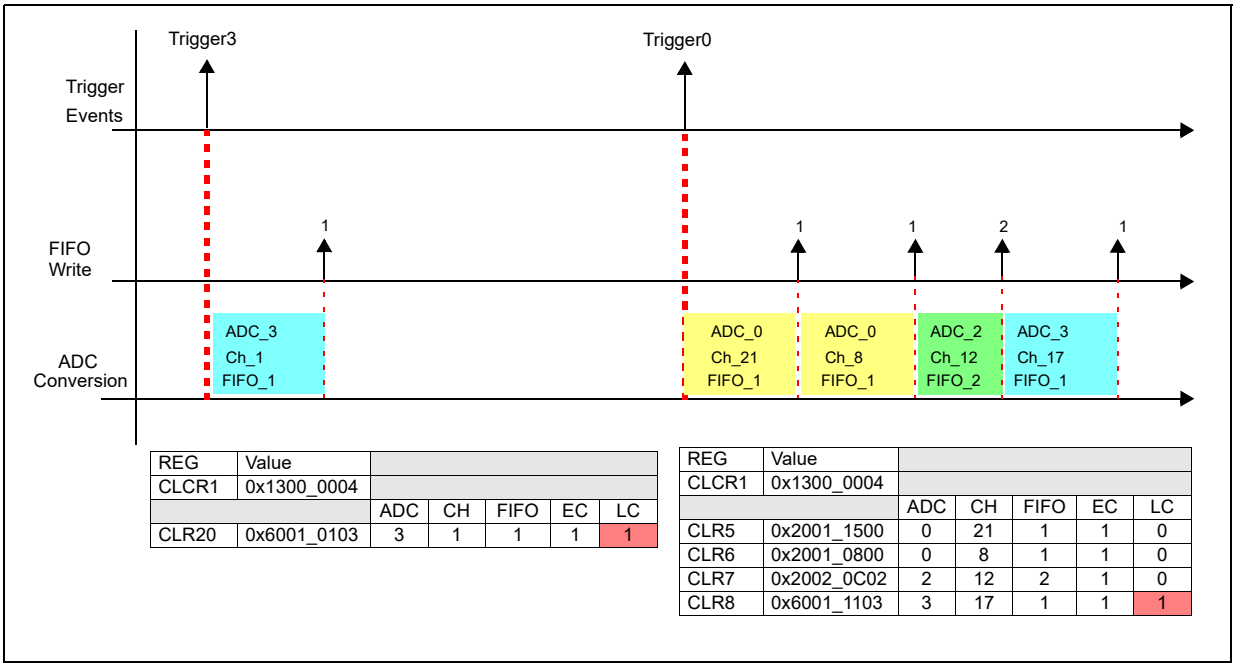
For example Trigger 0 generates command starting from CLR63. If 3 ADCs are to be triggered concurrently then commands will be generated as according to CLR63, CLR64 and CLR1 registers.

Figure 375. ADC command generation flow chart



The conversion timing diagram is shown as below to illustrate single conversion sequences.

Figure 376. Single mode conversion example

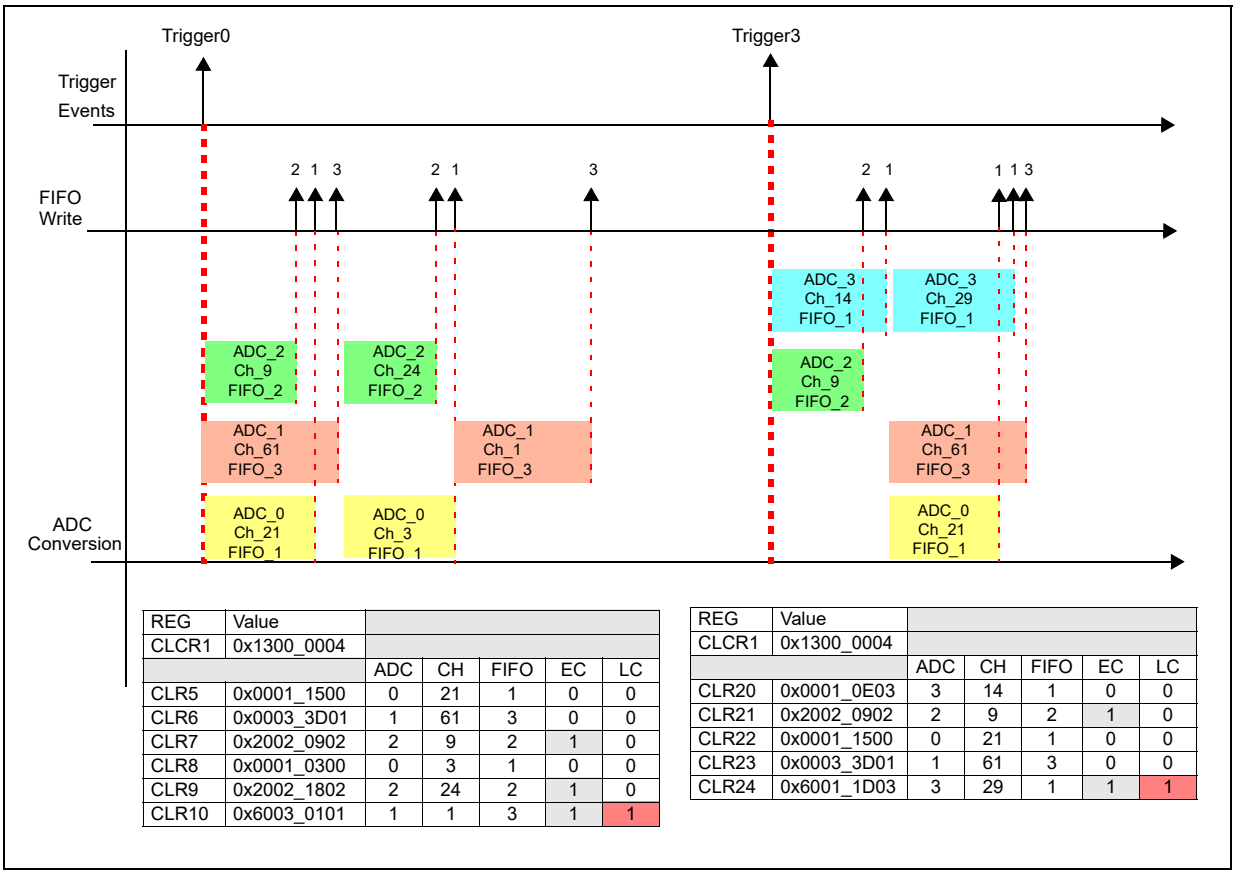


REG	Value					
CLCR1	0x1300_0004					
		ADC	CH	FIFO	EC	LC
CLR20	0x6001_0103	3	1	1	1	1

REG	Value					
CLCR1	0x1300_0004					
		ADC	CH	FIFO	EC	LC
CLR5	0x2001_1500	0	21	1	1	0
CLR6	0x2001_0800	0	8	1	1	0
CLR7	0x2002_0C02	2	12	2	1	0
CLR8	0x6001_1103	3	17	1	1	1

An example of multi-mode conversion is as below:

Figure 377. Multi mode conversion example



REG	Value					
CLCR1	0x1300_0004					
		ADC	CH	FIFO	EC	LC
CLR5	0x0001_1500	0	21	1	0	0
CLR6	0x0003_3D01	1	61	3	0	0
CLR7	0x2002_0902	2	9	2	1	0
CLR8	0x0001_0300	0	3	1	0	0
CLR9	0x2002_1802	2	24	2	1	0
CLR10	0x6003_0101	1	1	3	1	1

REG	Value					
CLCR1	0x1300_0004					
		ADC	CH	FIFO	EC	LC
CLR20	0x0001_0E03	3	14	1	0	0
CLR21	0x2002_0902	2	9	2	1	0
CLR22	0x0001_1500	0	21	1	0	0
CLR23	0x0003_3D01	1	61	3	0	0
CLR24	0x6001_1D03	3	29	1	1	1

The interrupt request bit (CIR) is used as an interrupt request to the CPU when ADC will complete the command (LC=1) with this bit set and it is only for CTU internal use.

35.6.2 ADC results

ADC results can be stored in the channel relevant standard result register and/or in one of the 8 FIFOs. The different FIFOs allow to dispatch ADC results according to the their type of acquisition (ex: phase current, rotor position, ground-noise, other). Each FIFO has its own interrupt and DMA request signal (plus an individual overflow error bit in the FIFO status register). The storage location is specified in the ADC command, that is the FIFOs are available only in CTU Control Mode.

Each entry of a FIFO is 32-bits and it stores converted ADC data and the associated channel number.

The FIFOs are implemented as a peripheral inside the addressing space of the CTU. The ADC communicates with this unit by sending a converted data. The depth of each FIFO can be customized to prevent overflow. A low depth FIFO can be utilized for low acquisition rate conversion and high depth FIFO can be utilized for current acquisitions.

Results in each FIFO can be read by 32-bits read transaction (both the result and the channel number are read in order to avoid blind acquisitions).

The result registers (only for the FIFOs) can be read from 2 different addresses in the CTU memory map. The format of the result depends on the address from which it is read. The available formats are unsigned right-justified format *FIFO right aligned register n (FRn)* (conversion result is unsigned right-justified data, that is the bits [11:0] are used for 10-bit resolution and the bits [15:12] always return zero when read) and signed left-justified format *FIFO left aligned register n (FLn)* (conversion result is signed left-justified data, that is the bit [15] is reserved for sign and is always read as zero for this ADC, the bits[14:3] represents the 1-b ADC data).

35.6.3 ADC-CTU interface timing

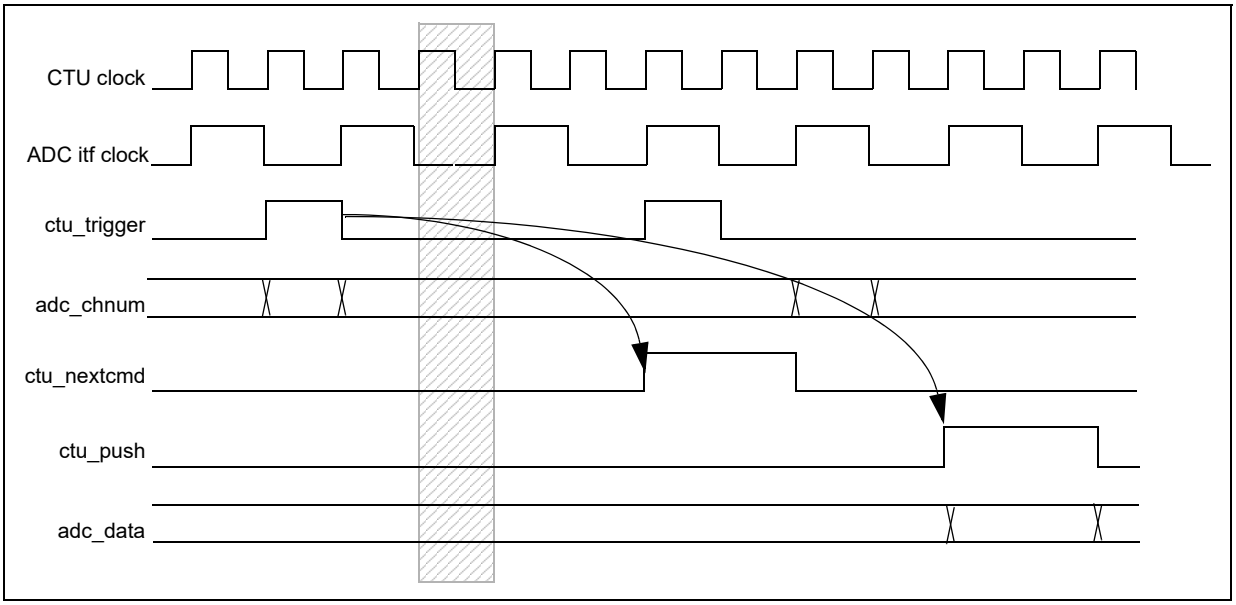
CTU is able to command a single conversion to one of the 4 ADCs (single conversion mode) or to many ADCs at the same time (multi-conversion mode).

CTU clock must be synchronous with the ADC digital interface module clock and must be same or faster than ADC digital interface module clock.

In single conversion mode it is possible to send command one at a time to the selected ADC. A new command can be sent only when the ongoing ADC completes its conversion (correspondent next_command is provided from the CTU_ADC interface). If it tries to send a new command before the next_command an overrun error occurs.

In multi-conversion mode, more than 1 ADCs are selected to start conversion. The channel number of these ADCs will be decided based on the respective command register. Similarly the FIFO in which the converted data is to be stored can be programmed differently for each ADC. For the next multi-conversion mode, the state machine waits till all conversions in the ongoing cycle has been completed. If it tries to send a new command before the next_command an overrun error occurs.

Figure 378. ADC-CTU interface timing



35.7 Low power mode

To reduce power consumption, stop mode mechanism is implemented in the SoC. This is done by generating a stop request from the magic carpet to ECTU. When ECTU receives the stop signals it provides following outputs:

1. ipg_stop_ack : This output is immediately asserted in the same cycle of stop mode request.
2. ipg_stop_ack_MT: This output is generated after synchronization of stop mode request on MT_clk.

It is not possible to stop only one of the 2 clocks and only one stop mode request is used for both.

The ECTU IP does not stop its state machine on receiving the stop mode request. Its stopped only when the magic carpet module gates the clocks. Therefore it is possible that the stop mode request may arrive during an ongoing conversion and as the acknowledge is immediately asserted, ECTU clock gets gated before saving the conversion result. It may result in inconsistent FIFO status.Therefore the software must ensure that there is no ongoing conversion before entering the STOP mode. This is applicable only for software controlled STOP mode.

It is also possible that ADC command generator state machine is waiting for the last command before the clock is gated. And when the clock is restarted, ECTU state machine may be out-of-sync with ADC digital interface. Therefore, a CTUCR.CTU_ADC_RESET bit is provided to reset the state machine and to restart the conversion after exiting from stop mode. It can be used for both software or hardware controlled STOP mode.

35.8 Safety features

35.8.1 Conversion time check

Conversion time for an ADC is defined as the time interval between start of a conversion and the end of a conversion (time between *ctu_trigger* and *ctu_nextcmd* in *Figure 378*). For the safety purposes it is useful to check if the ongoing conversion gets over within a specified time range. The value should be chosen considering the fact that the *ctu_nextcmd* is generated one cycle earlier than the actual end of conversion.

There is a 16-bit internal counter running on system interface clock (read/write bus clock), and starts counting at the start of a conversion. Every start of a conversion (ctu_trigger pulse) resets the counter. As all conversions (single as well as parallel) start with the same trigger, there is a single counter only for all conversions (supports maximum 4 parallel conversions).

For each ongoing conversion (maximum 4 parallel conversions), the conversion time check is done using CTU_EXPECTED_x and CTU_CNT_RANGE_x registers. These are 4 registers available to measure conversion time for 4 possible parallel conversions. In case of a single conversion, only CTU_EXPECTED_1/CTU_CNT_RANGE_1 registers are used. In case of two parallel conversions, CTU_EXPECTED_1/CTU_EXPECTED_2 registers are used and so on.

The conversion time range is programmed using CTU_CNT_RANGE_x bits. If a bit of this register is set, the correspondent bit of the CTU_EXPECTED_x register is used to compare time interval with the internal real time counter.

35.8.2 CTU reset

It is possible that CTU state machine is waiting for end of conversion signal from ADC, and due to any fault it does not arrive. Therefore it is required to reset the CTU state machine to start next conversion.

In case of any need to reset the CTU, following methods can be used.

- CTUCR.CTU_ADC_reset

  This bit when set by software, generates an internal pulse which puts the ADC command state machine in a reset state. This condition is treated similar as ‘last_command’ condition has occured and state machine comes out of the conversion mode for the particular trigger conversion sequence. Next trigger events (internal or external) would be treated normally.

- ADCDIG.MCR.CTUEN

  These bits are available in the SAR ADC Digital Interface and are used to program ADC Digital Interface in CTU mode. These bits are connected to *ctuen*<> inputs of the CTU. If all of the ADC interfaces connected to a CTU reset their respective CTUEN bits then ADC command state machine would be reset. If any of the ADC interface remains with CTUEN set, then ADC command state machine would not be reset.

The above methods reset only the state machine and do not reset any of control/status bits in CTU.

35.8.3 Motor function iMonitor

To enhance safety of the complete motor control computation cycle, a motor function monitor using a 32-b CRC engine is available. This engine updates a CRC based on following:

- Type of triggers generated (external trigger, eTimer<x>_trigger, ADC_trigger) along-with timestamp
- ADCs which are triggered for conversion
- ADC channel numbers
- ADC FIFO's numbers
- ADC end of conversion events along-with timestamp
- CTU error signals

All above information are used in CRC computation alongwith the timestamp provided by the TGS counter.

The algorithm for CRC is CRC-32 (Ethernet):

$x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$

The initial seed of the CRC is 0xFFFF_FFFF. The design implementation executes the CRC computation in a single cycle (parallel data loading) for each input.

Based on the programming of various command registers, input triggers events, ADC conversion time, it is possible to compute CRC during application development. Thus calculated CRC can be programmed in CTU_EXP_CRC_VAL as expected CRC for each MRS cycle.

CRC computation is triggered by the MRS event and a check with expected CRC (CTU_EXP_CRC_VAL) is performed at the next MRS event. In case of a mismatch, an error flag (CTUEFR.CRC_ERR) is generated as well as a signal is sent to FCCU.

To keep a check on any error condition during the MRS cycle, all error events related to CTUEFR[10:0] bits are latched. These latched error flags are XORed with the computed CRC[10:0] bits before its compared with the expected CRC. This ensures that no error condition has occured in the MRS cycle apart from the normal CRC check.

Based on a certain programming sequence of ADC command registers, Trigger sources etc, it provides an unique combination of events in a MRS cycle resulting in unique checksum. With the CRC comparison, it is ensured that all events have occured in the expected manner thus ensuring the safety of the motor control loop.

The checksum engine is controlled using CTUCR.CTU_CRC_EN bit and works as below:

Table 473. CRC computation steps

Cycle	CRC state	CRC input (32-bit)	CRC computation enable condition	Description
1	IDLE	If (trigger_event) {12'b0, Ext_trigger, eTimer2_trigger, eTimer1_trigger, |(ctu_trigger), TGS_Counter[15:0]} elsif (ctu_push_event) {ctu_push[15:0], TGS_Counter[15:0]}	If any of the trigger events (eTimer1, eTimer2, Ext_Trigger, ADC_trigger) occured, or ctu_push (end of conv) occured then only the CRC is computed else retains its previous value. **ctu_push** computations are enabled by CTU_MASK_CRC_VAL.MASK_PUSH=0 condition TGS_Counter computations are masked using CTU_MASK_CRC_VAL.MASK_TIMESTAMP = "111" else always enabled	Individual mask bits are available for trigger inputs. MASK_ext_TRG, MASK_eTimer2_TRG, MASK_eTimer1_TRG, MASK_ADC_TRG. If any trigger input is masked then the corresponding CRC input bit is put to 1'b0. Also, if any ctu_push events are occured in non-IDLE states, then that is latched and used in computation only in IDLE state.
2	TRIGGER	{16'b0, ctu_trigger[15:0]}	CTU_MASK_CRC_VAL.MASK_ADC_NUM=0	Each bit is used for each ADC as per its position.
3	ADC_CH_0	{4'b0, ctu_adc_channel[27:0]}	CTU_MASK_CRC_VAL.MASK_ADC_CH=0	7 bits are assigned for channel information for each ADC. [27:0] contains information about ADC0-3
4	ADC_CH_1	{4'b0, ctu_adc_channel[59:32]}	CTU_MASK_CRC_VAL.MASK_ADC_CH=0 & (Number of ADCs > 4)	7 bits are assigned for channel information for each ADC. [59:32] contains information about ADC4-7
5	ADC_CH_2	{4'b0, ctu_adc_channel[91:64]}	CTU_MASK_CRC_VAL.MASK_ADC_CH=0 & (Number of ADCs > 8)	7 bits are assigned for channel information for each ADC. [91:64] contains information about ADC8-11
6	ADC_CH_3	{4'b0, ctu_adc_channel[123:96]}	CTU_MASK_CRC_VAL.MASK_ADC_CH=0 & (Number of ADCs > 12)	7 bits are assigned for channel information for each ADC. [123:96] contains information about ADC12-15
7	FIFO_0	{ctu_fifo[31:0]}	CTU_MASK_CRC_VAL.MASK_ADC_FIFO=0	4 bits are assigned for FIFO information for each ADC. [31:0] contains information about ADC0-7
8	FIFO_1	{ctu_fifo[63:32]}	CTU_MASK_CRC_VAL.MASK_ADC_FIFO=0 & (Number of ADCs > 8)	4 bits are assigned for FIFO information for each ADC. [63:32] contains information about ADC8-15

Note: The CRC method when enabled provides a deterministic checksum calculation based on a defined set of trigger commands. During development once the configuration is validated, its associated CRC output can be read and stored as a reference value which will be checked for each conversion cycle and accordingly error alarm would be generated.

35.9 CTU error flags examples

Following examples are related to multiple programming mode only with maximum four concurrent conversions.

35.9.1 Invalid command error

This error condition occurs in following conditions:

- EC=0 and LC=1 in a command register
- Same ADC number appears more than once in CLRx in multiple conversion mode

Table 474. Invalid command error example

Trigger	Conversion cycle	CLRx[0:3]	Behavior
1	concurrent #1	0x00_01_05_01 0x00_02_04_00 0x00_04_03_02 0x02_01_02_03	- Conversion triggered for: ADC_1 Chnum_5 FIFO_1 ADC_0 Chnum_4 FIFO_2 ADC_2 Chnum_3 FIFO_4 ADC_3 Chnum_2 FIFO_1
	concurrent #2 (last command)	0x00_01_03_01 0x40_02_03_02 0x60_01_05_03	- Conversion triggered for: ADC_1 Chnum_3 FIFO_1 ADC_2 Chnum_3 FIFO_2 ADC_3 Chnum_5 FIFO_1 - EC=1 but LC=0 so CTUEFR.ICE error flag will be set. - LC=EC=1 so it will be considered as last command.
2	concurrent #1	0x00_01_03_01 0x00_02_02_01 0x60_01_05_03	- Conversion triggered for: ADC_1 Chnum_3 FIFO_1 ADC_3 Chnum_5 FIFO_1 - As ADC_1 is repeated therefore CTUEFR.ICE error flag will be set.

35.9.2 Maximum concurrent conversion error

This error occurs when number of conversions in a concurrent conversion exceeds 4. If it happens no conversion takes place and the trigger source is terminated. CTU then waits for next trigger.

Table 475. Maximum concurrent conversion error example

Trigger	Conversion cycle	CLRx[0:3]	Behavior
1	concurrent #1	0x00_01_05_01 0x00_02_04_00 0x00_04_03_02 0x00_01_02_03	- No conversion triggered - CTUEFR.MAX_CON error flag is set as EC bit is not set in these 4 command registers. - It is the last command for Trigger_1.
2	concurrent #1	0x00_01_03_01 0x40_02_03_02 0x00_01_05_03 0x40_01_05_03	- No conversion triggered - CTUEFR.MAX_CON error flag is set as EC bit is not set in these 4 command registers. - EC=1 but LC=0 so CTUEFR.ICE error flag will be set. - It is the last command for Trigger_2.

35.10 Register description

All registers are 32-bit wide.

The following sections detail the individual registers.

35.10.1 Synchronization

The following kind of synchronization mechanism are used:

1. The MRS occurrence can start the synchronization of the data from the ipg_clk domain to the MT_clk domain. This mode is used for all double-buffered registers.
2. TGSISR_RE bit is used to synchronize TGSISR register. When the user set TGSISR_RE the value write in theTGSISR is used.
3. The synchronization from the ipg_clk domain to the MT_clk domain is continuously performed for all not double-buffered registers (such as CTUCR, CTUIFR and so on). In this case two consecutive pulse will be filtered if it happens in a time less than 3 ipg_clk + 4 mt_clk. For this reason is not possible to write two consecutive software trigger in a time less than the specified one.
4. Similarly, when a software trigger is generated using CTUCR, the corresponding interrupt flags in CTUIFR will be set only after 3 ipg_clk+4 mt_clk delay. This is due to the internal synchronization logic present between 2 clock domains.

35.10.2 Double buffered registers

Some registers of the CTU are double buffered. This mechanism is necessary if we will be sure that data write from CPU bus will be stable when are used from CTU kernel. When MRS is set, the data stored in a register in ipg_clk clock domain will be copied into the same registers in the other clock domain. To avoid error condition due to a MRS signal that arrive when a write operation is not fully completed following two bits are used.

1. GRE (General register enable)
2. FGRE (General register disable)

The reload of all double-buffered registers is enabled by setting the GRE (General reload enable) bit in the CTU control register in order to guarantee the consistency of all double-buffered registers. The user must ensure that the trigger command list is updated before

that a MRS occurs. If the update is not performed, the previous values of all double-buffered registers remain active and the CTU will optionally perform an interrupt request.

All the double-buffered registers use the same bit (General Reload Enable - GRE) to enable the reload when the MRS occurs. This bit is used to signal that the user has updated the registers and so a reload can be performed without any problem. The MRS bit is R/S (Read/Set). If the bit is 1 the reload can be performed, while if this bit is 0, the reload is not performed. A correct reload resets this bit. It is also possible to reset this bit by software setting CGRE bit. This bit can be reset in a software way but, anyway, it is reset when GRE bit is at zero.

In order to verify if a reload error occurs, FGRE (**Flag GRE bit in the CTU Control Register**) bit is used. When one of the double-buffered registers is written, FGRE flag is set to 1 and it is reset by a correct reload.

When the MRS occurs while FGRE =1 and GRE =1, a correct reload is performed (because all intended registers have been updated before the MRS occurs).

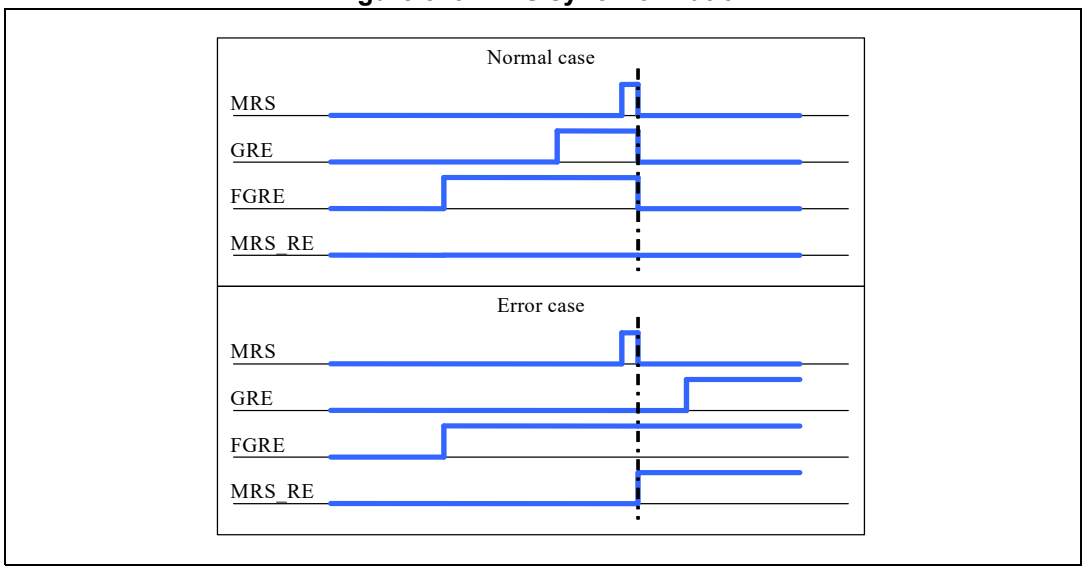
If FGRE=1 and GRE=0, a reload is not performed, the error flag (MRS_RE) is set and (if enabled) an interrupt for an error is performed (in this case at least one register was written but the update has not been ended before the MRS occurrence).

If FGRE=0 it is not necessary to perform a reload because all the double-buffered registers are unchanged.

If GRE=1 with FGRE=0, a MRS_RE is generated on the next master reload.

Because of the two asynchronous domains an error is provided if the GRE FGRE and MRS in the Motor control clock domain are not in the explained configuration. It is possible that an error is generated and GRE and FGRE are set. This is due to the time needed for the synchronization. If the MRS happens when GRE in the motor control domain is reset and the GRE in the motor control domain is not yet reset the error is provided. The following figure shows the timing diagram in the motor control clock domain of a correct reload and of an mistaken reload.

Figure 379. MRS synchronization



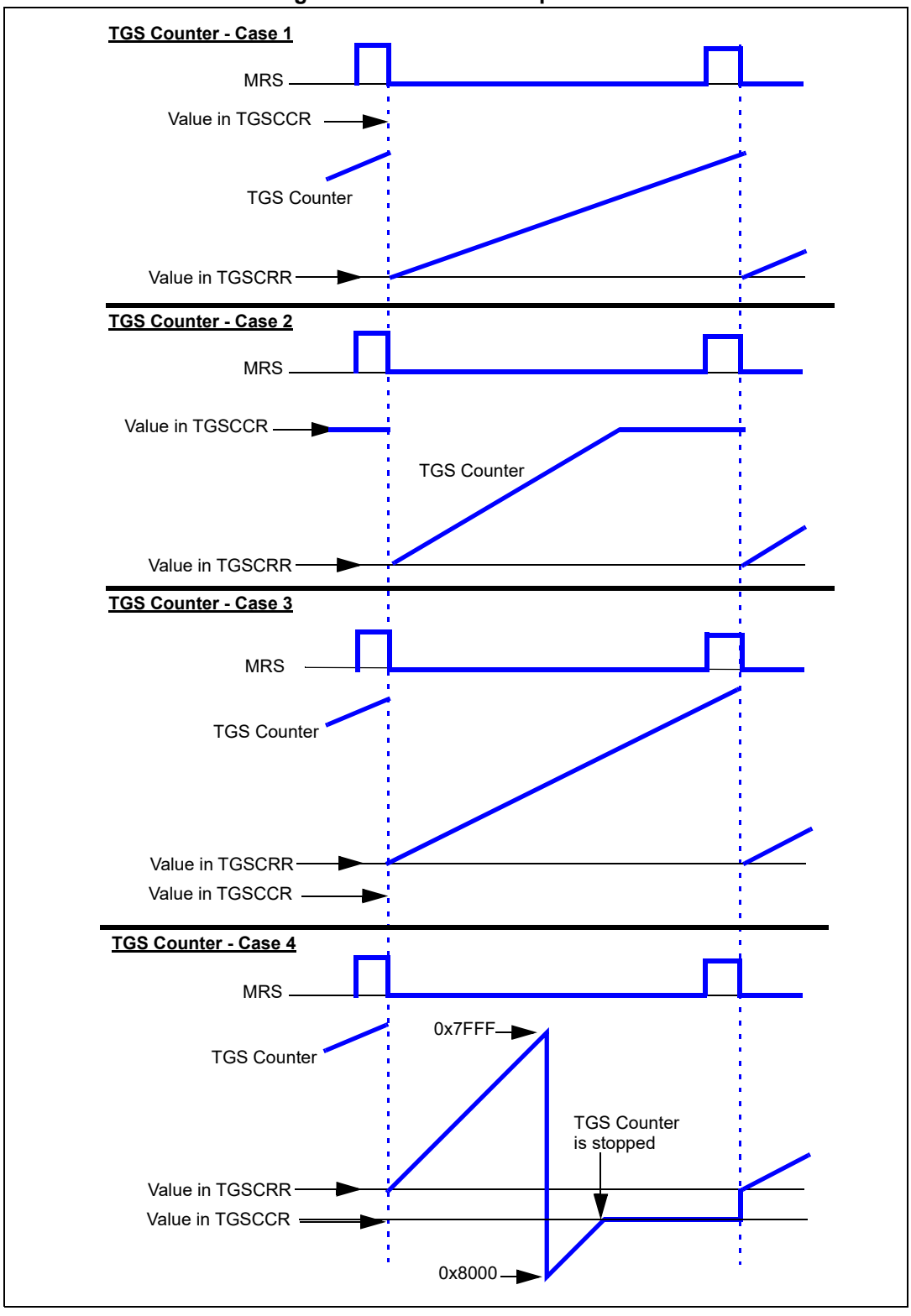
35.10.3 TGS counter precision

The TGS Counter must to be able to count as the one of the PWM: from negative to positive, that is from 0x8000 to 0x7FFF. In the figure below are shown some cases in order to explain the TGS counter operation. The compare operation to stop the TGS Counter is not enabled during the first counting cycle in order to allow the counting if the value of the TGSCRR is the same of the TGSCCR.

The precision is mandatory in motor control application, then the timing of the trigger generation must be deterministic. An offset of one or two clock cycles can be acceptable, but also the offset should be deterministic. After the design phase, the offset must be specified in user documentation.

If the TGS counter contains the same value of TGSCCR register an error is generated.

Figure 380. TGS counter precision



The precision is mandatory in motor control application, then the timing of the trigger generation must be deterministic. An offset of one or two clock cycles can be acceptable, but also the offset should be deterministic.

The trigger for the SU is generated always when the counter reaches the value in the trigger compare register. The number of clocks required to reach the value in the register starting from a new master reload is equal to:

$$(TxCR \times (Pres + 1)) + 1$$

35.10.4 Memory map

Table 476. TGS registers

Address offset	Register	Double buffered	Synchronization	Reset value
0x0000	Trigger Generator Subunit Input Selection Register (TGSISR)	Yes	TGSISR_RE	0x0000 0000
0x0004	Trigger Generator Subunit Control Register (TGSCR)	Yes	MRS	0x0000 0000
0x0008	Trigger Compare Register (TCR0)	Yes	MRS	0x0000 0000
0x000C	Trigger Compare Register 1 (TCR1)	Yes	MRS	0x0000 0000
0x0010	Trigger Compare Register 2 (TCR2)	Yes	MRS	0x0000 0000
0x0014	Trigger Compare Register 3 (TCR3)	Yes	MRS	0x0000 0000
0x0018	Trigger Compare Register 4 (TCR4)	Yes	MRS	0x0000 0000
0x001C	Trigger Compare Register 5 (TCR5)	Yes	MRS	0x0000 0000
0x0020	Trigger Compare Register 6 (TCR6)	Yes	MRS	0x0000 0000
0x0024	Trigger Compare Register 7 (TCR7)	Yes	MRS	0x0000 0000
0x0028–0x0047	Reserved			
0x0048	TGS Counter Compare Register (TGSCCR)	Yes	MRS	0x0000 0000
0x004C	TGS Counter Reload Register (TGSCRR)	Yes	MRS	0x0000 0000

35.10.4.1 FIFO registers

Table 477. SU registers

Address offset	Register	Double buffered	Synchronization	Reset value
0x0100	Commands List Control Register 1 (CLCR1)	Yes	MRS	0x0000 0000
0x0104	Commands List Control Register 2 (CLCR2)	Yes	MRS	0x0000 0000
0x0108	Commands List Control Register 3 (CLCR3)	Yes	MRS	0x0000 0000
0x010C	Commands List Control Register 4 (CLCR4)	Yes	MRS	0x0000 0000
0x0110	Trigger Handler Control Register 1 (THCR1)	Yes	MRS	0x0000 0000
0x0114	Trigger Handler Control Register 2 (THCR2)	Yes	MRS	0x0000 0000

Table 477. SU registers (continued)

Address offset	Register	Double buffered	Synchronization	Reset value
0x0118	Trigger Handler Control Register 3 (THCR3)	Yes	MRS	0x0000 0000
0x011C	Trigger Handler Control Register 4 (THCR4)	Yes	MRS	0x0000 0000
0x0120–0x02AC	Commands List Register 0–63 (CLR1–CLR64)	Yes	MRS	0x0000 0000

Table 478. FIFO registers

Address offset	Register	Double buffered	Synchronization	Reset value
0x0400	Control register (CR)	NO	No synchronization	0x0000_0000
0x0404	Control register FIFO 1 (FCR1)	NO	No synchronization	0x0000 0000
0x0408	Control register FIFO 2 (FCR2)	NO	No synchronization	0x0000 0000
0x040C	Threshold1 (TH1)	NO	No synchronization	0x0000 0000
0x0410	Threshold 2 (TH2)	NO	No synchronization	0x0000 0000
0x0414	Threshold 3 (TH3)	NO	No synchronization	0x0000 0000
0x0418	Threshold 4 (TH4)	NO	No synchronization	0x0000 0000
0x041C	Status register 1 (ST1)	NO	No synchronization	0x0000 0000
0x0420	Status register 2 (ST2)	NO	No synchronization	0x0000 0000
0x0424	FIFO right aligned data (FR0)	NO	Handshake	0x0000 0000
0x0428	FIFO right aligned data (FR1)	NO	Handshake	0x0000 0000
0x042C	FIFO right aligned data (FR2)	NO	Handshake	0x0000 0000
0x0430	FIFO right aligned data (FR3)	NO	Handshake	0x0000 0000
0x0434	FIFO right aligned data (FR4)	NO	Handshake	0x0000 0000
0x0438	FIFO right aligned data (FR5)	NO	Handshake	0x0000 0000
0x043C	FIFO right aligned data (FR6)	NO	Handshake	0x0000 0000
0x0440	FIFO right aligned data (FR7)	NO	Handshake	0x0000 0000
0x0444	FIFO right aligned data (FR8)	NO	Handshake	0x0000 0000
0x0448	FIFO right aligned data (FR9)	NO	Handshake	0x0000 0000
0x044C	FIFO right aligned data (FR10)	NO	Handshake	0x0000 0000
0x0450	FIFO right aligned data (FR11)	NO	Handshake	0x0000 0000
0x0454	FIFO right aligned data (FR12)	NO	Handshake	0x0000 0000
0x0458	FIFO right aligned data (FR13)	NO	Handshake	0x0000 0000
0x045C	FIFO right aligned data (FR14)	NO	Handshake	0x0000 0000
0x0460	FIFO right aligned data (FR15)	NO	Handshake	0x0000 0000
0x0464	FIFO left aligned data (FL0)	NO	Handshake	0x0000 0000
0x0468	FIFO left aligned data (FL1)	NO	Handshake	0x0000 0000

Table 478. FIFO registers (continued)

Address offset	Register	Double buffered	Synchronization	Reset value
0x046C	FIFO left aligned data (FL2)	NO	Handshake	0x0000 0000
0x0470	FIFO left aligned data (FL3)	NO	Handshake	0x0000 0000
0x0474	FIFO left aligned data (FL4)	NO	Handshake	0x0000 0000
0x0478	FIFO left aligned data (FL5)	NO	Handshake	0x0000 0000
0x047C	FIFO left aligned data (FL6)	NO	Handshake	0x0000 0000
0x0480	FIFO left aligned data (FL7)	NO	Handshake	0x0000 0000
0x0484	FIFO left aligned data (FL8)	NO	Handshake	0x0000 0000
0x0488	FIFO left aligned data (FL9)	NO	Handshake	0x0000 0000
0x048C	FIFO left aligned data (FL10)	NO	Handshake	0x0000 0000
0x0490	FIFO left aligned data (FL11)	NO	Handshake	0x0000 0000
0x0494	FIFO left aligned data (FL12)	NO	Handshake	0x0000 0000
0x0498	FIFO left aligned data (FL13)	NO	Handshake	0x0000 0000
0x049C	FIFO left aligned data (FL14)	NO	Handshake	0x0000 0000
0x04A0	FIFO left aligned data (FL15)	NO	Handshake	0x0000 0000

Table 479. Others CTU registers

Address offset	Register	Double buffered	Synchronization	Reset value
0x0500	CTU Error Flag Register (CTUEFR)	NO	Handshake	0x0000 0000
0x0504	CTU Interrupt Flag Register (CTUIFR)	NO	Handshake	0x0000 0000
0x0508	CTU Interrupt Register (CTUIR)	NO	No synchronization	0x0000 0000
0x050C	Control On Time Register (COTR)	Yes	MRS	0x0000 0000
0x0510	CTU Control Register (CTUCR)	NO	Handshake	0x0000 0000
0x0514	CTU digital Filter (CTU_FILTER)	NO	Filter enable	0x0000 0000
0x0518	CTU Configuration Status Register 1 (CTU_CFGSR1)	NO	No synchronization	as per static parameters
0x051C	CTU Configuration Status Register 2 (CTU_CFGSR2)	NO	No synchronization	as per static parameters
0x0520	CTU Expected Value Register 1 (CTU_EXPECTED_1)	NO	No synchronization	0x0000_FFFF
0x0524	CTU Expected Value Register 2 (CTU_EXPECTED_2)	NO	No synchronization	0x0000_FFFF
0x0528	CTU Expected Value Register 3 (CTU_EXPECTED_3)	NO	No synchronization	0x0000_FFFF
0x052C	CTU Expected Value Register 4 (CTU_EXPECTED_4)	NO	No synchronization	0x0000_FFFF

Table 479. Others CTU registers (continued)

Address offset	Register	Double buffered	Synchronization	Reset value
0x0530	CTU Counter Range 1 (CTU_CNT_RANGE_1)	NO	No synchronization	0x0000_0000
0x0534	CTU Counter Range 2 (CTU_CNT_RANGE_2)	NO	No synchronization	0x0000_0000
0x0538	CTU Counter Range 3 (CTU_CNT_RANGE_3)	NO	No synchronization	0x0000_0000
0x053C	CTU Counter Range 4 (CTU_CNT_RANGE_4)	NO	No synchronization	0x0000_0000
0x0540	CTU_EXP_CRC_VAL	Yes	MRS	0xFFFF_FFFF
0x544	CTU_COMP_CRC_VAL	Yes	Handshake	0xFFFF_FFFF
0x0548	CTU_MASK_CRC_VAL	Yes	MRS	0x0000_0000

35.10.5 Trigger generator subunit input selection register (TGSISR)

Offset: 0x000 Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	I15_FE	I15_RE	I14_FE	I14_RE	I13_FE	I13_RE	I12_FE	I12_RE	I11_FE	I11_RE	I10_FE	I10_RE	I9_FE	I9_RE	I8_FE	I8_RE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	I7_FE	I7_RE	I6_FE	I6_RE	I5_FE	I5_RE	I4_FE	I4_RE	I3_FE	I3_RE	I2_FE	I2_RE	I1_FE	I1_RE	I0_FE	I0_RE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 381. Trigger generator subunit input selection register (TGSISR)

Table 480. TGSISR field descriptions

Field	Description
0 I15_FE	ctu_trg_in[15] Falling edge Enable 0 disabled 1 enabled **Note:** See *Figure 33: Module to module connections* in *Chapter 7: Device configuration*.
1 I15_RE	ctu_trg_in[15] Rising edge Enable 0 disabled 1 enabled **Note:** See *Figure 33: Module to module connections* in *Chapter 7: Device configuration*.
2 I14_FE	ctu_trg_in[14] Falling edge Enable 0 disabled 1 enabled **Note:** See *Figure 33: Module to module connections* in *Chapter 7: Device configuration*.

Table 480. TGSISR field descriptions (continued)

Field	Description
3 I14_RE	ctu_trg_in[14] Rising edge Enable 0 disabled 1 enabled **Note:** See *Figure 33: Module to module connections* in *Chapter 7: Device configuration*.
4 I13_FE	ctu_trg_in[13] Falling edge Enable 0 disabled 1 enabled **Note:** See *Figure 33: Module to module connections* in *Chapter 7: Device configuration*.
5 I13_RE	ctu_trg_in[13] Rising edge Enable 0 disabled 1 enabled **Note:** See *Figure 33: Module to module connections* in *Chapter 7: Device configuration*.
6 I12_FE	ctu_trg_in[12] Falling edge Enable 0 disabled 1 enabled **Note:** See *Figure 33: Module to module connections* in *Chapter 7: Device configuration*.
7 I12_RE	ctu_trg_in[12] Rising edge Enable 0 disabled 1 enabled **Note:** See *Figure 33: Module to module connections* in *Chapter 7: Device configuration*.
8 I11_FE	ctu_trg_in[11] Falling edge Enable 0 disabled 1 enabled **Note:** See *Figure 33: Module to module connections* in *Chapter 7: Device configuration*.
9 I11_RE	ctu_trg_in[11] Rising edge Enable 0 disabled 1 enabled **Note:** See *Figure 33: Module to module connections* in *Chapter 7: Device configuration*.
10 I10_FE	ctu_trg_in[10] Falling edge Enable 0 disabled 1 enabled **Note:** See *Figure 33: Module to module connections* in *Chapter 7: Device configuration*.
11 I10_RE	ctu_trg_in[10] Rising edge Enable 0 disabled 1 enabled **Note:** See *Figure 33: Module to module connections* in *Chapter 7: Device configuration*.

Table 480. TGSISR field descriptions (continued)

Field	Description
12 I9_FE	ctu_trg_in[9] Falling edge Enable 0 disabled 1 enabled **Note:** See *Figure 33: Module to module connections* in *Chapter 7: Device configuration*.
13 I9_RE	ctu_trg_in[9] Rising edge Enable 0 disabled 1 enabled **Note:** See *Figure 33: Module to module connections* in *Chapter 7: Device configuration*.
14 I8_FE	ctu_trg_in[8] Falling edge Enable 0 disabled 1 enabled **Note:** See *Figure 33: Module to module connections* in *Chapter 7: Device configuration*.
15 I8_RE	ctu_trg_in[8] Rising edge Enable 0 disabled 1 enabled **Note:** See *Figure 33: Module to module connections* in *Chapter 7: Device configuration*.
16 I7_FE	ctu_trg_in[7] Falling edge Enable 0 disabled 1 enabled **Note:** See *Figure 33: Module to module connections* in *Chapter 7: Device configuration*.
17 I7_RE	ctu_trg_in[7] Rising edge Enable 0 disabled 1 enabled **Note:** See *Figure 33: Module to module connections* in *Chapter 7: Device configuration*.
18 I6_FE	ctu_trg_in[6] Falling edge Enable 0 disabled 1 enabled **Note:** See *Figure 33: Module to module connections* in *Chapter 7: Device configuration*.
19 I6_RE	ctu_trg_in[6] Rising edge Enable 0 disabled 1 enabled **Note:** See *Figure 33: Module to module connections* in *Chapter 7: Device configuration*.
20 I5_FE	ctu_trg_in[5] Falling edge Enable 0 disabled 1 enabled **Note:** See *Figure 33: Module to module connections* in *Chapter 7: Device configuration*.

Table 480. TGSISR field descriptions (continued)

Field	Description
21 I5_RE	ctu_trg_in[5] Rising edge Enable 0 disabled 1 enabled **Note:** See *Figure 33: Module to module connections* in *Chapter 7: Device configuration*.
22 I4_FE	ctu_trg_in[4] Falling edge Enable 0 disabled 1 enabled **Note:** See *Figure 33: Module to module connections* in *Chapter 7: Device configuration*.
23 I4_RE	ctu_trg_in[4] Rising edge Enable 0 dibled 1 enablbled **Note:** See *Figure 33: Module to module connections* in *Chapter 7: Device configuration*.
24 I3_FE	ctu_trg_in[3] Falling edge Enable 0 disabled 1 enabled **Note:** See *Figure 33: Module to module connections* in *Chapter 7: Device configuration*.
25 I3_RE	ctu_trg_in[3] Rising edge Enable 0 disabled 1 enabled **Note:** See *Figure 33: Module to module connections* in *Chapter 7: Device configuration*.
26 I2_FE	ctu_trg_in[2] Falling edge Enable 0 disabled 1 enabled **Note:** See *Figure 33: Module to module connections* in *Chapter 7: Device configuration*.
27 I2_RE	ctu_trg_in[2] Rising edge Enable 0 disabled 1 enabled **Note:** See *Figure 33: Module to module connections* in *Chapter 7: Device configuration*.
28 I1_FE	ctu_trg_in[1] Falling edge Enable 0 disabled 1 enabled **Note:** See *Figure 33: Module to module connections* in *Chapter 7: Device configuration*.
29 I1_RE	ctu_trg_in[1] Rising edge Enable 0 disabled 1 enabled **Note:** See *Figure 33: Module to module connections* in *Chapter 7: Device configuration*.

Table 480. TGSISR field descriptions (continued)

Field	Description
30 I0_FE	ctu_trg_in[0] Falling edge Enable 0 disabled 1 enabled **Note:** See *Figure 33: Module to module connections* in *Chapter 7: Device configuration*.
31 I0_RE	ctu_trg_in[0] Rising edge Enable 0 disabled 1 enabled **Note:** See *Figure 33: Module to module connections* in *Chapter 7: Device configuration*.

35.10.6 Trigger generator subunit control register (TGSCR)

Offset: 0x0004 Access: User Read/Write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	ET_TM	PRES		MRS_SM					TGS_M
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 382. Trigger generator subunit control register (TGSCR)

Table 481. TGSCR field descriptions

Field	Description
23 ET_TM	This bit is used to enable toggle mode for external trigger.
24:25 PRES	TGS prescaler selection bits 00 1 01 2 10 3 11 4
26:30 MRS_SM	MRS Selection in Sequential Mode (4-bits to select one of 16 input trigger signals).
31 TGS_M	Trigger Generator Subunit Mode 0 Triggered Mode 1 Sequential Mode

35.10.7 Trigger compare register *n* (TCR*n*)

There are 8 TCR registers to support 8 triggers.

Offset: 0x008 +n*0x4 (n=0 to 7) Access: User Read/Write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TnCR_value															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 383. Trigger compare register *n* (TCR*n*)

Table 482. TCR*n* field descriptions

Field	Description
16:31 TnCR_value	Value to be compared with the counter value to generate the trigger.

35.10.8 TGS counter compare register (TGSCCR)

Offset: 0x0048 Access: User Read/Write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TGSCCR_value															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 384. TGS counter compare register (TGSCCR)

Table 483. TGSCCR field descriptions

Field	Description
16:31 TGSCCR_value	This register is used to stop the counter when TGSCCR_value is reached.

35.10.9 TGS counter reload register (TGSCRR)

Offset: 0x004C Access: User Read/Write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TGSCRR_value															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 385. TGS counter reload register (TGSCRR)

Table 484. TGSCRR field descriptions

Field	Description
16:31 TGSCRR_value	This register is used to reload the counter when MRS (in triggered mode) or ES (in sequential mode) is one. The value of this register is load in the motor control domain after the master reload and is used only after an other master reload.

35.10.10 Commands list control register 1 (CLCR1)

Offset: 0x0100 Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	T3_INDEX							0	T2_INDEX						
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	T1_INDEX							0	T0_INDEX						
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 386. Commands list control register 1 (CLCR1)

Note: Tx_INDEX is always less than the total number of commands.

Table 485. CLCR1 field descriptions

Field	Description
1:7 T3_INDEX	Points to the (Ist command address-1) for Trigger 3.
9:15 T2_INDEX	Points to the (Ist command address-1) for Trigger 2.

Table 485. CLCR1 field descriptions (continued)

Field	Description
17:23 T1_INDEX	Points to the (Ist command address-1) for Trigger 1.
25:31 T0_INDEX	Points to the (Ist command address-1) for Trigger 0.

Note: First command for any trigger will start from CLR<Tx_INDEX+1> register.

35.10.11 Commands list control register 2 (CLCR2)

Offset: 0x0104 Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	T7_INDEX							0	T6_INDEX						
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	T5_INDEX							0	T4_INDEX						
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 387. Commands list control register 2 (CLCR2)

Table 486. CLCR2 field descriptions

Field	Description
1:7 T7_INDEX	Points to the (Ist command address-1) for Trigger 7.
9:15 T6_INDEX	Points to the (Ist command address-1) for Trigger 6.
17:23 T5_INDEX	Points to the (Ist command address-1) for Trigger 5.
25:31 T4_INDEX	Points to the (Ist command address-1) for Trigger 4.

Note: First command for any trigger will start from CLR<Tx_INDEX+1> register.

35.10.12 Commands list control register 3 (CLCR3)

This register is present only if number of supported triggers > 8.

Offset: 0x0108 Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	T11_INDEX							0	T10_INDEX						
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	T9_INDEX							0	T8_INDEX						
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 388. Commands list control register 3 (CLCR3)

Table 487. CLCR3 field descriptions

Field	Description
1:7 T11_INDEX	Points to the (Ist command address-1) for Trigger 11.
9:15 T10_INDEX	Points to the (Ist command address-1) for Trigger 10.
17:23 T9_INDEX	Points to the (Ist command address-1) for Trigger 9.
25:31 T8_INDEX	Points to the (Ist command address-1) for Trigger 8.

Note: First command for any trigger will start from CLR<Tx_INDEX+1> register.

35.10.13 Commands list control register 4 (CLCR4)

This register is present only if number of supported triggers > 12.

Offset: 0x010C Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	T15_INDEX							0	T14_INDEX						
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	T13_INDEX							0	T12_INDEX						
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 389. Commands list control register 4 (CLCR4)

Table 488. CLCR4 field descriptions

Field	Description
1:7 T15_INDEX	Points to the (Ist command address-1) for Trigger 15.
9:15 T14_INDEX	Points to the (Ist command address-1) for Trigger 14.
17:23 T13_INDEX	Points to the (Ist command address-1) for Trigger 13.
25:31 T12_INDEX	Points to the (Ist command address-1) for Trigger 12.

Note: First command for any trigger will start from CLR<Tx_INDEX+1> register.

35.10.14 Trigger handler control register 1 (THCR1)

Offset: 0x0110 Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	T3_E	T3_ETE	T3_T2E	T3_T1E	T3_ADCE	0	0	0	T2_E	T2_ETE	T2_T2E	T2_T1E	T2_ADCE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	T1_E	T1_ETE	T1_T2E	T1_T1E	T1_ADCE	0	0	0	T0_E	T0_ETE	T0_T2E	T0_T1E	T0_ADCE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 390. Trigger handler control register 1 (THCR1)

Table 489. THCR1 field descriptions

Field	Description
3 T3_E	Trigger 3 enable 0 disabled 1 enabled
4 T3_ETE	Trigger 3 External Trigger output enable 0 disabled 1 enabled
5 T3_T2E	Trigger 3 Timer 2 output enable 0 disabled 1 enabled
6 T3_T1E	Trigger 3 Timer 1 output enable 0 disabled 1 enabled

Table 489. THCR1 field descriptions (continued)

Field	Description
7 T3_ADCE	Trigger 3 ADC command output enable 0 disabled 1 enabled
11 T2_E	Trigger 2 enable 0 disabled 1 enabled
12 T2_ETE	Trigger 2 External Trigger output enable 0 disabled 1 enabled
13 T2_T2E	Trigger 2 Timer 2 output enable 0 disabled 1 enabled
14 T2_T1E	Trigger 2 Timer 1 output enable 0 disabled 1 enabled
15 T2_ADCE	Trigger 2 ADC command output enable 0 disabled 1 enabled
19 T1_E	Trigger 1 enable 0 disabled 1 enabled
20 T1_ETE	Trigger 1 External Trigger output enable 0 disabled 1 enabled
21 T1_T2E	Trigger 1 Timer 2 output enable 0 disabled 1 enabled
22 T1_T1E	Trigger 1 Timer 1 output enable 0 disabled 1 enabled
23 T1_ADCE	Trigger 1 ADC command output enable 0 disabled 1 enabled
27 T0_E	Trigger 0 enable 0 disabled 1 enabled
28 T0_ETE	Trigger 0 External Trigger output enable 0 disabled 1 enabled
29 T0_T2E	Trigger 0 Timer 2 output enable 0 disabled 1 enabled

Table 489. THCR1 field descriptions (continued)

Field	Description
30 T0_T1E	Trigger 0 Timer 1 output enable 0 disabled 1 enabled
31 T0_ADCE	Trigger 0 ADC command output enable 0 disabled 1 enabled

35.10.15 Trigger handler control register 2 (THCR2)

Offset: 0x0114 Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	T7_E	T7_ETE	T7_T2E	T7_T1E	T7_ADCE	0	0	0	T6_E	T6_ETE	T6_T2E	T6_T1E	T6_ADCE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	T5_E	T5_ETE	T5_T2E	T5_T1E	T5_ADCE	0	0	0	T4_E	T4_ETE	T4_T2E	T4_T1E	T4_ADCE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 391. Trigger handler control register 2 (THCR2)

Table 490. THCR2 field descriptions

Field	Description
3 T7_E	Trigger 7 enable 0 disabled 1 enabled
4 T7_ETE	Trigger 7 External Trigger output enable 0 disabled 1 enabled
5 T7_T2E	Trigger 7 Timer 2 output enable 0 disabled 1 enabled
6 T7_T1E	Trigger 7 Timer 1 output enable 0 disabled 1 enabled
7 T7_ADCE	Trigger 7 ADC command output enable 0 disabled 1 enabled

Table 490. THCR2 field descriptions (continued)

Field	Description
11 T6_E	Trigger 6 enable 0 disabled 1 enabled
12 T6_ETE	Trigger 6 External Trigger output enable 0 disabled 1 enabled
13 T6_T2E	Trigger 6 Timer 2 output enable 0 disabled 1 enabled
14 T6_T1E	Trigger 6 Timer 1 output enable 0 disabled 1 enabled
15 T6_ADCE	Trigger 6 ADC command output enable 0 disabled 1 enabled
19 T5_E	Trigger 5 enable 0 disabled 1 enabled
20 T5_ETE	Trigger 5 External Trigger output enable 0 disabled 1 enabled
21 T5_T2E	Trigger 5 Timer 2 output enable 0 disabled 1 enabled
22 T5_T1E	Trigger 5 Timer 1 output enable 0 disabled 1 enabled
23 T5_ADCE	Trigger 5 ADC command output enable 0 disabled 1 enabled
27 T4_E	Trigger 4 enable 0 disabled 1 enabled
28 T4_ETE	Trigger 4 External Trigger output enable 0 disabled 1 enabled
29 T4_T2E	Trigger 4 Timer 2 output enable 0 disabled 1 enabled

Table 490. THCR2 field descriptions (continued)

Field	Description
30 T4_T1E	Trigger 4 Timer 1 output enable 0 disabled 1 enabled
31 T4_ADCE	Trigger 4 ADC command output enable 0 disabled 1 enabled

35.10.16 Trigger handler control register 3 (THCR3)

This register is present only if number of supported triggers > 8.

Offset: 0x0118 Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	T11_E	T11_ETE	T11_T2E	T11_T1E	T11_ADCE	0	0	0	T10_E	T10_ETE	T10_T2E	T10_T1E	T10_ADCE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	T9_E	T9_ETE	T9_T2E	T9_T1E	T9_ADCE	0	0	0	T8_E	T8_ETE	T8_T2E	T8_T1E	T8_ADCE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 392. Trigger handler control register 3 (THCR3)

Table 491. THCR3 field descriptions

Field	Description
3 T11_E	Trigger 11 enable 0 disabled 1 enabled
4 T11_ETE	Trigger 11 External Trigger output enable 0 disabled 1 enabled
5 T11_T2E	Trigger 11 Timer 2 output enable 0 disabled 1 enabled
6 T11_T1E	Trigger 11 Timer 1 output enable 0 disabled 1 enabled
7 T11_ADCE	Trigger 11 ADC command output enable 0 disabled 1 enabled

Table 491. THCR3 field descriptions (continued)

Field	Description
11 T10_E	Trigger 10 enable 0 disabled 1 enabled
12 T10_ETE	Trigger 10 External Trigger output enable 0 disabled 1 enabled
13 T10_T2E	Trigger 10 Timer 2 output enable 0 disabled 1 enabled
14 T10_T1E	Trigger 10 Timer 1 output enable 0 disabled 1 enabled
15 T10_ADCE	Trigger 10 ADC command output enable 0 disabled 1 enabled
19 T9_E	Trigger 9 enable 0 disabled 1 enabled
20 T9_ETE	Trigger 9 External Trigger output enable 0 disabled 1 enabled
21 T9_T2E	Trigger 9 Timer 2 output enable 0 disabled 1 enabled
22 T9_T1E	Trigger 9 Timer 1 output enable 0 disabled 1 enabled
23 T9_ADCE	Trigger 9 ADC command output enable 0 disabled 1 enabled
27 T8_E	Trigger 8 enable 0 disabled 1 enabled
28 T8_ETE	Trigger 8 External Trigger output enable 0 disabled 1 enabled
29 T8_T2E	Trigger 8 Timer 2 output enable 0 disabled 1 enabled

Table 491. THCR3 field descriptions (continued)

Field	Description
30 T8_T1E	Trigger 8 Timer 1 output enable 0 disabled 1 enabled
31 T8_ADCE	Trigger 8 ADC command output enable 0 disabled 1 enabled

35.10.17 Trigger handler control register 4 (THCR4)

This register is present only if number of supported triggers > 12.

Offset: 0x011C Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	T15_E	T15_ETE	T15_T2E	T15_T1E	T15_ADCE	0	0	0	T14_E	T14_ETE	T14_T2E	T14_T1E	T14_ADCE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	T13_E	T13_ETE	T13_T2E	T13_T1E	T13_ADCE	0	0	0	T12_E	T12_ETE	T12_T2E	T12_T1E	T12_ADCE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 393. Trigger handler control register 4 (THCR4)

Table 492. THCR4 field descriptions

Field	Description
3 T15_E	Trigger 15 enable 0 disabled 1 enabled
4 T15_ETE	Trigger 15 External Trigger output enable 0 disabled 1 enabled
5 T15_T2E	Trigger 15 Timer 2 output enable 0 disabled 1 enabled
6 T15_T1E	Trigger 15 Timer 1 output enable 0 disabled 1 enabled

Table 492. THCR4 field descriptions (continued)

Field	Description
7 T15_ADCE	Trigger 15 ADC command output enable 0 disabled 1 enabled
11 T14_E	Trigger 14 enable 0 disabled 1 enabled
12 T14_ETE	Trigger 14 External Trigger output enable 0 disabled 1 enabled
13 T14_T2E	Trigger 14 Timer 2 output enable 0 disabled 1 enabled
14 T14_T1E	Trigger 14 Timer 1 output enable 0 disabled 1 enabled
15 T14_ADCE	Trigger 14 ADC command output enable 0 disabled 1 enabled
19 T13_E	Trigger 13 enable 0 disabled 1 enabled
20 T13_ETE	Trigger 13 External Trigger output enable 0 disabled 1 enabled
21 T13_T2E	Trigger 13 Timer 2 output enable 0 disabled 1 enabled
22 T13_T1E	Trigger 13 Timer 1 output enable 0 disabled 1 enabled
23 T13_ADCE	Trigger 13 ADC command output enable 0 disabled 1 enabled
27 T12_E	Trigger 12 enable 0 disabled 1 enabled
28 T12_ETE	Trigger 12 External Trigger output enable 0 disabled 1 enabled
29 T12_T2E	Trigger 12 Timer 2 output enable 0 disabled 1 enabled

Table 492. THCR4 field descriptions (continued)

Field	Description
30 T12_T1E	Trigger 12 Timer 1 output enable 0 disabled 1 enabled
31 T12_ADCE	Trigger 12 ADC command output enable 0 disabled 1 enabled

35.10.18 Commands list register *n* (CLR*n*)

The number of CLRn supported is 64.

This register is used to setup a command for the multiple ADCs.

Offset: 0x11C +n*0x4 (n=1 to 64) Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	CIR	LC	EC	0	0	0	0	0	0	0	0	0	FIFO_CH			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	ADC_CH							0	0	0	0	ADC_UNIT			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 394. Commands list register *n* (CLR*n*)

Table 493. CLR*n* field descriptions

Field	Description
0 CIR	Command Interrupt Request 0 disabled 1 enabled
1 LC	1 implies Last sequence for the trigger
2 EC	1 implies End of the concurrent conversion
12:15 FIFO_CH	FIFO number for the conversion. Max number supported is 8.
17:23 ADC_CH	ADC Channel number. Max number supported is 2^7.
28:31 ADC_UNIT	ADC unit number. Max number supported is 4.

35.10.19 Control register (CR)

The reset value must be corrected as per parameter CTU_NUM_FIFO.

CTU_NUM_FIFO=4 : Reset Value = 0x000F_0000,

CTU_NUM_FIFO=8 : Reset Value = 0x00FF_0000,

CTU_NUM_FIFO=12 : Reset Value = 0x0FFF_0000,

CTU_NUM_FIFO=16 : Reset Value = 0xFFFF_0000

The present design supports CTU_NUM_FIFO=8.

Offset: 0x0400 Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	EMPTY_CLR15	EMPTY_CLR14	EMPTY_CLR13	EMPTY_CLR12	EMPTY_CLR11	EMPTY_CLR10	EMPTY_CLR9	EMPTY_CLR8	EMPTY_CLR7	EMPTY_CLR6	EMPTY_CLR5	EMPTY_CLR4	EMPTY_CLR3	EMPTY_CLR2	EMPTY_CLR1	EMPTY_CLR0
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	DMA_EN15	DMA_EN14	DMA_EN13	DMA_EN12	DMA_EN11	DMA_EN10	DMA_EN9	DMA_EN8	DMA_EN7	DMA_EN6	DMA_EN5	DMA_EN4	DMA_EN3	DMA_EN2	DMA_EN1	DMA_EN0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 395. Control register (CR)

Table 494. CR field descriptions

Field	Description
0:15 EMPTY_CLR[15:0]	Empty Clear These bits reset the interrupt related to the empty flag even if the fifo is still empty. These bits are set by hardware with a positive edge on the empty flag and are reset by writing 1 in this field or writing a data in the fifo. When this bit is reset, FIFO empty interrupts are not generated even if FCRx.EMPTY_ENx & SRx.EMPTYx bits are set.
16:31 DMA_EN[15:0]	DMA Enable This bit enables the DMA interface for the FIFOx. 0 disabled 1 enabled

35.10.20 Control register 1 FIFO (FCR1)

Offset: 0x0404 Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	FIFO_OVERRUN_EN7	FIFO_OVERFLOW_EN7	FIFO_EMPTY_EN7	FIFO_FULL_EN7	FIFO_OVERRUN_EN6	FIFO_OVERFLOW_EN6	FIFO_EMPTY_EN6	FIFO_FULL_EN6	FIFO_OVERRUN_EN5	FIFO_OVERFLOW_EN5	FIFO_EMPTY_EN5	FIFO_FULL_EN5	FIFO_OVERRUN_EN4	FIFO_OVERFLOW_EN4	FIFO_EMPTY_EN4	FIFO_FULL_EN4
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	FIFO_OVERRUN_EN3	FIFO_OVERFLOW_EN3	FIFO_EMPTY_EN3	FIFO_FULL_EN3	FIFO_OVERRUN_EN2	FIFO_OVERFLOW_EN2	FIFO_EMPTY_EN2	FIFO_FULL_EN2	FIFO_OVERRUN_EN1	FIFO_OVERFLOW_EN1	FIFO_EMPTY_EN1	FIFO_FULL_EN1	FIFO_OVERRUN_EN0	FIFO_OVERFLOW_EN0	FIFO_EMPTY_EN0	FIFO_FULL_EN0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 396. Control register 1 FIFO (FCR1)

Table 495. FCR1 field descriptions

Field	Description
0 FIFO_OVERRUN_EN7	If this bit is set a FIFO_OVERRUN7 generates an interrupt.
1 FIFO_OVERFLOW_EN7	If this bit is set, a FIFO_OVERFLOW7 generates an interrupt.
2 FIFO_EMPTY_EN7	If this bit is set, a FIFO_EMPTY7 generates an interrupt.
3 FIFO_FULL_EN7	If this bit is set, a FIFO_FULL7 generates an interrupt.
4 FIFO_OVERRUN_EN6	If this bit is set a FIFO_OVERRUN6 generates an interrupt.
5 FIFO_OVERFLOW_EN6	If this bit is set, a FIFO_OVERFLOW6 generates an interrupt.
6 FIFO_EMPTY_EN6	If this bit is set, a FIFO_EMPTY6 generates an interrupt.
7 FIFO_FULL_EN6	If this bit is set, a FIFO_FULL6 generates an interrupt.

Table 495. FCR1 field descriptions (continued)

Field	Description
8 FIFO_OVERRUN_EN5	If this bit is set a FIFO_OVERRUN5 generates an interrupt.
9 FIFO_OVERFLOW_EN5	If this bit is set, a FIFO_OVERFLOW5 generates an interrupt.
10 FIFO_EMPTY_EN5	If this bit is set, a FIFO_EMPTY5 generates an interrupt.
11 FIFO_FULL_EN5	If this bit is set, a FIFO_FULL5 generates an interrupt.
12 FIFO_OVERRUN_EN4	If this bit is set a FIFO_OVERRUN4 generates an interrupt
13 FIFO_OVERFLOW_EN4	If this bit is set, a FIFO_OVERFLOW4 generates an interrupt.
14 FIFO_EMPTY_EN4	If this bit is set, a FIFO_EMPTY4 generates an interrupt.
15 FIFO_FULL_EN4	If this bit is set, a FIFO_FULL4 generates an interrupt.
16 FIFO_OVERRUN_EN3	If this bit is set a FIFO_OVERRUN3 generates an interrupt
17 FIFO_OVERFLOW_EN3	If this bit is set, a FIFO_OVERFLOW3 generates an interrupt.
18 FIFO_EMPTY_EN3	If this bit is set, a FIFO_EMPTY3 generates an interrupt.
19 FIFO_FULL_EN3	If this bit is set, a FIFO_FULL3 generates an interrupt.
20 FIFO_OVERRUN_EN2	If this bit is set a FIFO_OVERRUN2 generates an interrupt.
21 FIFO_OVERFLOW_EN2	If this bit is set, a FIFO_OVERFLOW2 generates an interrupt.
22 FIFO_EMPTY_EN2	If this bit is set, a FIFO_EMPTY2 generates an interrupt.
23 FIFO_FULL_EN2	If this bit is set, a FIFO_FULL2 generates an interrupt.
24 FIFO_OVERRUN_EN1	If this bit is set a FIFO_OVERRUN1 generates an interrupt.
25 FIFO_OVERFLOW_EN1	If this bit is set, a FIFO_OVERFLOW1 generates an interrupt.
26 FIFO_EMPTY_EN1	If this bit is set, a FIFO_EMPTY1 generates an interrupt.

Table 495. FCR1 field descriptions (continued)

Field	Description
27 FIFO_FULL_EN1	If this bit is set, a FIFO_FULL1 generates an interrupt.
28 FIFO_OVERRUN_EN0	If this bit is set a FIFO_OVERRUN0 generates an interrupt
29 FIFO_OVERFLOW_EN0	If this bit is set, a FIFO_OVERFLOW0 generates an interrupt.
30 FIFO_EMPTY_EN0	If this bit is set, a FIFO_EMPTY0 generates an interrupt.
31 FIFO_FULL_EN0	If this bit is set, a FIFO_FULL0 generates an interrupt.

35.10.21 Control register 2 FIFO (FCR2)

Offset: 0x0408 Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	FIFO_OVERRUN_EN15	FIFO_OVERFLOW_EN15	FIFO_EMPTY_EN15	FIFO_FULL_EN15	FIFO_OVERRUN_EN14	FIFO_OVERFLOW_EN14	FIFO_EMPTY_EN14	FIFO_FULL_EN14	FIFO_OVERRUN_EN13	FIFO_OVERFLOW_EN13	FIFO_EMPTY_EN13	FIFO_FULL_EN13	FIFO_OVERRUN_EN12	FIFO_OVERFLOW_EN12	FIFO_EMPTY_EN12	FIFO_FULL_EN12
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	FIFO_OVERRUN_EN11	FIFO_OVERFLOW_EN11	FIFO_EMPTY_EN11	FIFO_FULL_EN11	FIFO_OVERRUN_EN10	FIFO_OVERFLOW_EN10	FIFO_EMPTY_EN10	FIFO_FULL_EN10	FIFO_OVERRUN_EN9	FIFO_OVERFLOW_EN9	FIFO_EMPTY_EN9	FIFO_FULL_EN9	FIFO_OVERRUN_EN8	FIFO_OVERFLOW_EN8	FIFO_EMPTY_EN8	FIFO_FULL_EN8
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 397. Control register 2 FIFO (FCR2)

Note: This registers is available only when number of FIFOs > 8. If the supported FIFOs are less than 16 then the bits for unsupported FIFOs will be tied to '0' by design.

Table 496. FCR2 field descriptions

Field	Description
0 FIFO_OVERRUN_EN15	If this bit is set a FIFO_OVERRUN15 generates an interrupt.
1 FIFO_OVERFLOW_EN15	If this bit is set, a FIFO_OVERFLOW15 generates an interrupt.
2 FIFO_EMPTY_EN15	If this bit is set, a FIFO_EMPTY15 generates an interrupt.
3 FIFO_FULL_EN15	If this bit is set, a FIFO_FULL15 generates an interrupt.
4 FIFO_OVERRUN_EN14	If this bit is set a FIFO_OVERRUN14 generates an interrupt.
5 FIFO_OVERFLOW_EN14	If this bit is set, a FIFO_OVERFLOW14 generates an interrupt.
6 FIFO_EMPTY_EN14	If this bit is set, a FIFO_EMPTY14 generates an interrupt.
7 FIFO_FULL_EN14	If this bit is set, a FIFO_FULL14 generates an interrupt.
8 FIFO_OVERRUN_EN13	If this bit is set a FIFO_OVERRUN13 generates an interrupt.
9 FIFO_OVERFLOW_EN13	If this bit is set, a FIFO_OVERFLOW13 generates an interrupt.
10 FIFO_EMPTY_EN13	If this bit is set, a FIFO_EMPTY13 generates an interrupt.
11 FIFO_FULL_EN13	If this bit is set, a FIFO_FULL13 generates an interrupt.
12 FIFO_OVERRUN_EN12	If this bit is set a FIFO_OVERRUN12 generates an interrupt.
13 FIFO_OVERFLOW_EN12	If this bit is set, a FIFO_OVERFLOW12 generates an interrupt.
14 FIFO_EMPTY_EN12	If this bit is set, a FIFO_EMPTY12 generates an interrupt.
15 FIFO_FULL_EN12	If this bit is set, a FIFO_FULL12 generates an interrupt.
16 FIFO_OVERRUN_EN11	If this bit is set a FIFO_OVERRUN11 generates an interrupt.
17 FIFO_OVERFLOW_EN11	If this bit is set, a FIFO_OVERFLOW11 generates an interrupt.
18 FIFO_EMPTY_EN11	If this bit is set, a FIFO_EMPTY11 generates an interrupt.

Table 496. FCR2 field descriptions (continued)

Field	Description
19 FIFO_FULL_EN11	If this bit is set, a FIFO_FULL11 generates an interrupt.
20 FIFO_OVERRUN_EN10	If this bit is set a FIFO_OVERRUN10 generates an interrupt.
21 FIFO_OVERFLOW_EN10	If this bit is set, a FIFO_OVERFLOW10 generates an interrupt.
22 FIFO_EMPTY_EN10	If this bit is set, a FIFO_EMPTY10 generates an interrupt.
23 FIFO_FULL_EN10	If this bit is set, a FIFO_FULL10 generates an interrupt.
24 FIFO_OVERRUN_EN9	If this bit is set a FIFO_OVERRUN9 generates an interrupt.
25 FIFO_OVERFLOW_EN9	If this bit is set, a FIFO_OVERFLOW9 generates an interrupt.
26 FIFO_EMPTY_EN9	If this bit is set, a FIFO_EMPTY9 generates an interrupt.
27 FIFO_FULL_EN9	If this bit is set, a FIFO_FULL9 generates an interrupt.
28 FIFO_OVERRUN_EN8	If this bit is set a FIFO_OVERRUN8 generates an interrupt.
29 FIFO_OVERFLOW_EN8	If this bit is set, a FIFO_OVERFLOW8 generates an interrupt.
30 FIFO_EMPTY_EN8	If this bit is set, a FIFO_EMPTY8 generates an interrupt.
31 FIFO_FULL_EN8	If this bit is set, a FIFO_FULL8 generates an interrupt.

35.10.22 THRESHOLD register 1 (TH1)

Offset: 0x040C Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	THRESHOLD3								THRESHOLD2							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	THRESHOLD1								THRESHOLD0							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 398. THRESHOLD register 1 (TH1)

Table 497. TH1 field descriptions

Field	Description
0:7 THRESHOLD3	If the number of words in the fifo 3 becomes higher than the value set, the fifo_overflow3 bit is set to one.
8:15 THRESHOLD2	If the number of words in the fifo 2 becomes higher than the value set, the fifo_overflow 2 bit is set to one.
16:23 THRESHOLD1	If the number of words in the fifo 1 becomes higher than the value set, the fifo_overflow1 bit is set to one.
24:31 THRESHOLD0	If the number of words in the fifo 0 becomes higher than the value set, the fifo_overflow0 bit is set to one.

35.10.23 THRESHOLD register 2 (TH2)

Offset: 0x0410 Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	THRESHOLD7								THRESHOLD6							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	THRESHOLD5								THRESHOLD4							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 399. THRESHOLD register 2 (TH2)

Table 498. TH2 field descriptions

Field	Description
0:7 THRESHOLD7	If the number of words in the fifo 7 becomes higher than the value set, the fifo_overflow7 bit is set to one.
8:15 THRESHOLD6	If the number of words in the fifo 6 becomes higher than the value set, the fifo_overflow6 bit is set to one.
16:23 THRESHOLD5	If the number of words in the fifo 5 becomes higher than the value set, the fifo_overflow5 bit is set to one.
24:31 THRESHOLD4	If the number of words in the fifo 4 becomes higher than the value set, the fifo_overflow4 bit is set to one.

35.10.24 THRESHOLD register 3 (TH3)

Offset: 0x0414 Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	THRESHOLD11								THRESHOLD10							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	THRESHOLD9								THRESHOLD8							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 400. THRESHOLD register 3 (TH3)

Table 499. TH3 field descriptions

Field	Description
0:7 THRESHOLD11	If the number of words in the fifo 11 becomes higher than the value set, the fifo_overflow11 bit is set to one.
8:15 THRESHOLD10	If the number of words in the fifo 10 becomes higher than the value set, the fifo_overflow10 bit is set to one.
16:23 THRESHOLD9	If the number of words in the fifo 9 becomes higher than the value set, the fifo_overflow9 bit is set to one.
24:31 THRESHOLD8	If the number of words in the fifo 8 becomes higher than the value set, the fifo_overflow8 bit is set to one.

35.10.25 THRESHOLD register 4 (TH4)

Offset: 0x0418 Access: User Read/Write

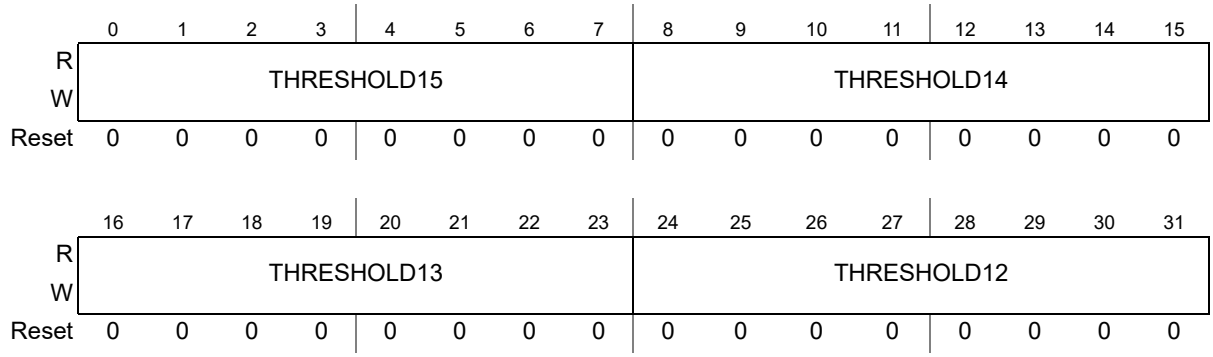


Figure 401. THRESHOLD register 4 (TH4)

Table 500. TH4 field descriptions

Field	Description
0:7 THRESHOLD15	If the number of words in the fifo 15 becomes higher than the value set, the fifo_overflow15 bit is set to one.
8:15 THRESHOLD14	If the number of words in the fifo 14 becomes higher than the value set, the fifo_overflow14 bit is set to one.
16:23 THRESHOLD13	If the number of words in the fifo 13 becomes higher than the value set, the fifo_overflow13 bit is set to one.
24:31 THRESHOLD12	If the number of words in the fifo 12 becomes higher than the value set, the fifo_overflow12 bit is set to one.

35.10.26 Status register 1 (ST1)

Offset: 0x041C Access: User Read/Write

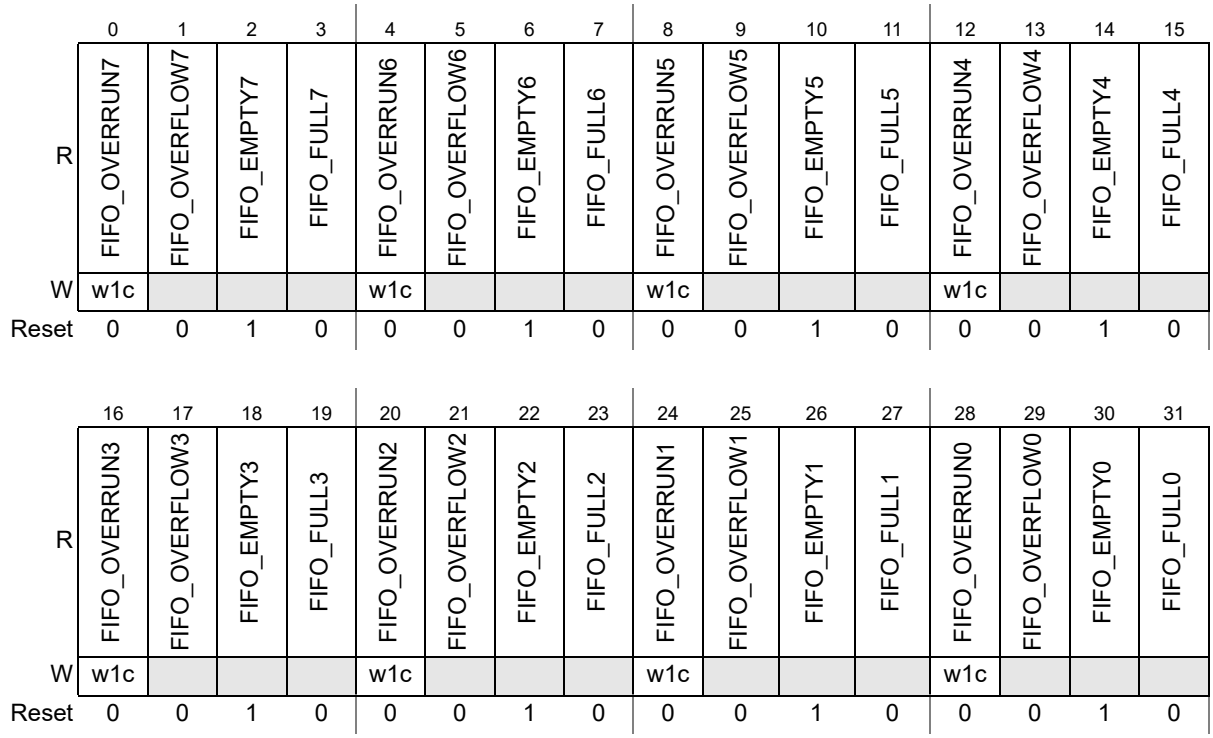


Figure 402. Status register 1 (ST1)

Table 501. ST1 field descriptions

Field	Description
0 FIFO_OVERRUN7	This bit is set to 1 if a write operation occurred when corresponding FIFO_FULL flag is set. This flag can be cleared by writing ‘1’ to it.
1 FIFO_OVERFLOW7	This bit is automatically put to one if the number of words became higher than the value set in the threshold 7. This bit is a read only bit.
2 FIFO_EMPTY7	This bit is set to one if the fifo 7 is empty. This bit is a read only bit.
3 FIFO_FULL7	This bit is set to one if the fifo 7 is full. This bit is a read only bit.
4 FIFO_OVERRUN6	This bit is set to 1 if a write operation occurred when corresponding FIFO_FULL flag is set. This flag can be cleared by writing ‘1’ to it.
5 FIFO_OVERFLOW6	This bit is automatically put to one if the number of words became higher than the value set in the threshold 6. This bit is a read only bit.
6 FIFO_EMPTY6	This bit is set to one if the fifo 6 is empty. This bit is a read only bit.
7 FIFO_FULL6	This bit is set to one if the fifo 6 is full. This bit is a read only bit.

Table 501. ST1 field descriptions (continued)

Field	Description
8 FIFO_OVERRUN5	This bit is set to 1 if a write operation occurred when corresponding FIFO_FULL flag is set. This flag can be cleared by writing ‘1’ to it.
9 FIFO_OVERFLOW5	This bit is automatically put to one if the number of words became higher than the value set in the threshold 5. This bit is a read only bit.
10 FIFO_EMPTY5	This bit is set to one if the fifo 5 is empty. This bit is a read only bit.
11 FIFO_FULL5	This bit is set to one if the fifo 5 is full. This bit is a read only bit.
12 FIFO_OVERRUN4	This bit is set to 1 if a write operation occurred when corresponding FIFO_FULL flag is set. This flag can be cleared by writing ‘1’ to it.
13 FIFO_OVERFLOW4	This bit is automatically put to one if the number of words became higher than the value set in the threshold 4. This bit is a read only bit.
14 FIFO_EMPTY4	This bit is set to one if the fifo 4 is empty. This bit is a read only bit.
15 FIFO_FULL4	This bit is set to one if the fifo 4 is full. This bit is a read only bit.
16 FIFO_OVERRUN3	This bit is set to 1 if a write operation occurred when corresponding FIFO_FULL flag is set. This flag can be cleared by writing ‘1’ to it.
17 FIFO_OVERFLOW3	This bit is automatically put to one if the number of words became higher than the value set in the threshold 3. This bit is a read only bit.
18 FIFO_EMPTY3	This bit is set to one if the fifo 3 is empty. This bit is a read only bit.
19 FIFO_FULL3	This bit is set to one if the fifo 3 is full. This bit is a read only bit.
20 FIFO_OVERRUN2	This bit is set to 1 if a write operation occurred when corresponding FIFO_FULL flag is set. This flag can be cleared by writing ‘1’ to it.
21 FIFO_OVERFLOW2	This bit is automatically put to one if the number of words became higher than the value set in the threshold 2. This bit is a read only bit.
22 FIFO_EMPTY2	This bit is set to one if the fifo 2 is empty. This bit is a read only bit.
23 FIFO_FULL2	This bit is set to one if the fifo 2 is full. This bit is a read only bit.
24 FIFO_OVERRUN1	This bit is set to 1 if a write operation occurred when corresponding FIFO_FULL flag is set. This flag can be cleared by writing ‘1’ to it.
25 FIFO_OVERFLOW1	This bit is automatically put to one if the number of words became higher than the value set in the threshold 1. This bit is a read only bit.
26 FIFO_EMPTY1	This bit is set to one if the fifo 1 is empty. This bit is a read only bit.

Table 501. ST1 field descriptions (continued)

Field	Description
27 FIFO_FULL1	This bit is set to one if the fifo 1 is full. This bit is a read only bit.
28 FIFO_OVERRUN0	This bit is set to 1 if a write operation occurred when corresponding FIFO_FULL flag is set. This flag can be cleared by writing '1' to it.
29 FIFO_OVERFLOW0	This bit is automatically put to one if the number of words became higher than the value set in the threshold 0. This bit is a read only bit.
30 FIFO_EMPTY0	This bit is set to one if the fifo 0 is empty. This bit is a read only bit.
31 FIFO_FULL0	This bit is set to one if the fifo 0 is full. This bit is a read only bit.

35.10.27 Status register 2 (ST2)

Offset: 0x0420 Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	FIFO_OVERRUN15	FIFO_OVERFLOW15	FIFO_EMPTY15	FIFO_FULL15	FIFO_OVERRUN14	FIFO_OVERFLOW14	FIFO_EMPTY14	FIFO_FULL14	FIFO_OVERRUN13	FIFO_OVERFLOW13	FIFO_EMPTY13	FIFO_FULL13	FIFO_OVERRUN12	FIFO_OVERFLOW12	FIFO_EMPTY12	FIFO_FULL12
W	w1c				w1c				w1c				w1c			
Reset	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	FIFO_OVERRUN11	FIFO_OVERFLOW11	FIFO_EMPTY11	FIFO_FULL11	FIFO_OVERRUN10	FIFO_OVERFLOW10	FIFO_EMPTY10	FIFO_FULL10	FIFO_OVERRUN9	FIFO_OVERFLOW9	FIFO_EMPTY9	FIFO_FULL9	FIFO_OVERRUN8	FIFO_OVERFLOW8	FIFO_EMPTY8	FIFO_FULL8
W	w1c				w1c				w1c				w1c			
Reset	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0

Figure 403. Status register 2 (ST2)

Note: This registers is available only when number of FIFOs > 8.

Table 502. ST2 field descriptions

Field	Description
0 FIFO_OVERRUN15	This bit is set to 1 if a write operation occurred when corresponding FIFO_FULL flag is set. This flag can be cleared by writing ‘1’ to it.
1 FIFO_OVERFLOW15	This bit is automatically put to one if the number of words became higher than the value set in the threshold 15. This bit is a read only bit.
2 FIFO_EMPTY15	This bit is set to one if the fifo 15 is empty. This bit is a read only bit.
3 FIFO_FULL15	This bit is set to one if the fifo 15 is full. This bit is a read only bit.
4 FIFO_OVERRUN14	This bit is set to 1 if a write operation occurred when corresponding FIFO_FULL flag is set. This flag can be cleared by writing ‘1’ to it.
5 FIFO_OVERFLOW14	This bit is automatically put to one if the number of words became higher than the value set in the threshold 14. This bit is a read only bit.
6 FIFO_EMPTY14	This bit is set to one if the fifo 14 is empty. This bit is a read only bit.
7 FIFO_FULL14	This bit is set to one if the fifo 14 is full. This bit is a read only bit.
8 FIFO_OVERRUN13	This bit is set to 1 if a write operation occurred when corresponding FIFO_FULL flag is set. This flag can be cleared by writing ‘1’ to it.
9 FIFO_OVERFLOW13	This bit is automatically put to one if the number of words became higher than the value set in the threshold 13. This bit is a read only bit.
10 FIFO_EMPTY13	This bit is set to one if the fifo 13 is empty. This bit is a read only bit.
11 FIFO_FULL13	This bit is set to one if the fifo 13 is full. This bit is a read only bit.
12 FIFO_OVERRUN12	This bit is set to 1 if a write operation occurred when corresponding FIFO_FULL flag is set. This flag can be cleared by writing ‘1’ to it.
13 FIFO_OVERFLOW12	This bit is automatically put to one if the number of words became higher than the value set in the threshold 12. This bit is a read only bit.
14 FIFO_EMPTY12	This bit is set to one if the fifo 12 is empty. This bit is a read only bit.
15 FIFO_FULL12	This bit is set to one if the fifo 12 is full. This bit is a read only bit.
16 FIFO_OVERRUN11	This bit is set to 1 if a write operation occurred when corresponding FIFO_FULL flag is set. This flag can be cleared by writing ‘1’ to it.
17 FIFO_OVERFLOW11	This bit is automatically put to one if the number of words became higher than the value set in the threshold 11. This bit is a read only bit.
18 FIFO_EMPTY11	This bit is set to one if the fifo 11 is empty. This bit is a read only bit.

Table 502. ST2 field descriptions (continued)

Field	Description
19 FIFO_FULL11	This bit is set to one if the fifo 11 is full. This bit is a read only bit.
20 FIFO_OVERRUN10	This bit is set to 1 if a write operation occurred when corresponding FIFO_FULL flag is set. This flag can be cleared by writing '1' to it.
21 FIFO_OVERFLOW10	This bit is automatically put to one if the number of words became higher than the value set in the threshold 10. This bit is a read only bit.
22 FIFO_EMPTY10	This bit is set to one if the fifo 10 is empty. This bit is a read only bit.
23 FIFO_FULL10	This bit is set to one if the fifo 10 is full. This bit is a read only bit.
24 FIFO_OVERRUN9	This bit is set to 1 if a write operation occurred when corresponding FIFO_FULL flag is set. This flag can be cleared by writing '1' to it.
25 FIFO_OVERFLOW9	This bit is automatically put to one if the number of words became higher than the value set in the threshold 9. This bit is a read only bit.
26 FIFO_EMPTY9	This bit is set to one if the fifo 9 is empty. This bit is a read only bit.
27 FIFO_FULL9	This bit is set to one if the fifo 9 is full. This bit is a read only bit.
28 FIFO_OVERRUN8	This bit is set to 1 if a write operation occurred when corresponding FIFO_FULL flag is set. This flag can be cleared by writing '1' to it.
29 FIFO_OVERFLOW8	This bit is automatically put to one if the number of words became higher than the value set in the threshold 8. This bit is a read only bit.
30 FIFO_EMPTY8	This bit is set to one if the fifo 8 is empty. This bit is a read only bit.
31 FIFO_FULL8	This bit is set to one if the fifo 8 is full. This bit is a read only bit.

35.10.28 FIFO right aligned register *n* (FR*n*)

Offset: 0x0424 + n*0x4 (n=0 to 7) Access: User Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	ADC_UNIT				0	N_CH						
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	DATA															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 404. FIFO right aligned register *n* (FR*n*)

Table 503. FR*n* field descriptions

Field	Description
4:7 ADC_UNIT	This field is used to indicate from which ADC the data is coming from 0000 The data is coming from ADC_0 0001 The data is coming from ADC_1 0010 The data is coming from ADC_2 0011 The data is coming from ADC_3 0100 The data is coming from ADC_4 0101 The data is coming from ADC_5 0110 The data is coming from ADC_6 0111 The data is coming from ADC_7 1000 The data is coming from ADC_8 1001 The data is coming from ADC_9 1010 The data is coming from ADC_10 1011 The data is coming from ADC_11 1100 The data is coming from ADC_12 1101 The data is coming from ADC_13 1110 The data is coming from ADC_14 1111 The data is coming from ADC_15
9:15 N_CH	Number of Channels This field is reloaded every read access and it indicates the ADC channel number for which conversion has taken place. Maximum number of channels supported = 2^7.
16:31 DATA	This field is used to store the converted data as is coming from ADC. The width of the Data is static with its unused bits tied as '0'. The data width supported is 12.

35.10.29 FIFO left aligned register *n* (FL*n*)

Offset: 0x0464 + *n**0x4 (*n*=0 to 7) Access: User Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	ADC_UNIT				0	N_CH						
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	DATA														
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 405. FIFO left aligned register *n* (FL*n*)

Table 504. FL*n* field descriptions

Field	Description
4:7 ADC_UNIT[3:0]	This field is used for indicate from which ADC the data is coming from 0000 The data is coming from ADC_0 0001 The data is coming from ADC_1 0010 The data is coming from ADC_2 0011 The data is coming from ADC_3 0100 The data is coming from ADC_4 0101 The data is coming from ADC_5 0110 The data is coming from ADC_6 0111 The data is coming from ADC_7 1000 The data is coming from ADC_8 1001 The data is coming from ADC_9 1010 The data is coming from ADC_10 1011 The data is coming from ADC_11 1100 The data is coming from ADC_12 1101 The data is coming from ADC_13 1110 The data is coming from ADC_14 1111 The data is coming from ADC_15
9:15 N_CH	Number of Channel This field is reloaded every read access and it indicates the ADC channel number for which conversion has taken place. Maximum number of channels supported = 2^7.
17:31 DATA	This field is used to store the converted data as is coming from ADC. The width of the Data is static with its unused bits tied as '0'. The DATA starts with its MSB at bit position 14. The data width supported is 12 and DATA will be stored at bits 14-3.

35.10.30 Cross triggering unit error flag register (CTUEFR)

Offset: 0x0500 Access: User Read/Write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	SAF_ERR_4	SAF_ERR_3	SAF_ERR_2	SAF_ERR_1	CRC_ERR	MAX_CON	ET_OE	Error_compare	T2_OE	T1_OE	ADC_OE	TGS_OSM	MRS_O	ICE	SM_TO	MRS_RE
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 406. Cross triggering unit error flag register (CTUEFR)

Table 505. CTUEFR field descriptions

Field	Description
16 SAF_ERR_4	Conversion time safety error bit 4 (For 4th parallel conversion) 1 If set then the time between the start and the end of conversion is not in expected range 0 The conversion time is within expected range
17 SAF_ERR_3	Conversion time safety error bit 3 (For 3th parallel conversion) 1 If set then the time between the start and the end of conversion is not in expected range 0 The conversion time is within expected range
18 SAF_ERR_2	Conversion time safety error bit 2 (For 2nd parallel conversion) 1 If set then the time between the start and the end of conversion is not in expected range 0 The conversion time is within expected range
19 SAF_ERR_1	Conversion time safety error bit 1 (For 1st parallel conversion/Single conversion) 1 If set then the time between the start and the end of conversion is not in expected range 0 The conversion time is within expected range
20 CRC_ERR	CRC comparison error bit 1 The programmed CRC is not matching with the computed CRC 0 no CRC comparison error occured
21 MAX_CON	Maximum concurrent command error flag 1 If number of concurrent commands exceeds static design parameter that is 4 0 No error
22 ET_OE	External Trigger generation Overrun Error
23 Error_compare	This bit is set to one if the counter reach the TGSCCR register. It is cleared writing one in this field.
24 T2_OE	Timer 2 trigger generation Overrun Error
25 T1_OE	Timer 1 trigger generation Overrun Error
26 ADC_OE	ADC command generation Overrun Error
27 TGS_OSM	TGS Overrun in Sequential Mode
28 MRS_O	Master Reload Signal Overrun
29 ICE	Invalid Command Error
30 SM_TO	Trigger Overrun (more than 8 EV) in TGS Sequential Mode
31 MRS_RE	Master Reload Signal Reload Error

35.10.31 Cross triggering unit interrupt flag register (CTUIFR)

Offset: 0x0504 Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	T15_I	T14_I	T13_I	T12_I	T11_I	T10_I	T9_I	T8_I	T7_I	T6_I	T5_I	T4_I	T3_I	T2_I	T1_I	T0_I
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	ADC_I	0	0	0	0	0	MRS_I	0
W									w1c						w1c	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 407. Cross triggering unit interrupt flag register (CTUIFR)

The design supports up to 8 triggers. The bits for unsupported triggers will always be read as ‘0’.

Table 506. CTUIFR field descriptions

Field	Description
0 T15_I	Trigger 15 interrupt flag
1 T14_I	Trigger 14 interrupt flag
2 T13_I	Trigger 13 interrupt flag
3 T12_I	Trigger 12 interrupt flag
4 T11_I	Trigger 11 interrupt flag
5 T10_I	Trigger 10 interrupt flag
6 T9_I	Trigger 9 interrupt flag
7 T8_I	Trigger 8 interrupt flag
8 T7_I	Trigger 7 interrupt flag
9 T6_I	Trigger 6 interrupt flag
10 T5_I	Trigger 5 interrupt flag

Table 506. CTUIFR field descriptions (continued)

Field	Description
11 T4_I	Trigger 4 interrupt flag
12 T3_I	Trigger 3 interrupt flag
13 T2_I	Trigger 2 interrupt flag
14 T1_I	Trigger 1 interrupt flag
15 T0_I	Trigger 0 interrupt flag
24 ADC_I	ADC command interrupt flag
30 MRS_I	MRS Interrupt flag

35.10.32 Cross triggering unit interrupt/DMA register (CTUIR)

Offset: 0x0508 Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	T15_IE	T14_IE	T13_IE	T12_IE	T11_IE	T10_IE	T9_IE	T8_IE	T7_IE	T6_IE	T5_IE	T4_IE	T3_IE	T2_IE	T1_IE	T0_IE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	SAF_ERR_EN_4	SAF_ERR_EN_3	SAF_ERR_EN_2	SAF_ERR_EN_1	0	0	0	0	0	0	0	0	DMA_DE	MRS_DMAE	MRS_IE	IEE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 408. Cross triggering unit interrupt/DMA register (CTUIR)

The design supports up to 8 triggers. The bits for unsupported triggers will be reserved and will always be read as ‘0’.

Table 507. CTUIR field descriptions

Field	Description
0 T15_IE	Trigger 15 interrupt enable 0 disabled 1 enabled
1 T14_IE	Trigger 14 interrupt enable 0 disabled 1 enabled
2 T13_IE	Trigger 13 interrupt enable 0 disabled 1 enabled
3 T12_IE	Trigger 12 interrupt enable 0 disabled 1 enabled
4 T11_IE	Trigger 11 interrupt enable 0 disabled 1 enabled
5 T10_IE	Trigger 10 interrupt enable 0 disabled 1 enabled
6 T9_IE	Trigger 9 interrupt enable 0 disabled 1 enabled
7 T8_IE	Trigger 8 interrupt enable 0 disabled 1 enabled
8 T7_IE	Trigger 7 interrupt enable 0 disabled 1 enabled
9 T6_IE	Trigger 6 interrupt enable 0 disabled 1 enabled
10 T5_IE	Trigger 5 interrupt enable 0 disabled 1 enabled
11 T4_IE	Trigger 4 interrupt enable 0 disabled 1 enabled
12 T3_IE	Trigger 3 interrupt enable 0 disabled 1 enabled
13 T2_IE	Trigger 2 interrupt enable 0 disabled 1 enabled

Table 507. CTUIR field descriptions (continued)

Field	Description
14 T1_IE	Trigger 1 interrupt enable 0 disabled 1 enabled
15 T0_IE	Trigger 0 interrupt enable 0 disabled 1 enabled
16 SAF_ERR_EN_4	SAF_ERR_EN_4 1 Enable Conversion time counter interrupt for 4th parallel conversion (SAF_ERR_4) 0 Disable Conversion time counter interrupt for 4th parallel conversion
17 SAF_ERR_EN_3	SAF_ERR_EN_3 1 Enable Conversion time counter interrupt for 3th parallel conversion (SAF_ERR_3) 0 Disable Conversion time counter interrupt for 3th parallel conversion
18 SAF_ERR_EN_2	SAF_ERR_EN_2 1 Enable Conversion time counter interrupt for 2nd parallel conversion (SAF_ERR_2) 0 Disable Conversion time counter interrupt for 2nd parallel conversion
19 SAF_ERR_EN_1	SAF_ERR_EN_1 1 Enable Conversion time counter interrupt for 1st parallel/single conversion (SAF_ERR_1) 0 Disable Conversion time counter interrupt for 1st parallel/single conversion
28 DMA_DE	If this bit is set, a dma done is like a write in the GRE bit. If this bit is reset then GRE bit has to be written (if something is changed in the double buffered register) otherwise an error will be generated on the next master reload.
29 MRS_DMAE	DMA transfer Enable on MRS occurrence
30 MRS_IE	MRS Interrupt Enable
31 IEE	Interrupt Error This bit enable all the or-ed interrupt flags and generate an interrupt if or-ed output is set and if it is enabled.

35.10.33 Control on-time register (COTR)

Offset: 0x050C Access: User Read/Write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	COTR							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 409. Control on-time register (COTR)

Table 508. COTR field descriptions

Field	Description
24:31 COTR	Control ON Time Register and Guard Time to be used for external trigger output

35.10.34 Cross triggering unit control register (CTUCR)

Offset: 0x0510 Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	T15_SG(1)	T14_SG(1)	T13_SG(1)	T12_SG(1)	T11_SG(1)	T10_SG(1)	T9_SG(1)	T8_SG(1)	T7_SG(1)	T6_SG(1)	T5_SG(1)	T4_SG(1)	T3_SG(1)	T2_SG(1)	T1_SG(1)	T0_SG(1)
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	CTU_CRC_EN	CTU_ADC_reset(1)	CTU_ODIS	FILTER_enable	CGRE	FGRE	MRS_SG(1)	GRE	TGSISR_RE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. Writing a one has some effect on the module, but it always reads as zero.

Figure 410. Cross triggering unit control register (CTUCR)

Table 509. CTUCR field descriptions

Field	Description
0 T15_SG	Trigger 15 software generated when written ‘1’. Writing a one has some effect on the module, but it always reads as zero.
1 T14_SG	Trigger 14 software generated when written ’1’. Writing a one has some effect on the module, but it always reads as zero.
2 T13_SG	Trigger 13 software generated when written ’1’. Writing a one has some effect on the module, but it always reads as zero.
3 T12_SG	Trigger 12 software generated when written ’1’. Writing a one has some effect on the module, but it always reads as zero.
4 T11_SG	Trigger 11 software generated when written ’1’. Writing a one has some effect on the module, but it always reads as zero.
5 T10_SG	Trigger 10 software generated when written ’1’. Writing a one has some effect on the module, but it always reads as zero.
6 T9_SG	Trigger 9 software generated when written ’1’. Writing a one has some effect on the module, but it always reads as zero.
7 T8_SG	Trigger 8 software generated when written ’1’. Writing a one has some effect on the module, but it always reads as zero.
8 T7_SG	Trigger 7 software generated when written ’1’. Writing a one has some effect on the module, but it always reads as zero.
9 T6_SG	Trigger 6 software generated when written ’1’. Writing a one has some effect on the module, but it always reads as zero.
10 T5_SG	Trigger 5 software generated when written ’1’. Writing a one has some effect on the module, but it always reads as zero.
11 T4_SG	Trigger 4 software generated when written ’1’. Writing a one has some effect on the module, but it always reads as zero.
12 T3_SG	Trigger 3 software generated when written ’1’. Writing a one has some effect on the module, but it always reads as zero.
13 T2_SG	Trigger 2 software generated when written ’1’. Writing a one has some effect on the module, but it always reads as zero.
14 T1_SG	Trigger 1 software generated when written ’1’. Writing a one has some effect on the module, but it always reads as zero.
15 T0_SG	Trigger 0 software generated when written ’1’. Writing a one has some effect on the module, but it always reads as zero.
23 CTU_CRC_EN	0 CRC computation not enabled 1 CRC computation enabled
24 CTU_ADC_reset	CTU ADC state machine reset when written ‘1’. Writing a one has some effect on the module, but it always reads as zero.
25 CTU_ODIS	CTU output disable 0 output enabled 1 output disabled

Table 509. CTUCR field descriptions (continued)

Field	Description
26 FILTER_enable	This bit is used to synchronize the Filter register value. It is set by the user and automatically reset when it is reloaded in MT_clk domain. The reloaded bit in MT_clk domain gets reset by MRS.
27 CGRE	Clear GRE
28 FGRE	GRE Flag (some double buffered register has been written)
29 MRS_SG	MRS software generated when written ‘1’. Writing a one has some effect on the module, but it always reads as zero.
30 GRE	General Reload Enable
31 TGSISR_RE	TGS Input Selection Register Reload Enable. Is set by the user and reset automatically when its reloaded in MT_clk domain. The reloaded bit in MT_clk domain gets reset by MRS.

35.10.35 Cross triggering unit digital filter (CTU_FILTER)

Offset: 0x0514 Access: User Read/Write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	FILTER_VALUE							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 411. Cross triggering unit digital filter (CTU_FILTER)

Table 510. CTU_FILTER field descriptions

Field	Description
24:31 FILTER_VALUE	If the ext_signal is latched filter value time at 1 the external value is considered 1, if is latched filtered value time at zero is considered zero. Any pulse less than FILTER_VALUE * MT_clk will be filtered.

35.10.36 Cross triggering unit configuration status register 1 (CTU_CFGSR1)

Offset: 0x0518 Access: User Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	ADC_CH_WIDTH		NUM_TRIG		NUM_FIFO		MAX_CONCURRENT				
W																
Reset	0	0	0	0	0	1	1	0	1	0	1	0	0	1	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	NUM_COMMANDS								ADC_DATA_W			NUM_ADC				
W																
Reset	0	1	0	0	0	0	0	0	0	1	0	0	0	1	0	0

Figure 412. Cross triggering unit configuration status register 1 (CTU_CFGSR1)

Table 511. CTU_CFGSR1 field descriptions

Field	Description
5:6 ADC_CH_WIDTH	Channel width of the ADCs Numberof channels = $2^{ADC_CH_WIDTH}$ (The design supports channel width = 7) 00 4 01 5 10 6 11 7
7:8 NUM_TRIG	Number of internal triggers (The design supports channel width = 8) 00 Not used 01 8 triggers 10 12 triggers 11 16 triggers
9:10 NUM_FIFO	Number of FIFOs (The design supports channel width = 8) 00 4 FIFOs 01 8 FIFOs 10 12 FIFOs 11 16 FIFOs
11:15 MAX_CONCURRENT	Maximum number of concurrent conversions possible (maximum possible is equal to number of ADCs) = 4
16:23 NUM_COMMANDS	Number of CLRn registers (maximum value = 64)

Table 511. CTU_CFGSR1 field descriptions (continued)

Field	Description
24:26 ADC_DATA_W	Width of ADC converted data output (The design supports date width = 12) 000 8 bit 001 10 bit 010 12 bit 011 14 bit 100 16 bit
27:31 NUM_ADC	Number of ADCs supported by design (The design supports 4 ADCs)

35.10.37 Cross triggering unit configuration status register 2 (CTU_CFGSR2)

Offset: 0x051C Access: User Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	FIFO15_DEPTH		FIFO14_DEPTH		FIFO13_DEPTH		FIFO12_DEPTH		FIFO11_DEPTH		FIFO10_DEPTH		FIFO9_DEPTH		FIFO8_DEPTH	
W																
Reset(1)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	FIFO7_DEPTH		FIFO6_DEPTH		FIFO5_DEPTH		FIFO4_DEPTH		FIFO3_DEPTH		FIFO2_DEPTH		FIFO1_DEPTH		FIFO0_DEPTH	
W																
Reset(1)	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

1. The reset value is device dependent. Please see the *Chapter 7: Device configuration* for details.

Figure 413. CTU_CFGSR2 register

The bits related to unused FIFOs will always be read as ‘0’. The design supports only 8 FIFOs.

Table 512. CTU_CFGSR2 field descriptions

Field	Description
0:1 FIFO15_DEPTH	Number of words in FIFO 15 (maximum 16 words possible) (The design supports 16) 00 2 words 01 4 words 10 8 words 11 16 words
2:3 FIFO14_DEPTH	Number of words in FIFO 14 (maximum 16 words possible) (The design supports 16) 00 2 words 01 4 words 10 8 words 11 16 words
4:5 FIFO13_DEPTH	Number of words in FIFO 13 (maximum 16 words possible) (The design supports 16) 00 2 words 01 4 words 10 8 words 11 16 words
6:7 FIFO12_DEPTH	Number of words in FIFO 12 (maximum 16 words possible) (The design supports 16) 00 2 words 01 4 words 10 8 words 11 16 words
8:9 FIFO11_DEPTH	Number of words in FIFO 11 (maximum 16 words possible) (The design supports 16) 00 2 words 01 4 words 10 8 words 11 16 words
10:11 FIFO10_DEPTH	Number of words in FIFO 10 (maximum 16 words possible) (The design supports 16) 00 2 words 01 4 words 10 8 words 11 16 words
12:13 FIFO9_DEPTH	Number of words in FIFO 9(maximum 16 words possible) (The design supports 16) 00 2 words 01 4 words 10 8 words 11 16 words

Table 512. CTU_CFGSR2 field descriptions (continued)

Field	Description
14:15 FIFO8_DEPTH	Number of words in FIFO 8 (maximum 16 words possible) (The design supports 16) 00 2 words 01 4 words 10 8 words 11 16 words
16:17 FIFO7_DEPTH	Number of words in FIFO 7 (maximum 16 words possible) (The design supports 16) 00 2 words 01 4 words 10 8 words 11 16 words
18:19 FIFO6_DEPTH	Number of words in FIFO 6 (maximum 16 words possible) (The design supports 16) 00 2 words 01 4 words 10 8 words 11 16 words
20:21 FIFO5_DEPTH	Number of words in FIFO 5 (maximum 16 words possible) (The design supports 16) 00 2 words 01 4 words 10 8 words 11 16 words
22:23 FIFO4_DEPTH	Number of words in FIFO 4 (maximum 16 words possible) (The design supports 16) 00 2 words 01 4 words 10 8 words 11 16 words
24:25 FIFO3_DEPTH	Number of words in FIFO 3 (maximum 16 words possible) (The design supports 16) 00 2 words 01 4 words 10 8 words 11 16 words
26:27 FIFO2_DEPTH	Number of words in FIFO 2 (maximum 16 words possible) (The design supports 16) 00 2 words 01 4 words 10 8 words 11 16 words

Table 512. CTU_CFGSR2 field descriptions (continued)

Field	Description
28:29 FIFO1_DEPTH	Number of words in FIFO 1 (maximum 16 words possible) (The design supports 16) 00 2 words 01 4 words 10 8 words 11 16 words
30:31 FIFO0_DEPTH	Number of words in FIFO 0 (maximum 16 words possible) (The design supports 16) 00 2 words 01 4 words 10 8 words 11 16 words

35.10.38 Cross triggering unit expected value n (CTU_EXPECTED*n)*

Offset: 0x051C + *n**0x4 (*n*=1 to 4) Access: User Read/Write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	CTU_EXPECTED_x															
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Figure 414. Cross triggering unit expected value n (CTU_EXPECTED*n*)

Table 513. CTU_EXPECTED*n* field descriptions

Field	Description
16:31 CTU_EXPECTED_x	This value is the number of system clock cycle needed for the conversion time for x^{th} parallel conversion. For single conversion only CTU_EXPECTED_1 register is used.

35.10.39 Cross triggering unit counter range (CTU_CNT_RANGE*n*)

Offset: 0x052C + *n**0x4 (n=1 to 4) Access: User Read/Write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	CTU_CNT_RANGE_x							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 415. Cross triggering unit counter range (CTU_CNT_RANGE*n*)

Table 514. CTU_CNT_RANGE*n* field descriptions

Field	Description
24:31 CTU_CNT_RANGE_x	Setting at '1' at any bit masks the same bit of the CTU_EXPECTED_x register and of the internal counter. This is used to specify a range of counter values for the expected conversion time. For single conversion only CTU_CNT_RANGE_1 register is used.

35.10.40 CTU expected CRC register (CTU_EXP_CRC_VAL)

Offset: 0x0540 Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	CTU_EXP_CRC_VAL															
W																
Reset	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	CTU_EXP_CRC_VAL															
W																
Reset	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1

Figure 416. CTU expected CRC register (CTU_EXP_CRC_VAL)

Table 515. CTU_EXP_CRC_VAL field descriptions

Field	Description
0:31 CTU_EXP_CRC_VAL	This register provides the expected 32-bit CRC value to be compared at the end of every MRS cycle

35.10.41 CTU computed CRC register (CTU_COMP_CRC_VAL)

Offset: 0x0544 Access: User Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	CTU_COMP_CRC_VAL															
W																
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	CTU_COMP_CRC_VAL															
W																
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Figure 417. CTU computed CRC register (CTU_COMP_CRC_VAL)

Table 516. CTU_COMP_CRC_VAL field descriptions

Field	Description
0:31 CTU_COMP_CRC_VAL	This register provides the computed 32-bit CRC value as is computed by the internal CRC controller. It can be checked by software for any debugging purpose.

35.10.42 CTU Mask CRC computation register (CTU_MASK_CRC_VAL)

Offset: 0x0548 Access: User Read/Write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	MASK_ TIMESTAMP			MASK_ext_TRG	MASK_eTIMER2_TRG	MASK_eTIMER1_TRG	MASK_ADC_TRG	MASK_ERR	MASK_PUSH	MASK_ADC_FIFO	MASK_ADC_CH	MASK_ADC_NUM
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 418. CTU Mask CRC computation register (CTU_MASK_CRC_VAL)

Table 517. CTU_MASK_CRC_VAL field descriptions

Field	Description
20:22 MASK_TIMESTAMP	These bits are used to mask the TGS_Counter[15:0] bits for CRC calculation. They will be used for debug purposes. 000 TGS_Counter used for CRC calculation 111 TGS_Counter is not used for CRC calculation. Other values are reserved.
23 MASK_ext_TRG	1 ext_trigger output will not be used in CRC computation. Input to CRC will be 1'b0 always 0 ext_trigger output will be used in CRC computation
24 MASK_eTIMER2_TRG	1 eTimer2 trigger output will not be used in CRC computation. Input to CRC will be 1'b0 always 0 eTimer2 trigger output will be used in CRC computation
25 MASK_eTIMER1_TRG	1 eTimer1 trigger output will not be used in CRC computation. Input to CRC will be 1'b0 always 0 eTimer1 trigger output will be used in CRC computation
26 MASK_ADC_TRG	1 ADC trigger output will not be used in CRC computation. Input to CRC will be 1'b0 always 0 ADC trigger output will be used in CRC computation
27 MASK_ERR	1 Error Flags will not be used in CRC computation. Input to CRC will be 16'h0 always 0 Error flags will be used in CRC computation
28 MASK_PUSH	1 ctu_push signals will not be used in CRC computation. Input to CRC will be 16'h0 always 0 ctu_push signals will be used in CRC computation (This flag control usage of ctu_push for all ADCs. Either all ctu_push's will be used or none)
29 MASK_ADC_FIFO	1 ADC FIFO information will not be used in CRC computation. Input to CRC will be 4'h0 always 0 ADC FIFO information will be used in CRC computation (This flag control usage of FIFO numbers for all ADCs. Either all FIFO# will be used or none)
30 MASK_ADC_CH	1 ADC channel information will not be used in CRC computation. Input to CRC will be 7'b0 always 0 ADC channel information will be used in CRC computation (This flag control usage of ADC CH_NUM for all ADCs. Either all will be used or none)
31 MASK_ADC_NUM	1 Triggered ADC information will not be used in CRC compuration. Input to CRC will be 4'b0 0 Triggered ADC information will be used in CRC computation

36 Motor Control Pulse Width Modulator Module (FlexPWM)

36.1 Introduction

36.1.1 Overview

The pulse width modulator module (FlexPWM) contains one or more PWM submodules each of which is set up to control a single half-bridge power stage. There may also be one or more fault channels.

This PWM is capable of controlling most motor types: AC induction motors (ACIM), Permanent Magnet AC motors (PMAC), both brushless (BLDC) and brush DC motors (BDC), switched (SRM) and variable reluctance motors (VRM), and stepper motors.

36.1.2 Features

- 16 bits of resolution for center, edge aligned, and asymmetrical PWMs
- PWM outputs can operate as complimentary pairs or independent channels
- Can accept signed numbers for PWM generation
- Independent control of both edges of each PWM output
- Synchronization to external hardware or other PWM supported
- Double buffered PWM registers
  - Integral reload rates from 1 to 16
  - Half cycle reload capability
- Multiple output trigger events can be generated per PWM cycle via hardware
- Support for double switching PWM outputs
- Fault inputs can be assigned to control multiple PWM outputs
- Programmable filters for fault inputs
- Independently programmable PWM output polarity
- Independent top and bottom deadtime insertion
- Each complementary pair can operate with its own PWM frequency and deadtime values
- Individual software-control for each PWM output
- All outputs can be programmed to change simultaneously via a "Force Out" event
- PWMX pin can optionally output a third PWM signal from each submodule
- Channels not used for PWM generation can be used for buffered output compare functions
- Channels not used for PWM generation can be used for input capture functions
- Enhanced dual edge capture functionality
- The option to supply the source for each complementary PWM signal pair from any of the following:
  - External digital pin
  - Internal timer channel
  - External ADC input, taking into account values set in ADC high and low limit registers

36.1.3 Modes of Operation

Care must be exercised when using this module in certain chip operating modes. Some motors (such 3-phase AC motors) require regular software updates for proper operation. Failure to do so could result in destroying the motor or inverter. Because of this, PWM outputs are placed in their inactive states in STOP mode, and optionally under WAIT/HALT mode and debug mode. PWM outputs will be reactivated (assuming they were active to begin with) when these modes are exited.

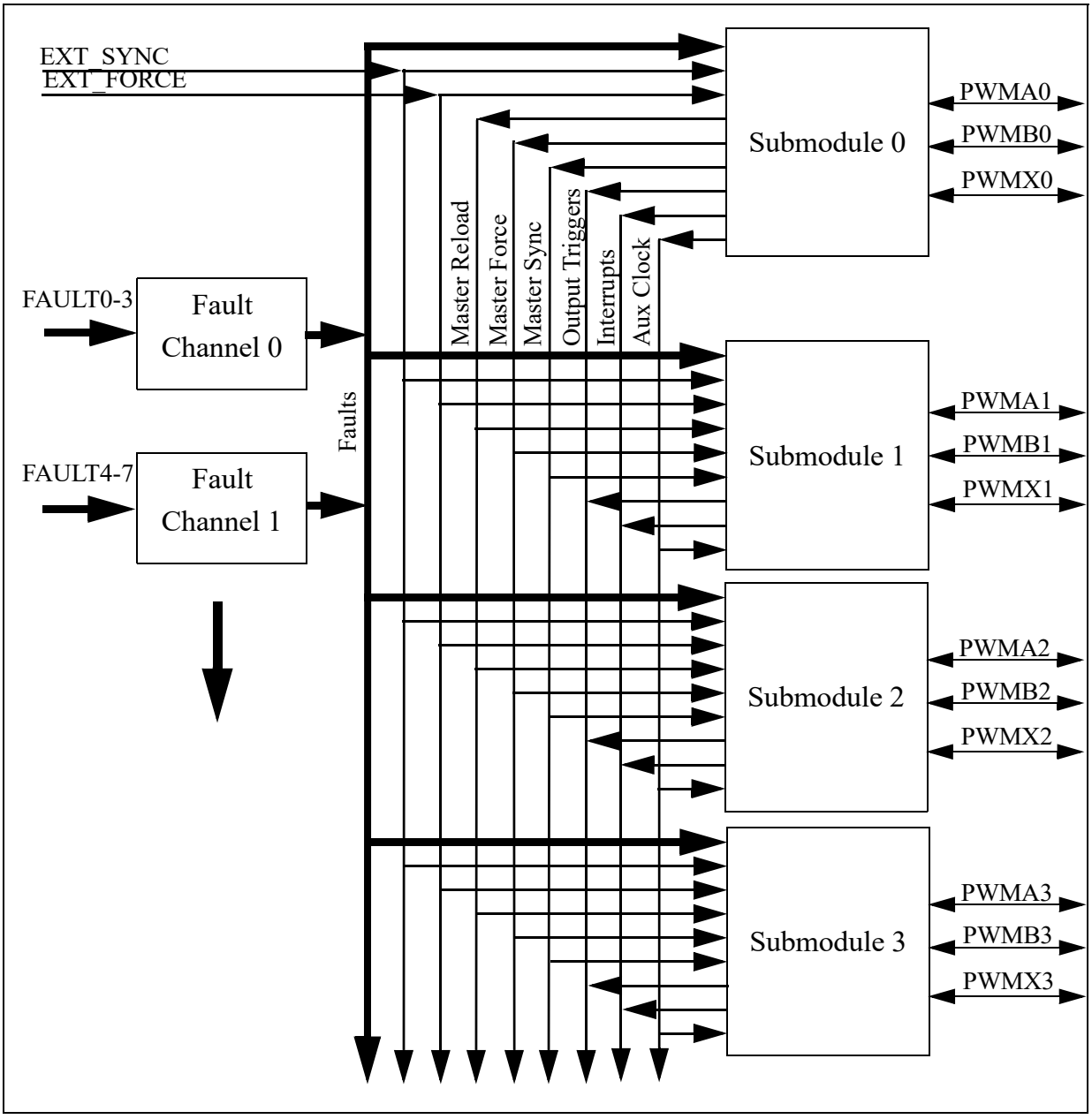
Table 518. Modes When PWM Operation is Restricted

Mode	Description
STOP	Peripheral and CPU clocks are stopped. PWM outputs are driven inactive.
WAIT/HALT	CPU clocks are stopped while peripheral clocks continue to run. PWM outputs are driven inactive as a function of the WAITEN bit. WAIT mode may be referred to as HALT mode at the SoC level.
DEBUG	CPU and peripheral clocks continue to run, but CPU maybe stalled for periods of time. PWM outputs are driven inactive as a function of the DBGEN bit.

36.1.4 Block Diagrams

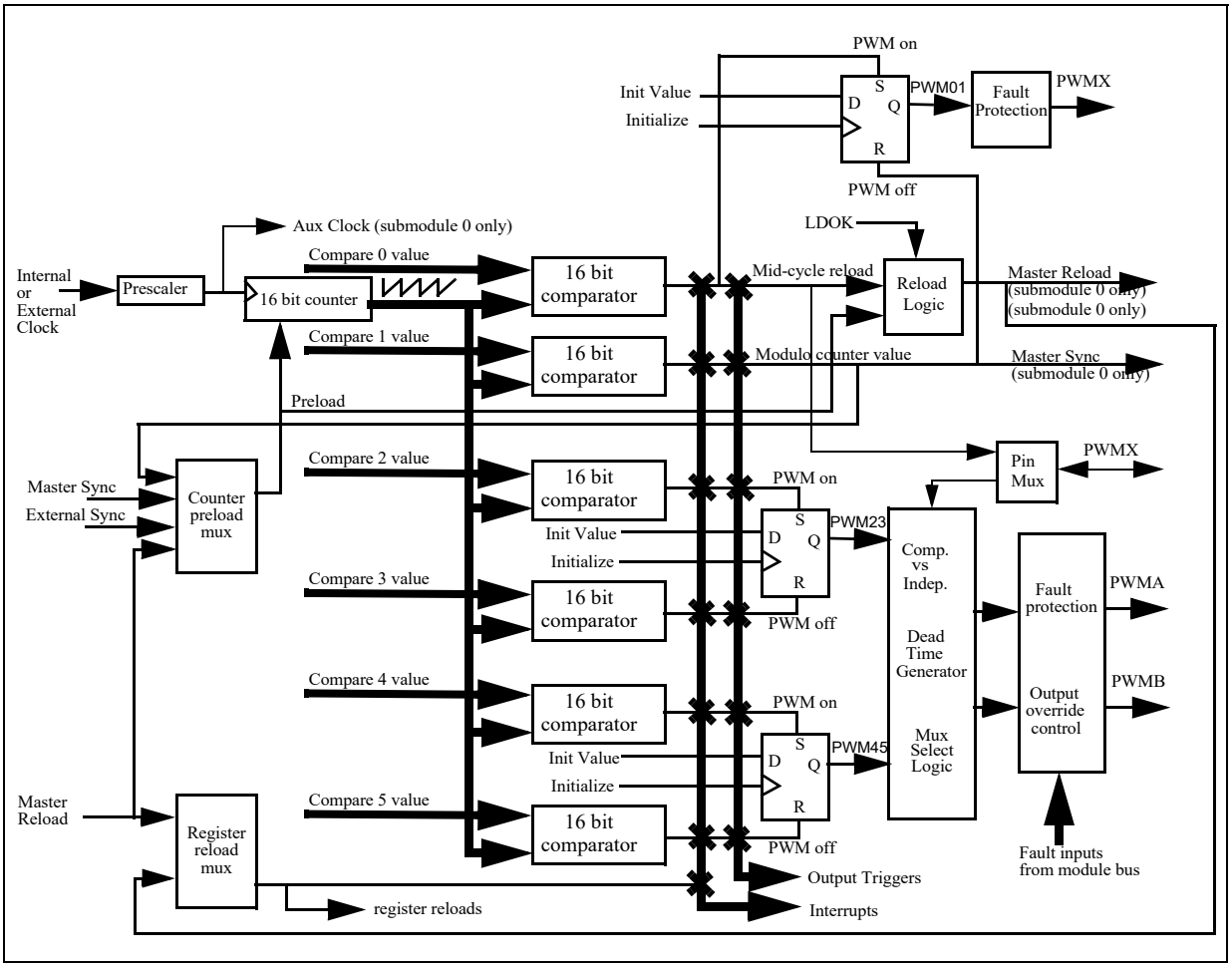
36.1.4.1 Module Level

Figure 419. PWM Block Diagram



36.1.4.2 PWM Submodule

Figure 420. PWM Submodule Block Diagram



36.2 External Signal Descriptions

The PWM module has external pins named PWMA[n], PWMB[n], PWMX[n], FAULT[n], EXT_SYNC, EXT_FORCE, EXTA[n], and EXTB[n]. The PWM module also has an on chip input called EXT_CLK and an on-chip output called OUT_TRIG[n].

36.2.1 PWMA[n] and PWMB[n] - External PWM Pair

These pins are the output pins of the PWM channels. They can be independent PWM signals or a complementary pair. When not needed as an output, they can be used as inputs to the input capture circuitry.

36.2.2 PWMX[n] - Auxiliary PWM signal

These pins are the auxiliary output pins of the PWM channels. They can be independent PWM signals. When not needed as an output, they can be used as inputs to the input capture circuitry or they can be used to generate the IPOL bit during deadtime correction.

36.2.3 FAULT[n] - Fault inputs

These are input pins for disabling selected PWM outputs.

36.2.4 EXT_SYNC - External Synchronization Signal

This input signal allows a source external to the PWM to initialize the PWM counter. In this manner the PWM can be synchronized to external circuitry.

36.2.5 EXT_FORCE - External Output Force Signal

This input signal allows a source external to the PWM to force an update of the PWM outputs. In this manner the PWM can be synchronized to external circuitry. An example would be to simultaneously switch all of the PWM outputs on a commutation boundary for trapezoidal control of a BLDC motor. The boundary can be established via external logic or an on-chip timer.

36.2.6 EXTA[n] and EXTB[n] - Alternate PWM Control Signals

These pins allow an alternate source to control the PWMA and PWMB outputs although typically, the EXTA input will be used for the generation of a complementary pair. Typical connections include ADC results registers, TMR outputs, GPIO inputs, and comparator outputs. The SIM chapter of the chip system spec will describe the possible connections and the bits to control any muxing.

36.2.7 OUT_TRIG0[n] and OUT_TRIG1[n] - Output Triggers

These outputs allow the PWM submodules to control timing of the ADC conversions. See *Section 36.3.16: Output Trigger Control register n (TCTRLn)* for a description of how to enable these outputs and how the compare registers match up to the output triggers.

36.2.8 EXT_CLK - External Clock Signal

This on-chip input signal allows an on-chip source external to the PWM (typically a Timer) to control the PWM clocking. In this manner the PWM can be synchronized to the Timer. This signal must be generated synchronously to the PWM's clock since it is not resynchronized in the PWM.

36.2.9 EXT_SWITCH - External Switch Signal

This input signal allows a source external to the PWM to control the PWM generation source within submodule1. This signal only functions on submodule1 and selects the source of the generated PWM signal to be either submodule1 or submodule0.

36.3 Memory Map and Registers

The address of a register is the sum of a base address and an address offset. The base address is defined at the core level and the address offset is defined at the module level. There are a set of registers for each PWM submodule, for the configuration logic, and for each Fault channel.

Registers are only accessible using 16- or 32-bit accesses. Byte accesses are not allowed.

These registers are repeated for each PWM submodule. The base address of submodule 0 is the same as the base address for the PWM module as a whole. The base address of submodule 1 is $50. This is the base address of the PWM module plus an offset based on the number of registers in a submodule. The base address of submodule 2 is equal to the base address of submodule 1 plus this same offset.

36.3.1 Module Memory Map

Table 519 is the FlexPWM memory map.

The actual availability and number of registers and their configuration are chip-specific. For this information, see "Device Configuration" chapter.

Table 519. FlexPWM memory map

Address offset	Register	Location
PWM submodule registers (*n* = 0 to 3)		
0x0000 + *n**0x50	Counter register *n* (CNT*n*)	*Section 36.3.2*
0x0002 + *n**0x50	Initial Count register *n* (INIT*n*)	*Section 36.3.3*
0x0004 + *n**0x50	Control 2 register *n* (CTRL2*n*)	*Section 36.3.4*
0x0006 + *n**0x50	Control 1 register *n* (CTRL1*n*)	*Section 36.3.5*
0x0008 + *n**0x50	Value 0 register *n* (VAL0*n*)	*Section 36.3.6*
0x000A + *n**0x50	Value 1 register *n* (VAL1*n*)	*Section 36.3.7*
0x000C + *n**0x50	Value 2 register *n* (VAL2*n*)	*Section 36.3.8*
0x000E + *n**0x50	Value 3 register *n* (VAL3*n*)	*Section 36.3.9*
0x0010 + *n**0x50	Value 4 register *n* (VAL4*n*)	*Section 36.3.10*
0x0012 + *n**0x50	Value 5 register *n* (VAL5*n*)	*Section 36.3.11*
0x0014 + *n**0x50– 0x0015 + *n**0x50	Reserved	
0x0016 + *n**0x50– 0x0017 + *n**0x50	Reserved	
0x0018 + *n**0x50	Output Control register *n* (OCTRL*n*)	*Section 36.3.12*
0x001A + *n**0x50	Status register *n* (STS*n*)	*Section 36.3.13*
0x001C + *n**0x50	Interrupt Enable register *n* (INTEN*n*)	*Section 36.3.14*
0x001E + *n**0x50	DMA Enable register *n* (DMAEN*n*)	*Section 36.3.15*
0x0020 + *n**0x50	Output Trigger Control register *n* (TCTRL*n*)	*Section 36.3.16*
0x0022 + *n**0x50	Fault Disable Mapping register *n* (DISMAP*n*)	*Section 36.3.17*
0x0024 + *n**0x50	Deadtime Count 0 register *n* (DTCNT0*n*)	*Section 36.3.18*
0x0026 + *n**0x50	Deadtime Count 1 register *n* (DTCNT1*n*)	*Section 36.3.19*
0x0028 + *n**0x50	Capture Control A register *n* (CAPTCTRLA*n*)	*Section 36.3.20*
0x002A + *n**0x50	Capture Compare A register *n* (CAPTCMPA*n*)	*Section 36.3.21*
0x002C + *n**0x50	Capture Control B register *n* (CAPTCTRLB*n*)	*Section 36.3.22*

Table 519. FlexPWM memory map (continued)

Address offset	Register	Location
0x002E + *n**0x50	Capture Compare B register *n* (CAPTCMPB*n*)	*Section 36.3.23*
0x0030 + *n**0x50	Capture Control X register *n* (CAPTCTRLX*n*)	*Section 36.3.24*
0x0032 + *n**0x50	Capture Compare X register *n* (CAPTCMPX*n*)	*Section 36.3.25*
0x0034 + *n**0x50	Capture Value 0 register *n* (CVAL0*n*)	*Section 36.3.26*
0x0036 + *n**0x50	Capture Value 0 Cycle register *n* (CVAL0CYC*n*)	*Section 36.3.27*
0x0038 + *n**0x50	Capture Value 1 register *n* (CVAL1*n*)	*Section 36.3.28*
0x003A + *n**0x50	Capture Value 1 Cycle register *n* (CVAL1CYC*n*)	*Section 36.3.29*
0x003C + *n**0x50	Capture Value 2 register *n* (CVAL2*n*)	*Section 36.3.30*
0x003E + *n**0x50	Capture Value 2 Cycle register *n* (CVAL2CYC*n*)	*Section 36.3.31*
0x0040 + *n**0x50	Capture Value 3 register *n* (CVAL3*n*)	*Section 36.3.32*
0x0042 + *n**0x50	Capture Value 3 Cycle register *n* (CVAL3CYC*n*)	*Section 36.3.33*
0x0044 + *n**0x50	Capture Value 4 register *n* (CVAL4*n*)	*Section 36.3.34*
0x0046 + *n**0x50	Capture Value 4 Cycle register *n* (CVAL4CYC*n*)	*Section 36.3.35*
0x0048 + *n**0x50	Capture Value 5 register *n* (CVAL5*n*)	*Section 36.3.36*
0x004A + *n**0x50	Capture Value 5 Cycle register *n* (CVAL5CYC*n*)	*Section 36.3.37*
0x013C–0x013F	Reserved	
Configuration Logic Registers (see *Section 36.3.38*)		
0x0140	Output Enable register (OUTEN)	*Section 36.3.38.1*
0x0142	Output Mask register (MASK)	*Section 36.3.38.2*
0x0144	Software Controlled Output register (SWCOUT)	*Section 36.3.38.3*
0x0146	Deadtime Source Select register (DTSRCSEL)	*Section 36.3.38.4*
0x0148	Master Control register (MCTRL)	*Section 36.3.38.5*
0x014A–0x014B	Reserved	
Fault Channel registers (see *Section 36.3.39*)		
0x014C	Fault Control register (FCTRL)	*Section 36.3.39.1*
0x014E	Fault Status register (FSTS)	*Section 36.3.39.2*
0x0150	Fault Filter register (FFILT)	*Section 36.3.39.3*

36.3.2 Counter register *n* (CNT*n*)

This read-only register displays the state of the signed 16-bit submodule counter.

Offset: 0x0000 + *n**0x50 (*n*=0 to 3) Access: Read-only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	CNT															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 421. Counter register *n* (CNT*n*)

Table 520. CNT*n* field descriptions

Field	Description
0:15 CNT	Counter Register These read only bits show the current value of the PWM submodule counter.

36.3.3 Initial Count register n (INITn)

The 16-bit signed value in this buffered, read/write register defines the initial count value for the PWM in PWM clock periods. This is the value loaded into the submodule counter when local sync, master sync, or master reload is asserted (based on the value of INIT_SEL) or when FORCE is asserted and force init is enabled. For PWM operation, the buffered contents of this register are loaded into the counter at the start of every PWM cycle.

Offset: 0x0002 + *n**0x50 (*n*=0 to 3) Access: R/W

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	INIT															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 422. Initial Count register *n* (INIT*n*)

Table 521. INIT*n* field descriptions

Field	Description
0:15 INIT	Initial Count Register This 16-bit signed value is the initial count value for the PWM in PWM clock periods

Note: The INIT register is buffered. The value written does not take effect until the MCTRL[LDOK] bit is set and the next PWM load cycle begins or CTRL1[LDMOD] is set. This register cannot be written when MCTRL[LDOK] is set. Reading INIT reads the value in a buffer and not necessarily the value the PWM generator is currently using.

36.3.4 Control 2 register n (CTRL2n)

Offset: 0x0004 + *n**0x50 (*n*=0 to 3) Access: R/W

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DBGEN	WAITEN	INDEP	PWM23_INIT	PWM45_INIT	PWMX_INIT	INIT_SEL		FRCEN	0	FORCE_SEL			RELOAD_SEL	CLK_SEL	
W										FORCE						
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 423. Control 2 register *n* (CTRL2*n*)

Table 522. CTRL2*n* field descriptions

Field	Description
0 DBGEN	Debug Enable When set to one, the PWM will continue to run while the chip is in debug mode. If the device enters debug mode and this bit is zero, then the PWM outputs will be disabled until debug mode is exited. At that point the PWM pins will resume operation as programmed in the PWM registers. For certain types of motors (such as 3-phase AC), it is imperative that this bit be left in its default state (in which the PWM is disabled in debug mode). Failure to do so could result in damage to the motor or inverter. For other types of motors (example: DC motors), this bit might safely be set to one, enabling the PWM in debug mode. The key point is PWM parameter updates will not occur in debug mode. Any motors requiring such updates should be disabled during debug mode. If in doubt, leave this bit set to zero.
1 WAITEN	WAIT Enable When set to one, the PWM will continue to run while the chip is in WAIT/HALT mode. In this mode, the peripheral clock continues to run but the CPU clock does not. If the device enters WAIT/HALT mode and this bit is zero, then the PWM outputs will be disabled until WAIT/HALT mode is exited. At that point the PWM pins will resume operation as programmed in the PWM registers. For certain types of motors (such as 3-phase AC), it is imperative that this bit be left in its default state (in which the PWM is disabled in WAIT/HALT mode). Failure to do so could result in damage to the motor or inverter. For other types of motors (example: DC motors), this bit might safely be set to one, enabling the PWM in WAIT/HALT mode. The key point is PWM parameter updates will not occur in this mode. Any motors requiring such updates should be disabled during WAIT/HALT mode. If in doubt, leave this bit set to zero. For certain types of motors (such as 3-phase AC), it is imperative that this bit be left in its default state (in which the PWM is disabled in WAIT/HALT mode). Failure to do so could result in damage to the motor or inverter. For other types of motors (example: DC motors), this bit might safely be set to one, enabling the PWM in WAIT/HALT mode. The key point is PWM parameter updates will not occur in this mode. Any motors requiring such updates should be disabled during WAIT/HALT mode. If in doubt, leave this bit set to zero.

Table 522. CTRL2*n* field descriptions (continued)

Field	Description
2 INDEP	Independent or Complementary Pair Operation This bit determines if the PWMA and PWMB channels will be independent PWMs or a complementary PWM pair. 1 PWMA and PWMB outputs are independent PWMs 0 PWMA and PWMB form a complementary PWM pair
3 PWM23_INIT	PWM23 Initial Value This read/write bit determines the initial value for PWM23 and the value to which it is forced when FORCE_INIT, *Figure 478: PWM Generation Hardware*, is asserted.
4 PWM45_INIT	PWM45 Initial Value This read/write bit determines the initial value for PWM45 and the value to which it is forced when FORCE_INIT, *Figure 478: PWM Generation Hardware*, is asserted.
5 PWMX_INIT	PWMX Initial Value This read/write bit determines the initial value for PWMX and the value to which it is forced when FORCE_INIT, *Figure 478: PWM Generation Hardware*, is asserted.
6:7 INIT_SEL	Initialization Control Select These read/write bits control the source of the INIT signal which goes to the counter. 00 Local sync (PWMX) causes initialization 01 Master reload from submodule 0 causes initialization. This setting should not be used in submodule 0 as it will force the INIT signal to logic 0 10 Master sync from submodule 0 causes initialization. This setting should not be used in submodule 0 as it will force the INIT signal to logic 0 11 EXT_SYNC causes initialization
8 FRCEN	Force Initialization Enable This bit allows the FORCE_OUT signal to initialize the counter without regard to the signal selected by INIT_SEL. This is a software controlled initialization. 1 Initialization from a Force Out event is enabled 0 Initialization from a Force Out event is disabled
9 FORCE	Force Initialization If the FORCE_SEL bits are set to 000, writing a 1 to this bit results in a Force Out event. This causes the following actions to be taken: – The PWMA and PWMB output pins will assume values based on the SEL23 and SEL45 bits. – If the FRCEN bit is set, the counter value will be initialized with the INIT register value.

Table 522. CTRL2*n* field descriptions (continued)

Field	Description
10:12 FORCE_SEL	Force Source Select This read/write bit determines the source of the FORCE OUTPUT signal for this submodule. 000 The local force signal, FORCE, from this submodule is used to force updates 001 The master force signal from submodule 0 is used to force updates. This setting should not be used in submodule 0 as it will hold the FORCE OUTPUT signal to logic 0 010 The local reload signal from this submodule is used to force updates without regard to the state of LDOK 011 The master reload signal from submodule0 is used to force updates if LDOK is set. This setting should not be used in submodule0 as it will hold the FORCE OUTPUT signal to logic 0 100 The local sync signal from this submodule is used to force updates 101 The master sync signal from submodule0 is used to force updates. This setting should not be used in submodule0 as it will hold the FORCE OUTPUT signal to logic 0 110 The external force signal, EXT_FORCE, from outside the PWM module causes updates 111 Reserved
13 RELOAD_SEL	Reload Source Select This read/write bit determines the source of the RELOAD signal for this submodule. When this bit is set, the LDOK bit in submodule 0 should be used since the local LDOK bit will be ignored. 1 The master RELOAD signal (from submodule 0) is used to reload registers. This setting should not be used in submodule 0 as it will force the RELOAD signal to logic 0. 0 The local RELOAD signal is used to reload registers.
14:15 CLK_SEL	Clock Source Select These read/write bits determine the source of the clock signal for this submodule. 00 The IPBus clock is used as the clock for the local prescaler and counter. 01 EXT_CLK is used as the clock for the local prescaler and counter. 10 Submodule 0's clock (AUX_CLK) is used as the source clock for the local prescaler and counter. This setting should not be used in submodule 0 as it will force the clock to logic 0. 11 Reserved

36.3.5 Control 1 register n (CTRL1n)

Offset: 0x0006 + *n**0x50 (*n*=0 to 3) Access: R/W

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	LDFQ				HALF	FULL	DT		0	PRSC			PHSSHFT	LDMOD	0	DBLEN
W																
Reset	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0

Figure 424. Control 1 register *n* (CTRL1*n*)

Table 523. CTRL1*n* field descriptions

Field	Description
0:3 LDFQ	Load Frequency These buffered read/write bits select the PWM load frequency according to *Table 524*. Reset clears the LDFQ bits, selecting loading every PWM opportunity. A PWM opportunity is determined by the HALF and FULL bits. **Note:** The LDFQx bits take effect when the current load cycle is complete regardless of the state of the LDOK bit. Reading the LDFQx bits reads the buffered values and not necessarily the values currently in effect.
4 HALF	Half Cycle Reload This read/write bit enables half-cycle reloads. A half cycle is defined by when the submodule counter matches the VAL0 register and does not have to be half way through the PWM cycle. 1 Half-cycle reloads enabled 0 Half-cycle reloads disabled
5 FULL	Full Cycle Reload This read/write bit enables full-cycle reloads. A full cycle is defined by when the submodule counter matches the VAL1 register. Either the HALF or FULL bit must be set in order to move the buffered data into the registers used by the PWM generators. If both the HALF and FULL bits are set, then reloads can occur twice per cycle. 1 Full-cycle reloads enabled 0 Full-cycle reloads disabled
6:7 DT	Deadtime These read only bits reflect the sampled values of the PWMX input at the end of each deadtime. Sampling occurs at the end of deadtime 0 for DT[0] and the end of deadtime 1 for DT[1]. Reset clears these bits.
9:11 PRSC	Prescaler These buffered read/write bits select the divide ratio of the PWM clock frequency selected by CLK_SEL as illustrated in *Table 525*.
12 PHSSHFT	Phase Shift Mode Enable This read/write bit is only functional in submodule 0. It is used to enable the EXT_SWITCH input to select the source of the generated PWM23/PWM45 signals within submodule 0. If this enable is clear or the EXT_SWITCH signal is 0, then the PWM23/PWM45 signals generated within submodule 0 are used by the force out logic in submodule 0. If this enable is set and the EXT_SWITCH signal is also set, then the PWM23/PWM45 signals from submodule 1 are used by the force out logic in submodule 0. 1 Phase shifted mode is enabled. The EXT_SWITCH input is used to select generated PWM source for force out logic in submodule 0 0 Phase shifted mode is disabled. The EXT_SWITCH input is ignored. Normal operation

Table 523. CTRL1*n* field descriptions (continued)

Field	Description
13 LDMOD	Load Mode Select This read/write bit selects the timing of loading the buffered registers for this submodule. 1 Buffered registers of this submodule are loaded and take effect immediately upon LDOK being set 0 Buffered registers of this submodule are loaded and take effect at the next PWM reload if LDOK is set
15 DBLEN	Double Switching Enable This read/write bit enables the double switching PWM behavior. 1 Double switching enabled 0 Double switching disabled

Table 524. PWM Reload Frequency

LDFQ	PWM reload frequency
0000	Every PWM opportunity
0001	Every 2 PWM opportunities
0010	Every 3 PWM opportunities
0011	Every 4 PWM opportunities
0100	Every 5 PWM opportunities
0101	Every 6 PWM opportunities
0110	Every 7 PWM opportunities
0111	Every 8 PWM opportunities
1000	Every 9 PWM opportunities
1001	Every 10 PWM opportunities
1010	Every 11 PWM opportunities
1011	Every 12 PWM opportunities
1100	Every 13 PWM opportunities
1101	Every 14 PWM opportunities
1110	Every 15 PWM opportunities
1111	Every 16 PWM opportunities

Table 525. PWM Prescaler

PRSC	PWM clock frequency
000	f_{clk}
001	$f_{clk}/2$
010	$f_{clk}/4$
011	$f_{clk}/8$
100	$f_{clk}/16$
101	$f_{clk}/32$
110	$f_{clk}/64$
111	$f_{clk}/128$

Note: Reading the PRSCx bits reads the buffered values and not necessarily the values currently in effect. The PRSCx bits take effect at the beginning of the next PWM cycle and only when the load okay bit, LDOK, is set or LDMOD is set. This field cannot be written when LDOK is set.

36.3.6 Value 0 register n (VAL0n)

Offset: 0x0008 + *n**0x50 (*n*=0 to 3) Access: R/W

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	VAL0															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 425. Value 0 register *n* (VAL0*n*)

Table 526. VAL0*n* field descriptions

Field	Description
0:15 VAL0	Value Register 0

The 16-bit signed value in this buffered, read/write register defines the mid-cycle reload point for the PWM in PWM clock periods. This value also defines when the PWMX signal is set and the local sync signal is reset.

Note: The VAL0 register is buffered. The value written does not take effect until the LDOK bit is set and the next PWM load cycle begins or LDMOD is set. VAL0 cannot be written when LDOK is set. Reading VAL0 reads the value in a buffer. It is not necessarily the value the PWM generator is currently using.

36.3.7 Value 1 register n (VAL1n)

Offset: 0x000A + *n**0x50 (*n*=0 to 3) Access: R/W

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	VAL1															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 426. Value 1 register *n* (VAL1*n*)

Table 527. VAL1*n* field descriptions

Field	Description
0:15 VAL1	Value Register 1

The 16-bit signed value written to this buffered, read/write register defines the modulo count value (maximum count) for the submodule counter. Upon reaching this count value, the counter will reload itself with the contents of the INIT register and assert the local sync signal while resetting PWMX

Note: The VAL1 register is buffered. The value written does not take effect until the LDOK bit is set and the next PWM load cycle begins or LDMOD is set. VAL1 cannot be written when LDOK is set. Reading VAL1 reads the value in a buffer and not necessarily the value the PWM generator is currently using.

If VAL1 register defines the timer period (Local Sync is selected as the counter initialization signal), a 100% duty cycle cannot be achieved on the PWMX output. The PWMX output will be low for a minimum of one count every cycle after the count reaches VAL1.

When the Master Sync signal (only originated by the Local Sync from submodule 0) is used to control the timer period, the VAL1 register can be free for other functions such as PWM generation without duty cycle limitation

36.3.8 Value 2 register n (VAL2n)

Offset: 0x000C + *n**0x50 (*n*=0 to 3) Access: R/W

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	VAL2															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 427. Value 2 register *n* (VAL2*n*)

Table 528. VAL2*n* field descriptions

Field	Description
0:15 VAL2	Value Register 2

The 16-bit signed value in this buffered, read/write register defines the count value to set PWM23 high (*Figure 420*).

Note: The VAL2 register is buffered. The value written does not take effect until the LDOK bit is set and the next PWM load cycle begins or LDMOD is set. VAL2 cannot be written when LDOK is set. Reading VAL2 reads the value in a buffer and not necessarily the value the PWM generator is currently using.

36.3.9 Value 3 register n (VAL3n)

Offset: 0x000E + *n**0x50 (*n*=0 to 3) Access: R/W

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	VAL3															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 428. Value 3 register *n* (VAL3*n*)

Table 529. VAL3*n* field descriptions

Field	Description
0:15 VAL3	Value Register 3

The 16-bit signed value in this buffered, read/write register defines the count value to set PWM23 low (*Figure 420*).

Note: The VAL3 register is buffered. The value written does not take effect until the LDOK bit is set and the next PWM load cycle begins or LDMOD is set. VAL3 cannot be written when LDOK is set. Reading VAL3 reads the value in a buffer and not necessarily the value the PWM generator is currently using.

36.3.10 Value 4 register n (VAL4n)

Offset: 0x0010 + *n**0x50 (*n*=0 to 3) Access: R/W

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	VAL4															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 429. Value 4 register *n* (VAL4*n*)

Table 530. VAL4*n* field descriptions

Field	Description
0:15 VAL4	Value Register 4

The 16-bit signed value in this buffered, read/write register defines the count value to set PWM45 high (*Figure 420*).

Note: The VAL4 register is buffered. The value written does not take effect until the LDOK bit is set and the next PWM load cycle begins or LDMOD is set. VAL4 cannot be written when LDOK is set. Reading VAL4 reads the value in a buffer and not necessarily the value the PWM generator is currently using.

36.3.11 Value 5 register n (VAL5n)

Offset: 0x0012 + *n**0x50 (*n*=0 to 3) Access: R/W

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	VAL5															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 430. Value 5 register *n* (VAL5*n*)

Table 531. VAL5*n* field descriptions

Field	Description
0:15 VAL5	Value Register 5

The 16-bit signed value in this buffered, read/write register defines the count value to set PWM45 low (*Figure 420*).

Note: The VAL5 register is buffered. The value written does not take effect until the LDOK bit is set and the next PWM load cycle begins or LDMOD is set. VAL5 cannot be written when LDOK is set. Reading VAL5 reads the value in a buffer and not necessarily the value the PWM generator is currently using.

36.3.12 Output Control register n (OCTRLn)

Offset: 0x0018 + *n**0x50 (*n*=0 to 3) Access: R/W

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	PWMA_IN	PWMB_IN	PWMX_IN	0	0	POLA	POLB	POLX	0	0	PWMAFS		PWMBFS		PWMXFS	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 431. Output Control register *n* (OCTRL*n*)

Table 532. OCTRL*n* field descriptions

Field	Description
0 PWMA_IN	PWMA Input This read only bit shows the logic value currently being driven into the PWMA input.
1 PWMB_IN	PWMB Input This read only bit shows the logic value currently being driven into the PWMB input.
2 PWMX_IN	PWMX Input This read only bit shows the logic value currently being driven into the PWMX input.
5 POLA	PWMA Output Polarity This bit inverts the PWMA output polarity. 1 PWMA output inverted. A low level on the PWMA pin represents the "on" or "active" state. 0 PWMA output not inverted. A high level on the PWMA pin represents the "on" or "active" state.
6 POLB	PWMB Output Polarity This bit inverts the PWMB output polarity. 1 PWMB output inverted. A low level on the PWMB pin represents the "on" or "active" state. 0 PWMB output not inverted. A high level on the PWMB pin represents the "on" or "active" state.
7 POLX	PWMX Output Polarity This bit inverts the PWMX output polarity. 1 PWMX output inverted. A low level on the PWMX pin represents the "on" or "active" state. 0 PWMX output not inverted. A high level on the PWMX pin represents the "on" or "active" state.
10:11 PWMAFS	PWMA Fault State These bits determine the fault state for the PWMA output during fault conditions and STOP mode. It may also define the output state during WAIT/HALT mode and DEBUG mode depending on the setting of WAITEN and DBGEN. 00 Output is forced to logic 0 state prior to consideration of output polarity control. 01 Output is forced to logic 1 state prior to consideration of output polarity control. 10 Output is tristated. 11 Output is tristated.
12:13 PWMBFS	PWMB Fault State These bits determine the fault state for the PWMB output during fault conditions and STOP mode. It may also define the output state during WAIT/HALT mode and DEBUG mode depending on the setting of WAITEN and DBGEN. 00 Output is forced to logic 0 state prior to consideration of output polarity control. 01 Output is forced to logic 1 state prior to consideration of output polarity control. 10 Output is tristated. 11 Output is tristated.
14:15 PWMXFS	PWMX Fault State These bits determine the fault state for the PWMX output during fault conditions and STOP mode. It may also define the output state during WAIT/HALT mode and DEBUG mode depending on the setting of WAITEN and DBGEN. 00 Output is forced to logic 0 state prior to consideration of output polarity control. 01 Output is forced to logic 1 state prior to consideration of output polarity control. 10 Output is tristated. 11 Output is tristated.

36.3.13 Status register n (STSn)

Offset: 0x001A + *n**0x50 (*n*=0 to 3) Access: R/W

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	RUF	REF	RF	CFA1	CFA0	CFB1	CFB0	CFX1	CFX0	CMPF					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 432. Status register *n* (STS*n*)

Table 533. STS*n* field descriptions

Field	Description
1 RUF	Registers Updated Flag This read only flag is set when one of the INIT, VALx, or PRSC registers has been written resulting in non-coherent data in the set of double buffered registers. Clear RUF by a proper reload sequence consisting of a reload signal while LDOK = 1. Reset clears RUF. 1 At least one of the double buffered registers has been updated since the last reload. 0 No register update has occurred since last reload.
2 REF	Reload Error Flag This read/write flag is set when a reload cycle occurs while LDOK is 0 and the double buffered registers are in a non-coherent state (RUF = 1). Clear REF by writing a logic one to the REF bit. Reset clears REF. 1 Reload signal occurred with non-coherent data and LDOK = 0 0 No reload error occurred
3 RF	Reload Flag This read/write flag is set at the beginning of every reload cycle regardless of the state of the LDOK bit. Clear RF by writing a logic one to the RF bit when VALDE is clear (non-DMA mode). RF can also be cleared by the DMA done signal when VALDE is set (DMA mode). Reset clears RF. 1 New reload cycle since last RF clearing 0 No new reload cycle since last RF clearing
4 CFA1	Capture Flag A1 This bit is set when the word count of the Capture A1 FIFO (CA1CNT) exceeds the value of the CFAWM field. This bit is cleared by writing a one to this bit position if CA1DE is clear (non-DMA mode) or by the DMA done signal if CA1DE is set (DMA mode). Reset clears this bit.
5 CFA0	Capture Flag A0 This bit is set when the word count of the Capture A0 FIFO (CA0CNT) exceeds the value of the CFAWM field. This bit is cleared by writing a one to this bit position if CA0DE is clear (non-DMA mode) or by the DMA done signal if CA0DE is set (DMA mode). Reset clears this bit.
6 CFB1	Capture Flag B1 This bit is set when the word count of the Capture B1 FIFO (CB1CNT) exceeds the value of the CFBWM field. This bit is cleared by writing a one to this bit position if CB1DE is clear (non-DMA mode) or by the DMA done signal if CB1DE is set (DMA mode). Reset clears this bit.

Table 533. STS*n* field descriptions (continued)

Field	Description
7 CFB0	Capture Flag B0 This bit is set when the word count of the Capture B0 FIFO (CB0CNT) exceeds the value of the CFBWM field. This bit is cleared by writing a one to this bit position if CB0DE is clear (non-DMA mode) or by the DMA done signal if CB0DE is set (DMA mode). Reset clears this bit.
8 CFX1	Capture Flag X1 This bit is set when the word count of the Capture X1 FIFO (CX1CNT) exceeds the value of the CFXWM field. This bit is cleared by writing a one to this bit position if CX1DE is clear (non-DMA mode) or by the DMA done signal if CX1DE is set (DMA mode). Reset clears this bit.
9 CFX0	Capture Flag X0 This bit is set when the word count of the Capture X0 FIFO (CX0CNT) exceeds the value of the CFXWM field. This bit is cleared by writing a one to this bit position if CX0DE is clear (non-DMA mode) or by the DMA done signal if CX0DE is set (DMA mode). Reset clears this bit.
10:15 CMPF	Compare Flags These bits are set when the submodule counter value matches the value of one of the VALx registers. Clear these bits by writing a 1 to a bit position. 1 A compare event has occurred for a particular VALx value. 0 No compare event has occurred for a particular VALx value.

36.3.14 Interrupt Enable register n (INTENn)

Offset: 0x001C + *n**0x50 (*n*=0 to 3) Access: R/W

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	REIE	RIE	CA1IE	CA0IE	CB1IE	CB0IE	CX1IE	CX0IE	CMPIE					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 433. Interrupt Enable register *n* (INTEN*n*)

Table 534. INTEN*n* field descriptions

Field	Description
2 REIE	Reload Error Interrupt Enable This read/write bit enables the reload error flag (REF) to generate CPU interrupt requests. Reset clears RIE. 1 REF CPU interrupt requests enabled 0 REF CPU interrupt requests disabled
3 RIE	Reload Interrupt Enable This read/write bit enables the reload flag (RF) to generate CPU interrupt requests. Reset clears RIE. 1 RF CPU interrupt requests enabled 0 RF CPU interrupt requests disabled

Table 534. INTEN*n* field descriptions (continued)

Field	Description
4 CA1IE	Capture A 1 Interrupt Enable This bit allows the CFA1 flag to create an interrupt request to the CPU. Do not set both this bit and the CA1DE bit. 1 Interrupt request enabled for CFA1 0 Interrupt request disabled for CFA1
5 CA0IE	Capture A 0 Interrupt Enable This bit allows the CFA0 flag to create an interrupt request to the CPU. Do not set both this bit and the CA0DE bit. 1 Interrupt request enabled for CFA0 0 Interrupt request disabled for CFA0
6 CB1IE	Capture B 1 Interrupt Enable This bit allows the CFB1 flag to create an interrupt request to the CPU. Do not set both this bit and the CB1DE bit. 1 Interrupt request enabled for CFB1 0 Interrupt request disabled for CFB1
7 CB0IE	Capture B 0 Interrupt Enable This bit allows the CFB0 flag to create an interrupt request to the CPU. Do not set both this bit and the CB0DE bit. 1 Interrupt request enabled for CFB0 0 Interrupt request disabled for CFB0
8 CX1IE	Capture X 1 Interrupt Enable This bit allows the CFX1 flag to create an interrupt request to the CPU. Do not set both this bit and the CX1DE bit. 1 Interrupt request enabled for CFX1 0 Interrupt request disabled for CFX1
9 CX0IE	Capture X 0 Interrupt Enable This bit allows the CFX0 flag to create an interrupt request to the CPU. Do not set both this bit and the CX0DE bit. 1 Interrupt request enabled for CFX0 0 Interrupt request disabled for CFX0
10:15 CMPIE	Compare Interrupt Enables These bits enable the CMPF flags to cause a compare interrupt request to the CPU. 1 The corresponding CMPF bit will cause an interrupt request. 0 The corresponding CMPF bit will not cause an interrupt request.

36.3.15 DMA Enable register n (DMAENn)

Offset: 0x001E + *n**0x50 (*n*=0 to 3) Access: R/W

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	VALDE	FAND	CAPTDE		CA1DE	CA0DE	CB1DE	CB0DE	CX1DE	CX0DE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 434. DMA Enable register *n* (DMAEN*n*)

Table 535. DMAEN*n* field descriptions

Field	Description
6 VALDE	Value Registers DMA Enable This read/write bit enables DMA write requests for the VALx registers when RF is set. Reset clears VALDE. 1 DMA write requests for the VALx registers enabled 0 DMA write requests disabled
7 FAND	FIFO Watermark AND Control This read/write bit works in conjunction with the CAPTDE field when it is set to watermark mode (CAPTDE = 01). While the CAxDE, CBxDE, and CXxDE bits determine which FIFO watermarks the DMA read request is sensitive to, this bit determines if the selected watermarks are AND'ed together or OR'ed together in order to create the request. 1 Selected FIFO watermarks are AND'ed together. 0 Selected FIFO watermarks are OR'ed together.
8:9 CAPTDE	Capture DMA Enable Source Select These read/write bits select the source of enabling the DMA read requests for the capture FIFOs. Reset clears these bits. 00 Read DMA requests disabled 01 Exceeding a FIFO watermark sets the DMA read request. This requires at least 1 of the CA1DE, CA0DE, CB1DE, CB0DE, CX1DE, or CX0DE bits to also be set in order to determine which watermark(s) the DMA request is sensitive to. 10 A local sync (VAL1 matches counter) sets the read DMA request. 11 A local reload (RF being set) sets the read DMA request.
10 CA1DE	Capture A1 FIFO DMA Enable This read/write bit enables DMA read requests for the Capture A1 FIFO data when CFA1 is set. Reset clears CA1DE. Do not set both this bit and the CA1IE bit.
11 CA0DE	Capture A0 FIFO DMA Enable This read/write bit enables DMA read requests for the Capture A0 FIFO data when CFA0 is set. Reset clears CA0DE. Do not set both this bit and the CA0IE bit.
12 CB1DE	Capture B1 FIFO DMA Enable This read/write bit enables DMA read requests for the Capture B1 FIFO data when CFB1 is set. Reset clears CB1DE. Do not set both this bit and the CB1IE bit.
13 CB0DE	Capture B0 FIFO DMA Enable This read/write bit enables DMA read requests for the Capture B0 FIFO data when CFB0 is set. Reset clears CB0DE. Do not set both this bit and the CB0IE bit.
14 CX1DE	Capture X1 FIFO DMA Enable This read/write bit enables DMA read requests for the Capture X1 FIFO data when CFX1 is set. Reset clears CX1DE. Do not set both this bit and the CX1IE bit.
15 CX0DE	Capture X0 FIFO DMA Enable This read/write bit enables DMA read requests for the Capture X0 FIFO data when CFX0 is set. Reset clears CX0DE. Do not set both this bit and the CX0IE bit.

36.3.16 Output Trigger Control register n (TCTRLn)

Offset: 0x0020 + *n**0x50 (*n*=0 to 3) Access: R/W

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	OUT_TRIG_EN					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 435. Output Trigger Control register *n* (TCTRL*n*)

Table 536. TCTRL*n* field descriptions

Field	Description
10:15 OUT_TRIG_EN	Output Trigger Enables These bits enable the generation of OUT_TRIG0 and OUT_TRIG1 outputs based on the counter value matching the value in one or more of the VAL0-5 registers. VAL0, VAL2, and VAL4 are used to generate OUT_TRIG0 and VAL1, VAL3, and VAL5 are used to generate OUT_TRIG1. The OUT_TRIGx signals are only asserted as long as the counter value matches the VALx value, therefore up to six triggers can be generated (three each on OUT_TRIG0 and OUT_TRIG1) per PWM cycle per submodule. 1 OUT_TRIGx will set when the counter value matches the VALx value. 0 OUT_TRIGx will not set when the counter value matches the VALx value.

36.3.17 Fault Disable Mapping register n (DISMAPn)

This register determines which PWM pins are disabled by the fault protection inputs, illustrated in *Figure 482: Deadtime Insertion and Fine Control Logic* in *Section 36.4.3.11: Fault Protection*. Reset sets all of the bits in the fault disable mapping register.

Offset: 0x0022 + *n**0x50 (*n*=0 to 3) Access: R/W

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	1	1	1	1	DISX				DISB				DISA			
W																
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Figure 436. Fault Disable Mapping register *n* (DISMAP*n*)

Table 537. DISMAP*n* field descriptions

Field	Description
4:7 DISX	PWMX Fault Disable Mask Each of the four bit of this read/write field is one-to-one associated with the four FAULTx inputs. The PWMX output will be turned off if there is a logic 1 on a FAULTx input and a 1 in the corresponding bit of the DISX field. A reset sets all DISX bits.

Table 537. DISMAP*n* field descriptions (continued)

Field	Description
8:11 DISB	PWMB Fault Disable Mask Each of the four bit of this read/write field is one-to-one associated with the four FAULTx inputs. The PWMB output will be turned off if there is a logic 1 on a FAULTx input and a 1 in the corresponding bit of the DISB field. A reset sets all DISB bits.
12:15 DISA	PWMA Fault Disable Mask Each of the four bit of this read/write field is one-to-one associated with the four FAULTx inputs. The PWMA output will be turned off if there is a logic 1 on a FAULTx input and a 1 in the corresponding bit of the DISA field. A reset sets all DISA bits.

36.3.18 Deadtime Count 0 register n (DTCNT0n)

Deadtime operation is only applicable to complementary channel operation. The 11-bit values written to these registers are in terms of IPBus clock cycles regardless of the setting of PRSC and/or CLK_SEL. Reset sets the deadtime count registers to a default value of 0x07FF, selecting a deadtime of 2047 IPBus clock cycles.

Offset: 0x0024 + *n**0x50 (*n*=0 to 3) Access: R/W

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	DTCNT0n										
W																
Reset	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1

Figure 437. Deadtime Count 0 register *n* (DTCNT0*n*)

Table 538. DTCNT0*n* field descriptions

Field	Description
5:15 DTCNT0n	Deadtime Count Register 0 The DTCNT0 field is used to control the deadtime during 0 to 1 transitions of the PWMA output (assuming normal polarity).

36.3.19 Deadtime Count 1 register n (DTCNT1n)

Deadtime operation is only applicable to complementary channel operation. The 11-bit values written to these registers are in terms of IPBus clock cycles regardless of the setting of PRSC and/or CLK_SEL. Reset sets the deadtime count registers to a default value of 0x07FF, selecting a deadtime of 2047 IPBus clock cycles.

Offset: 0x0026 + *n**0x50 (*n*=0 to 3) Access: R/W

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	DTCNT1*n*										
W																
Reset	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1

Figure 438. Deadtime Count 1 register n (DTCNT1*n*)

Table 539. DTCNT1*n* field descriptions

Field	Description
5:15 DTCNT1*n*	Deadtime Count Register 1 The DTCNT1 field is used to control the deadtime during 0 to 1 transitions of the complementary PWMB output.

36.3.20 Capture Control A register n (CAPTCTRLAn)

Offset: 0x0028 + *n**0x50 (*n*=0 to 3) Access: R/W

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	CA1CNT			CA0CNT			CFAWM		EDGCNTA_EN	INP_SELA	EDGA1		EDGA0		ONESHOTA	ARMA
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 439. Capture Control A register *n* (CAPTCTRLA*n*)

Table 540. CAPTCTRLA*n* field descriptions

Field	Description
0:2 CA1CNT	Capture A1 FIFO Word Count This field reflects the number of words in the Capture A1 FIFO.
3:5 CA0CNT	Capture A0 FIFO Word Count This field reflects the number of words in the Capture A0 FIFO.
6:7 CFAWM	Capture A FIFOs Water Mark This field represents the water mark level for capture A FIFOs. The capture flags, CFA1 and CFA0, won't be set until the word count of the corresponding FIFO is greater than this water mark level.
8 EDGCNTA_EN	Edge Counter A Enable This bit enables the edge counter which counts rising and falling edges on the PWMA input signal. 1 Edge counter enabled 0 Edge counter disabled and held in reset.
9 INP_SELA	Input Select A This bit selects between the raw PWMA input signal and the output of the edge counter/compare circuitry as the source for the input capture circuit. 1 Output of edge counter/compare selected as source. 0 Raw PWMA input signal selected as source. **Note:** When INPSELA = 1, the internal edge counter is enabled and the rising and/or falling edges specified by the EDGA0 and EDGA1 fields are ignored. The software must still place a value other than 00 in either or both of the EDGA0 and/or EDGA1 fields in order to enable one or both of the capture registers.

Table 540. CAPTCTRLA*n* field descriptions (continued)

Field	Description
10:11 EDGA1	Edge A 1 These bits control the input capture 1 circuitry by determining which input edges cause a capture event. 00 Disabled 01 Capture falling edges 10 Capture rising edges 11 Capture any edge
12:13 EDGA0	Edge A 0 These bits control the input capture 0 circuitry by determining which input edges cause a capture event. 00 Disabled 01 Capture falling edges 10 Capture rising edges 11 Capture any edge
14 ONESHOTA	One Shot Mode A This bit selects between free running and one shot mode for the input capture circuitry. 1 One shot mode is selected If both capture circuits are enabled, then capture circuit 0 is armed first after the ARMA bit is set. Once a capture occurs, capture circuit 0 is disarmed and capture circuit 1 is armed. After capture circuit 1 performs a capture, it is disarmed and the ARMA bit is cleared. No further captures will be performed until the ARMA bit is set again. If only one of the capture circuits is enabled, then a single capture will occur on the enabled capture circuit and the ARMA bit is then cleared. 0 Free running mode is selected If both capture circuits are enabled, then capture circuit 0 is armed first after the ARMA bit is set. Once a capture occurs, capture circuit 0 is disarmed and capture circuit 1 is armed. After capture circuit 1 performs a capture, it is disarmed and capture circuit 0 is re-armed. The process continues indefinitely. If only one of the capture circuits is enabled, then captures continue indefinitely on the enabled capture circuit.
15 ARMA	Arm A Setting this bit high starts the input capture process. This bit can be cleared at any time to disable input capture operation. This bit is self cleared when in one shot mode and the enabled capture circuit(s) has had a capture event(s). To restart capture operations clear this bit, read CVAL2 and CVAL3 until CA0CNT and CA1CNT are 0, then set ARMA. 1 Input capture operation as specified by the EDGAx bits is enabled. 0 Input capture operation is disabled.

36.3.21 Capture Compare A register n (CAPTCMPAn)

Offset: 0x002A +*n**0x50 (*n*=0 to 3) Access: R/W

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	EDGCNTA								EDGCMPA							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 440. Capture Compare A register *n* (CAPTCMPA*n*)

Table 541. CAPTCMPA*n* field descriptions

Field	Description
0:7 EDGCNTA	Edge Counter A This read only field contains the edge counter value for the PWMA input capture circuitry.
8:15 EDGCMPA	Edge Compare A This read/write field is the compare value associated with the edge counter for the PWMA input capture circuitry.

36.3.22 Capture Control B register n (CAPTCTRLBn)

Offset: 0x002C + *n**0x50 (*n*=0 to 3) Access: R/W

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	CB1CNT			CB0CNT			CFBWM		EDG CNTB_EN	INP_SELB	EDGB1		EDGB0		ONESHOTB	ARMB
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 441. Capture Control B register *n* (CAPTCTRLB*n*)

Table 542. CAPTCMPA*n* field descriptions

Field	Description
0:2 CB1CNT	Capture B1 FIFO Word Count This field reflects the number of words in the Capture B1 FIFO.
3:5 CB0CNT	Capture B0 FIFO Word Count This field reflects the number of words in the Capture B0 FIFO.
6:7 CFBWM	Capture B FIFOs Water Mark This field represents the water mark level for capture B FIFOs. The capture flags, CFB1 and CFB0, won't be set until the word count of the corresponding FIFO is greater than this water mark level.
8 EDGCNTB_EN	Edge Counter B Enable This bit enables the edge counter which counts rising and falling edges on the PWMB input signal. 1 Edge counter enabled 0 Edge counter disabled and held in reset

Table 542. CAPTCMPA*n* field descriptions (continued)

Field	Description
9 INP_SELB	Input Select B This bit selects between the raw PWMB input signal and the output of the edge counter/compare circuitry as the source for the input capture circuit. 1 Output of edge counter/compare selected as source 0 Raw PWMB input signal selected as source **Note:** When INPSELB = 1, the internal edge counter is enabled and the rising and/or falling edges specified by the EDGB0 and EDGB1 fields are ignored. The software must still place a value other than 00 in either or both of the EDGB0 and/or EDGB1 fields in order to enable one or both of the capture registers.
10:11 EDGB1	Edge B 1 These bits control the input capture 1 circuitry by determining which input edges cause a capture event. 00 Disabled 01 Capture falling edges 10 Capture rising edges 11 Capture any edge
12:13 EDGB0	Edge B 0 These bits control the input capture 0 circuitry by determining which input edges cause a capture event. 00 Disabled 01 Capture falling edges 10 Capture rising edges
14 ONESHOTB	One Shot Mode B This bit selects between free running and one shot mode for the input capture circuitry. 1 One shot mode is selected If both capture circuits are enabled, then capture circuit 0 is armed first after the ARMB bit is set. Once a capture occurs, capture circuit 0 is disarmed and capture circuit 1 is armed. After capture circuit 1 performs a capture, it is disarmed and the ARMB bit is cleared. No further captures will be performed until the ARMB bit is set again. If only one of the capture circuits is enabled, then a single capture will occur on the enabled capture circuit and the ARMB bit is then cleared. 0 Free running mode is selected If both capture circuits are enabled, then capture circuit 0 is armed first after the ARMB bit is set. Once a capture occurs, capture circuit 0 is disarmed and capture circuit 1 is armed. After capture circuit 1 performs a capture, it is disarmed and capture circuit 0 is re-armed. The process continues indefinitely. If only one of the capture circuits is enabled, then captures continue indefinitely on the enabled capture circuit.
15 ARMB	Arm B Setting this bit high starts the input capture process. This bit can be cleared at any time to disable input capture operation. This bit is self cleared when in one shot mode and the enabled capture circuit(s) has had a capture event(s). To restart capture operations clear this bit, read CVAL4 and CVAL5 until CB0CNT and CB1CNT are 0, then set ARMB. 1 Input capture operation as specified by the EDGBx bits is enabled. 0 Input capture operation is disabled.

36.3.23 Capture Compare B register n (CAPTCMPBn)

Offset: 0x002E + *n**0x50 (*n*=0 to 3) Access: R/W

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	EDGCNTB								EDGCMPB							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 442. Capture Compare B register *n* (CAPTCMPB*n*)

Table 543. CAPTCMPB*n* field descriptions

Field	Description
0:7 EDGCNTB	Edge Counter B This read only field contains the edge counter value for the PWMB input capture circuitry.
8:15 EDGCMPB	Edge Compare B This read/write field is the compare value associated with the edge counter for the PWMB input capture circuitry.

36.3.24 Capture Control X register n (CAPTCTRLXn)

Offset: 0x0030 + *n**0x50 (*n*=0 to 3) Access: R/W

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	CX1CNT			CX0CNT			CFXWM		EDGCNTX_EN	INP_SELX	EDGX1		EDGX0		ONESHOTX	ARMX
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 443. Capture Control X register *n* (CAPTCTRLX*n*)

Table 544. CAPTCTRLX*n* field descriptions

Field	Description
0:2 CX1CNT	Capture X1 FIFO Word Count This field reflects the number of words in the Capture X1 FIFO.
3:5 CX0CNT	Capture X0 FIFO Word Count This field reflects the number of words in the Capture X0 FIFO.
6:7 CFXWM	Capture X FIFOs Water Mark This field represents the water mark level for capture X FIFOs. The capture flags, CFX1 and CFX0, won't be set until the word count of the corresponding FIFO is greater than this water mark level.

Table 544. CAPTCTRLX*n* field descriptions (continued)

Field	Description
8 EDGCNTX_EN	Edge Counter X Enable This bit enables the edge counter which counts rising and falling edges on the PWMX input signal. 1 Edge counter enabled 0 Edge counter disabled and held in reset
9 INP_SELX	Input Select X This bit selects between the raw PWMX input signal and the output of the edge counter/compare circuitry as the source for the input capture circuit. 1 Output of edge counter/compare selected as source 0 Raw PWMX input signal selected as source **Note:** When INPSELX = 1, the internal edge counter is enabled and the rising and/or falling edges specified by the EDGX0 and EDGX1 fields are ignored. **Note:** The software must still place a value other than 00 in either or both of the EDGX0 and/or EDGX1 fields in order to enable one or both of the capture registers.
10:11 EDGX1	Edge X 1 These bits control the input capture 1 circuitry by determining which input edges cause a capture event. 00 Disabled 01 Capture falling edges 10 Capture rising edges 11 Capture any edge
12:13 EDGX0	Edge X 0 These bits control the input capture 0 circuitry by determining which input edges cause a capture event. 00 Disabled 01 Capture falling edges 10 Capture rising edges 11 Capture any edge

Table 544. CAPTCTRLX*n* field descriptions (continued)

Field	Description
14 ONESHOTX	One Shot Mode Aux This bit selects between free running and one shot mode for the input capture circuitry. 1 One shot mode is selected If both capture circuits are enabled, then capture circuit 0 is armed first after the ARMX bit is set. Once a capture occurs, capture circuit 0 is disarmed and capture circuit 1 is armed. After capture circuit 1 performs a capture, it is disarmed and the ARMX bit is cleared. No further captures will be performed until the ARMX bit is set again. If only one of the capture circuits is enabled, then a single capture will occur on the enabled capture circuit and the ARMX bit is then cleared. 0 Free running mode is selected If both capture circuits are enabled, then capture circuit 0 is armed first after the ARMX bit is set. Once a capture occurs, capture circuit 0 is disarmed and capture circuit 1 is armed. After capture circuit 1 performs a capture, it is disarmed and capture circuit 0 is re-armed. The process continues indefinitely. If only one of the capture circuits is enabled, then captures continue indefinitely on the enabled capture circuit.
15 ARMX	Arm X Setting this bit high starts the input capture process. This bit can be cleared at any time to disable input capture operation. This bit is self cleared when in one shot mode and the enabled capture circuit(s) has had a capture event(s). To restart capture operations clear this bit, read CVAL0 and CVAL1 until CX0CNT and CX1CNT are 0, then set ARMX. 1 Input capture operation as specified by the EDGX bits is enabled. 0 Input capture operation is disabled.

36.3.25 Capture Compare X register n (CAPTCMPXn)

Offset: 0x0032 + *n**0x50 (*n*=0 to 3) Access: R/W

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	EDGCNTX								EDGCMPX							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 444. Capture Compare X register *n* (CAPTCMPX*n*)

Table 545. CAPTCMPX*n* field descriptions

Field	Description
0:7 EDGCNTX	Edge Counter X This read only field contains the edge counter value for the PWMX input capture circuitry.
8:15 EDGCMPX	This read/write field is the compare value associated with the edge counter for the PWMX input capture circuitry.

36.3.26 Capture Value 0 register n (CVAL0n)

Offset: 0x0034 + *n**0x50 (*n*=0 to 3) Access: R/W

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	CAPTVAL0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 445. Capture Value 0 register *n* (CVAL0*n*)

Table 546. CVAL0*n* field descriptions

Field	Description
0:15 CAPTVAL0	Capture Value 0

This read only register stores the value captured from the submodule counter. Exactly when this capture occurs is defined by the EDGX0 bits. This is actually a 4 deep FIFO and not a single register.

36.3.27 Capture Value 0 Cycle register n (CVAL0CYCn)

Offset: 0x0036 + *n**0x50 (*n*=0 to 3) Access: Read-only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	CVAL0CYC		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 446. Capture Value 0 Cycle register *n* (CVAL0CYC*n*)

Table 547. CVAL0CYC*n* field descriptions

Field	Description
2:013:15 CVAL0CYC	Capture Value 0 Cycle

This read only register stores the cycle number corresponding to the value captured in CVAL0. The PWM cycle is reset to 0 and is incremented each time the counter is loaded with the INIT value. Eventually, this counter rolls over to 0. This is actually a 4 deep FIFO and not a single register.

36.3.28 Capture Value 1 register n (CVAL1n)

Offset: 0x0038 + *n**0x50 (*n*=0 to 3) Access: Read-only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	CAPTVAL1															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 447. Capture Value 1 register *n* (CVAL1*n*)

Table 548. CVAL1*n* field descriptions

Field	Description
0:15 CAPTVAL1	Capture Value 1

This read only register stores the value captured from the submodule counter. Exactly when this capture occurs is defined by the EDGX1 bits. This is actually a 4 deep FIFO and not a single register.

36.3.29 Capture Value 1 Cycle register n (CVAL1CYCn)

Offset: 0x003A + *n**0x50 (*n*=0 to 3) Access: Read-only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	CVAL1CYC		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 448. Capture Value 1 Cycle register *n* (CVAL1CYC*n*)

Table 549. CVAL1CYC*n* field descriptions

Field	Description
13:15 CVAL1CYC	Capture Value 1 Cycle

This read only register stores the cycle number corresponding to the value captured in CVAL1. The PWM cycle is reset to 0 and is incremented each time the counter is loaded with the INIT value. Eventually, this counter rolls over to 0. This is actually a 4 deep FIFO and not a single register.

36.3.30 Capture Value 2 register n (CVAL2n)

Offset: 0x003C + *n**0x50 (*n*=0 to 3) Access: Read-only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	CVAL2															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 449. Capture Value 2 register *n* (CVAL2*n*)

Table 550. CVAL2*n* field descriptions

Field	Description
0:15 CVAL2	Capture Value 2

This read only register stores the value captured from the submodule counter. Exactly when this capture occurs is defined by the EDGA0 bits. This is actually a 4 deep FIFO and not a single register.

36.3.31 Capture Value 2 Cycle register n (CVAL2CYCn)

Offset: 0x003E + *n**0x50 (*n*=0 to 3) Access: Read-only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	CVAL2CYC		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 450. Capture Value 2 Cycle register *n* (CVAL2CYC*n*)

Table 551. CVAL2CYC*n* field descriptions

Field	Description
13:15 CVAL2CYC	Capture Value 2 Cycle

This read only register stores the cycle number corresponding to the value captured in CVAL2. The PWM cycle is reset to 0 and is incremented each time the counter is loaded with the INIT value. Eventaully, this counter rolls over to 0. This is actually a 4 deep FIFO and not a single register.

36.3.32 Capture Value 3 register n (CVAL3n)

Offset: 0x0040 + *n**0x50 (*n*=0 to 3) Access: Read-only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	CVAL3															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 451. Capture Value 3 register *n* (CVAL3*n*)

Table 552. CVAL3*n* field descriptions

Field	Description
0:15 CVAL3	Capture Value 3

This read only register stores the value captured from the submodule counter. Exactly when this capture occurs is defined by the EDGA1 bits. This is actually a 4 deep FIFO and not a single register.

36.3.33 Capture Value 3 Cycle register n (CVAL3CYCn)

Offset: 0x0042 + *n**0x50 (*n*=0 to 3) Access: Read-only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	CVAL3CYC		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 452. Capture Value 3 Cycle register *n* (CVAL3CYC*n*)

Table 553. CVAL3CYC*n* field descriptions

Field	Description
13:15 CVAL3CYC	Capture Value 3 Cycle

This read only register stores the cycle number corresponding to the value captured in CVAL3. The PWM cycle is reset to 0 and is incremented each time the counter is loaded with the INIT value. Eventually, this counter rolls over to 0. This is actually a 4 deep FIFO and not a single register.

36.3.34 Capture Value 4 register n (CVAL4n)

Offset: 0x0044 + *n**0x50 (*n*=0 to 3) Access: Read-only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	CVAL4															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 453. Capture Value 4 register *n* (CVAL4*n*)

Table 554. CVAL4*n* field descriptions

Field	Description
0:15 CVAL4	Capture Value 4

This read only register stores the value captured from the submodule counter. Exactly when this capture occurs is defined by the EDGB0 bits. This is actually a 4 deep FIFO and not a single register.

36.3.35 Capture Value 4 Cycle register n (CVAL4CYCn)

Offset: 0x0046 + *n**0x50 (*n*=0 to 3) Access: Read-only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	CVAL4CYC			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 454. Capture Value 4 Cycle register *n* (CVAL4CYC*n*)

Table 555. CVAL4CYC*n* field descriptions

Field	Description
12:15 CVAL4CYC	Capture Value 4 Cycle

This read only register stores the cycle number corresponding to the value captured in CVAL4. The PWM cycle is reset to 0 and is incremented each time the counter is loaded with the INIT value. Eventually, this counter rolls over to 0. This is actually a 4 deep FIFO and not a single register.

36.3.36 Capture Value 5 register n (CVAL5n)

Offset: 0x0048 + *n**0x50 (*n*=0 to 3) Access: Read-only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	CVAL5															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 455. Capture Value 5 register *n* (CVAL5*n*)

Table 556. CVAL5*n* field descriptions

Field	Description
0:15 CVAL5	Capture Value 5

This read only register stores the value captured from the submodule counter. Exactly when this capture occurs is defined by the EDGB1 bits. This is actually a 4 deep FIFO and not a single register.

36.3.37 Capture Value 5 Cycle register n (CVAL5CYCn)

Offset: 0x004A + *n**0x50 (*n*=0 to 3) Access: Read-only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	CVAL5CYC			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 456. Capture Value 5 Cycle register *n* (CVAL5CYC*n*)

Table 557. CVAL5CYC*n* field descriptions

Field	Description
12:15 CVAL5CYC	Capture Value 5 Cycle

This read only register stores the cycle number corresponding to the value captured in CVAL5. The PWM cycle is reset to 0 and is incremented each time the counter is loaded with the INIT value. Eventaully, this counter rolls over to 0. This is actually a 4 deep FIFO and not a single register.

36.3.38 Configuration Registers

The base address of the configuration registers is equal to the base address of the mcPWM plus as offset of $140.

36.3.38.1 Output Enable register (OUTEN)

Offset: 0x0140 Access: R/W

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	PWMA_EN				PWMB_EN				PWMX_EN			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 457. Output Enable register (OUTEN)

Table 558. OUTEN field descriptions

Field	Description
4:7 PWMA_EN	PWMA Output Enables These bits enable the PWMA outputs of each submodule. These bits should be set to 0 (output disabled) when a PWMA pin is being used for input capture. 1 PWMA output enabled 0 PWMA output disabled
8:11 PWMB_EN	PWMB Output Enables These bits enable the PWMB outputs of each submodule. These bits should be set to 0 (output disabled) when a PWMB pin is being used for input capture. 1 PWMB output enabled 0 PWMB output disabled
12:15 PWMX_EN	PWMX Output Enables These bits enable the PWMX outputs of each submodule. These bits should be set to 0 (output disabled) when a PWMX pin is being used for input capture or deadtime correction. 1 PWMX output enabled 0 PWMX output disabled

36.3.38.2 Mask register (MASK)

Offset: 0x0142 Access: R/W

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0				MASKA				MASKB				MASKX			
W	UPDATE_MASK															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 458. Mask register (MASK)

Note: The MASKx bits are double buffered and do not take effect until a FORCE_OUT event occurs within the appropriate submodule. Refer to Figure 479 to see how FORCE_OUT is generated. Reading the MASK bits reads the buffered value and not necessarily the value currently in effect. This double buffering can be overridden by setting the UPDATE_MASK bits.

Table 559. Mask Register (MASK) field descriptions

Field	Description
0:3 UPDATE_MASK	Update Mask Bits Immediately These self-clearing bits force the MASK* bits to be immediately updated within each of the submodules without waiting for a FORCE_OUT event. Software may write to any or all of these bits and may set these bits in the same write operation that updates the MASKA, MASKB, and MASKX fields of this register. 1 Immediate operation. MASK* bits within the corresponding submodules are updated on the following clock edge after setting this bit 0 Normal operation. MASK* bits within the submodules aren't updated until a FORCE_OUT event occurs within the submodule
4:7 MASKA	PWMA Masks These bits mask the PWMA outputs of each submodule forcing the output to logic 0 prior to consideration of the output polarity. 1 PWMA output masked 0 PWMA output normal
8:11 MASKB	PWMB Masks These bits mask the PWMB outputs of each submodule forcing the output to logic 0 prior to consideration of the output polarity. 1 PWMB output masked 0 PWMB output normal
12:15 MASKX	PWMX Masks These bits mask the PWMX outputs of each submodule forcing the output to logic 0 prior to consideration of the output polarity. 1 PWMX output masked 0 PWMX output normal

36.3.38.3 Software Controlled Output register (SWCOUT)

Offset: 0x0144 Access: R/W

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	OUT23_3	OUT45_3	OUT23_2	OUT45_2	OUT23_1	OUT45_1	OUT23_0	OUT45_0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 459. Software Controlled Output register (SWCOUT)

Note: These bits are double buffered and do not take effect until a FORCE_OUT event occurs within the appropriate submodule. Refer to Figure 479 to see how FORCE_OUT is generated. Reading these bits reads the buffered value and not necessarily the value currently in effect.

Table 560. SWCOUT field descriptions

Field	Description
8 OUT23_3	Software Controlled Output 23_3 This bit is only used when SEL23 for submodule 3 is set to 0b10. It allows software control of which signal is supplied to the deadtime generator of that submodule. 1 A logic 1 is supplied to the deadtime generator of submodule 3 instead of PWM23 0 A logic 0 is supplied to the deadtime generator of submodule 3 instead of PWM23
9 OUT45_3	Software Controlled Output 45_3 This bit is only used when SEL45 for submodule 3 is set to 0b10. It allows software control of which signal is supplied to the deadtime generator of that submodule. 1 A logic 1 is supplied to the deadtime generator of submodule 3 instead of PWM45 0 A logic 0 is supplied to the deadtime generator of submodule 3 instead of PWM45
10 OUT23_2	Software Controlled Output 23_2 This bit is only used when SEL23 for submodule 2 is set to 0b10. It allows software control of which signal is supplied to the deadtime generator of that submodule. 1 A logic 1 is supplied to the deadtime generator of submodule 2 instead of PWM23 0 A logic 0 is supplied to the deadtime generator of submodule 2 instead of PWM23
11 OUT45_2	Software Controlled Output 45_2 This bit is only used when SEL45 for submodule 2 is set to 0b10. It allows software control of which signal is supplied to the deadtime generator of that submodule. 1 A logic 1 is supplied to the deadtime generator of submodule 2 instead of PWM45 0 A logic 0 is supplied to the deadtime generator of submodule 2 instead of PWM45
12 OUT23_1	Software Controlled Output 23_1 This bit is only used when SEL23 for submodule 1 is set to 0b10. It allows software control of which signal is supplied to the deadtime generator of that submodule. 1 A logic 1 is supplied to the deadtime generator of submodule 1 instead of PWM23 0 A logic 0 is supplied to the deadtime generator of submodule 1 instead of PWM23
13 OUT45_1	Software Controlled Output 45_1 This bit is only used when SEL45 for submodule 1 is set to 0b10. It allows software control of which signal is supplied to the deadtime generator of that submodule. 1 A logic 1 is supplied to the deadtime generator of submodule 1 instead of PWM45 0 A logic 0 is supplied to the deadtime generator of submodule 1 instead of PWM45
14 OUT23_0	Software Controlled Output 23_0 This bit is only used when SEL23 for submodule 0 is set to 0b10. It allows software control of which signal is supplied to the deadtime generator of that submodule. 1 A logic 1 is supplied to the deadtime generator of submodule 0 instead of PWM23 0 A logic 0 is supplied to the deadtime generator of submodule 0 instead of PWM23
15 OUT45_0	Software Controlled Output 45_0 This bit is only used when SEL45 for submodule 0 is set to 0b10. It allows software control of which signal is supplied to the deadtime generator of that submodule. 1 A logic 1 is supplied to the deadtime generator of submodule 0 instead of PWM45 0 A logic 0 is supplied to the deadtime generator of submodule 0 instead of PWM45

36.3.38.4 Deadtime Source Select register (DTSRCSEL)

Offset: 0x0146 Access: R/W

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	SEL23_3		SEL45_3		SEL23_2		SEL45_2		SEL23_1		SEL45_1		SEL23_0		SEL45_0	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 460. Deadtime Source Select register (DTSRCSEL)

Note: The deadtime source select bits are double buffered and do not take effect until a FORCE_OUT event occurs within the appropriate submodule. Refer to Figure 479 to see how FORCE_OUT is generated. Reading these bits reads the buffered value and not necessarily the value currently in effect.

Table 561. DTSRCSEL field descriptions

Field	Description
0:1 SEL23_3	PWM23_3 Control Select This field selects possible over-rides to the generated PWM23 signal in submodule 3 that will be passed to the deadtime logic upon the occurrence of a "Force Out" event in that submodule. 00 Generated PWM23_3 signal is used by the deadtime logic. 01 Inverted generated PWM23_3 signal is used by the deadtime logic. 10 OUT23_3 bit is used by the deadtime logic. 11 EXTA[3] signal is used by the deadtime logic.
2:3 SEL45_3	PWM45_3 Control Select This field selects possible over-rides to the generated PWM45 signal in submodule 3 that will be passed to the deadtime logic upon the occurrence of a "Force Out" event in that submodule. 00 Generated PWM45_3 signal is used by the deadtime logic. 01 Inverted generated PWM45_3 signal is used by the deadtime logic. 10 OUT45_3 bit is used by the deadtime logic. 11 EXTB[3] signal is used by the deadtime logic.
4:5 SEL23_2	PWM23_2 Control Select This field selects possible over-rides to the generated PWM23 signal in submodule 2 that will be passed to the deadtime logic upon the occurrence of a "Force Out" event in that submodule. 00 Generated PWM23_2 signal is used by the deadtime logic. 01 Inverted generated PWM23_2 signal is used by the deadtime logic. 10 OUT23_2 bit is used by the deadtime logic. 11 EXTA[2] signal is used by the deadtime logic.
6:7 SEL45_2	PWM45_2 Control Select This field selects possible over-rides to the generated PWM45 signal in submodule 2 that will be passed to the deadtime logic upon the occurrence of a "Force Out" event in that submodule. 00 Generated PWM45_2 signal is used by the deadtime logic. 01 Inverted generated PWM45_2 signal is used by the deadtime logic. 10 OUT45_2 bit is used by the deadtime logic. 11 EXTB[2] signal is used by the deadtime logic.

Table 561. DTSRCSEL field descriptions (continued)

Field	Description
8:9 SEL23_1	PWM23_1 Control Select This field selects possible over-rides to the generated PWM23 signal in submodule 1 that will be passed to the deadtime logic upon the occurrence of a "Force Out" event in that submodule. 00 Generated PWM23_1 signal is used by the deadtime logic. 01 Inverted generated PWM23_1 signal is used by the deadtime logic. 10 OUT23_1 bit is used by the deadtime logic. 11 EXTA[1] signal is used by the deadtime logic.
10:11 SEL45_1	PWM45_1 Control Select This field selects possible over-rides to the generated PWM45 signal in submodule 1 that will be passed to the deadtime logic upon the occurrence of a "Force Out" event in that submodule. 00 Generated PWM45_1 signal is used by the deadtime logic. 01 Inverted generated PWM45_1 signal is used by the deadtime logic. 10 OUT45_1 bit is used by the deadtime logic. 11 EXTB[1] signal is used by the deadtime logic.
12:13 SEL23_0	PWM23_0 Control Select This field selects possible over-rides to the generated PWM23 signal in submodule 0 that will be passed to the deadtime logic upon the occurrence of a "Force Out" event in that submodule. 00 Generated PWM23_0 signal is used by the deadtime logic. 01 Inverted generated PWM23_0 signal is used by the deadtime logic. 10 OUT23_0 bit is used by the deadtime logic. 11 EXTA[0] signal is used by the deadtime logic.
14:15 SEL45_0	PWM45_0 Control Select This field selects possible over-rides to the generated PWM45 signal in submodule 0 that will be passed to the deadtime logic upon the occurrence of a "Force Out" event in that submodule. 00 Generated PWM45_0 signal is used by the deadtime logic. 01 Inverted generated PWM45_0 signal is used by the deadtime logic. 10 OUT45_0 bit is used by the deadtime logic. 11 EXTB[0] signal is used by the deadtime logic.

36.3.38.5 Master Control register (MCTRL)

Offset: 0x0148 Access: R/W

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	IPOL				RUN				0				LDOK			
W									CLDOK							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 461. Master Control register (MCTRL)

Note: The relationship between the fields of MCTRL and the submodules is as follows:

- *Field[3] refers to submodule 3.*
- *Field[2] refers to submodule 2.*
- *Field[1] refers to submodule 1.*

- *Field[0] refers to submodule 0.*

Table 562. MCTRL field descriptions

Field	Description
0:3 IPOL	Current Polarity This buffered read/write bit is used to select between PWM23 and PWM45 as the source for the generation of the complementary PWM pair output in each submodule. IPOL is ignored in independent mode. 1 PWM45 (*Figure 420*) is used to generate complementary PWM pair 0 PWM23 (*Figure 420*) is used to generate complementary PWM pair **Note:** The IPOL bit does not take effect until a FORCE_OUT event takes place in the appropriate submodule. Reading the IPOL bit reads the buffered value and not necessarily the value currently in effect.
4:7 RUN	Run This read/write bit enables the clocks to the PWM generator in each submodule. When RUN equals zero, the submodule counter is reset. A reset clears RUN. 1 PWM generator enabled 0 PWM generator disabled **Note:** For proper initialization of the LDOK and RUN bits, see *Section 36.4.4.5*.
8:11 CLDOK	Clear Load Okay This write only bit is used to clear the LDOK bit. Write a 1 to this location to clear the corresponding LDOK. If a reload occurs with LDOK set at the same time that CLDOK is written, then the reload will not be performed and LDOK will be cleared. This bit is self clearing and always reads as a 0.
12:15 LDOK	Load Okay This read/set bit loads the PRSC bits of CTRL1 and the INIT and VALx registers into a set of buffers in each submodule. The buffered prescaler divisor, submodule counter modulus value, and PWM pulse width take effect at the next PWM reload if LDMOD is clear or immediately if LDMOD is set. Set LDOK by reading it when it is logic zero and then writing a logic one to it. The VALx, INIT, and PRSC registers cannot be written while LDOK is set. In master relaod mode (CTRL2[RELOAD_SEL]=1) it is only required to set the LDOK bit for submodule 0, but it is recommended to also set the LDOK bits for the corresponding slave submodules in order to prevent unwanted writes to the registers in the slave submodules. LDOK is automatically cleared after the new values are loaded, or can be manually cleared before a reload by writing a logic 1 to CLDOK. This bit cannot be written with a zero. LDOK can be set in DMA mode when the DMA indicates that it has completed the update of all PRSC, INIT, and VALx registers. Reset clears LDOK. 1 Load prescaler, modulus, and PWM values 0 Do not load new values

Note: For proper initialization of the LDOK and RUN bits, see Section 36.4.4.5.

36.3.39 Fault Channel Registers

The base address of the fault logic is equal to the base address of the mcPWM plus an offset of $14C.

36.3.39.1 Fault Control register (FCTRL)

Offset: 0x014C Access: R/W

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	FLVL				FAUTO				FSAFE				FIE			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 462. Fault Control register (FCTRL)

Table 563. MCTRL field descriptions

Field	Description
0:3 FLVL	Fault Level These read/write bits select the active logic level of the individual fault inputs. A reset clears FLVL. 1 A logic 1 on the fault input indicates a fault condition. 0 A logic 0 on the fault input indicates a fault condition.
4:7 FAUTO	Automatic Fault Clearing These read/write bits select automatic or manual clearing of faults. A reset clears FAUTO. 1 Automatic fault clearing. PWM outputs disabled by this fault are enabled when the FFPINx bit is clear at the start of a half cycle or full cycle depending on the state of the FFULL bits without regard to the state of FFLAGx bit. 0 Manual fault clearing. PWM outputs disabled by this fault are not enabled until the FFLAGx bit is clear at the start of a half cycle or full cycle depending the state of the FFULL bits. This is further controlled by the FSAFE bits.
8:11 FSAFE	Fault Safety Mode These read/write bits select the safety mode during manual fault clearing. A reset clears FSAFE. 1 Safe mode. PWM outputs disabled by this fault are not enabled until the FFLAGx bit is clear and the FFPINx bit is clear at the start of a half cycle or full cycle depending on the state of the FFULL bits. 0 Normal mode. PWM outputs disabled by this fault are not enabled until the FFLAGx bit is clear at the start of a half cycle or full cycle depending on the state of the FFULL bits without regard to the state of the FFPINx bit. The PWM outputs disabled by this fault input will not be re-enabled until the actual FAULTx input signal de-asserts since the fault input will combinationally disable the PWM outputs (as programmed in DISMAP).
12:15 FIE	Fault Interrupt Enables This read/write bit enables CPU interrupt requests generated by the FAULTx pins. A reset clears FIE. 1 FAULTx CPU interrupt requests enabled 0 FAULTx CPU interrupt requests disabled

Note: The FFPINx bit may indicate a fault condition still exists even though the actual fault signal at the FAULTx pin is clear due to the fault filter latency.

The fault protection circuit is independent of the FIEx bit and is always active. If a fault is detected, the PWM outputs are disabled according to the disable mapping register.

36.3.39.2 Fault Status register (FSTS)

Offset: 0x014E Access: R/W

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	FTEST	FFPIN				FFULL				FFLAG			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 463. Fault Status register (FSTS)

Table 564. FSTS field descriptions

Field	Description
3 FTEST	Fault Test These read/write bit is used to simulate a fault condition. Setting this bit will cause a simulated fault to be sent into all of the fault filters. The condition will propagate to the fault flags and possibly the PWM outputs depending on the DISMAP settings. Clearing this bit removes the simulated fault condition. 1 Cause a simulated fault 0 No fault
4:7 FFPIN	Filtered Fault Pins These read-only bits reflect the current state of the filtered FAULTx pins converted to high polarity. A logic 1 indicates a fault condition exists on the filtered FAULTx pin. A reset has no effect on FFPIN.
8:11 FFULL	Full Cycle These read/write bits are used to control the timing for re-enabling the PWM outputs after a fault condition. These bits apply to both automatic and manual clearing of a fault condition. 1 PWM outputs are re-enabled only at the start of a full cycle. 0 PWM outputs are re-enabled at the start of a full or half cycle.
12:15 FFLAG	Fault Flags These read-only flag is set within two CPU cycles after a transition to active on the FAULTx pin. Clear FFLAGx by writing a logic one to it. A reset clears FFLAG. 1 Fault on the FAULTx pin 0 No fault on the FAULTx pin

36.3.39.3 Fault Filter register (FFILT)

Offset: 0x0150 Access: R/W

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	GSTR	0	0	0	0	FILT_CNT			FILT_PER							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 464. Fault Filter register (FFILT)

The settings in this register are shared among each of the fault input filters.

Table 565. FFILT field descriptions

Field	Description
0 GSTR	Fault Glitch Stretch Enable This bit is used to enable the fault glitch stretching logic. This logic ensures that narrow fault glitches are stretched to be at least 2 IPBus clock cycles wide. In some cases a narrow fault input can cause problems due to the short PWM output shutdown/re-activation time. The stretching logic ensures that a glitch on the fault input, when the fault filter is disabled, will be registered in the fault flags. 1 Input fault signals will be stretched to at least 2 IPBus clock cycles. 0 Fault input glitch stretching is disabled.
5:7 FILT_CNT	Fault Filter Count These bits represent the number of consecutive samples that must agree prior to the input filter accepting an input transition. A value of 0 represents 3 samples. A value of 7 represents 10 samples. The value of FILT_CNT affects the input latency as described in *Section 36.3.39.4*.
8:15 FILT_PER	Fault Filter Period These bits represent the sampling period (in IPBus clock cycles) of the fault pin input filter. Each input is sampled multiple times at the rate specified by FILT_PER. If FILT_PER is 0x00 (default), then the input filter is bypassed. The value of FILT_PER affects the input latency as described in *Section 36.3.39.4*.

36.3.39.4 Input Filter Considerations

The FILT_PER value should be set such that the sampling period is larger than the period of the expected noise. This way a noise spike will only corrupt one sample. The FILT_CNT value should be chosen to reduce the probability of noisy samples causing an incorrect transition to be recognized. The probability of an incorrect transition is defined as the probability of an incorrect sample raised to the FILT_CNT+3 power.

The values of FILT_PER and FILT_CNT must also be traded off against the desire for minimal latency in recognizing input transitions. Turning on the input filter (setting FILT_PER to a non-zero value) introduces a latency of ((FILT_CNT+4) x FILT_PER x IPBus clock period). Note that even when the filter is enabled, there is a combinational path to disable the PWM outputs. This is to ensure rapid response to fault conditions and also to ensure fault response if the PWM module loses its clock. The latency induced by the filter will be seen in the time to set the FFLAG and FFPIN bits of the FSTS register.

36.4 Functional Description

36.4.1 Block Diagram

A block diagram of the PWM is shown in *Figure 419*.

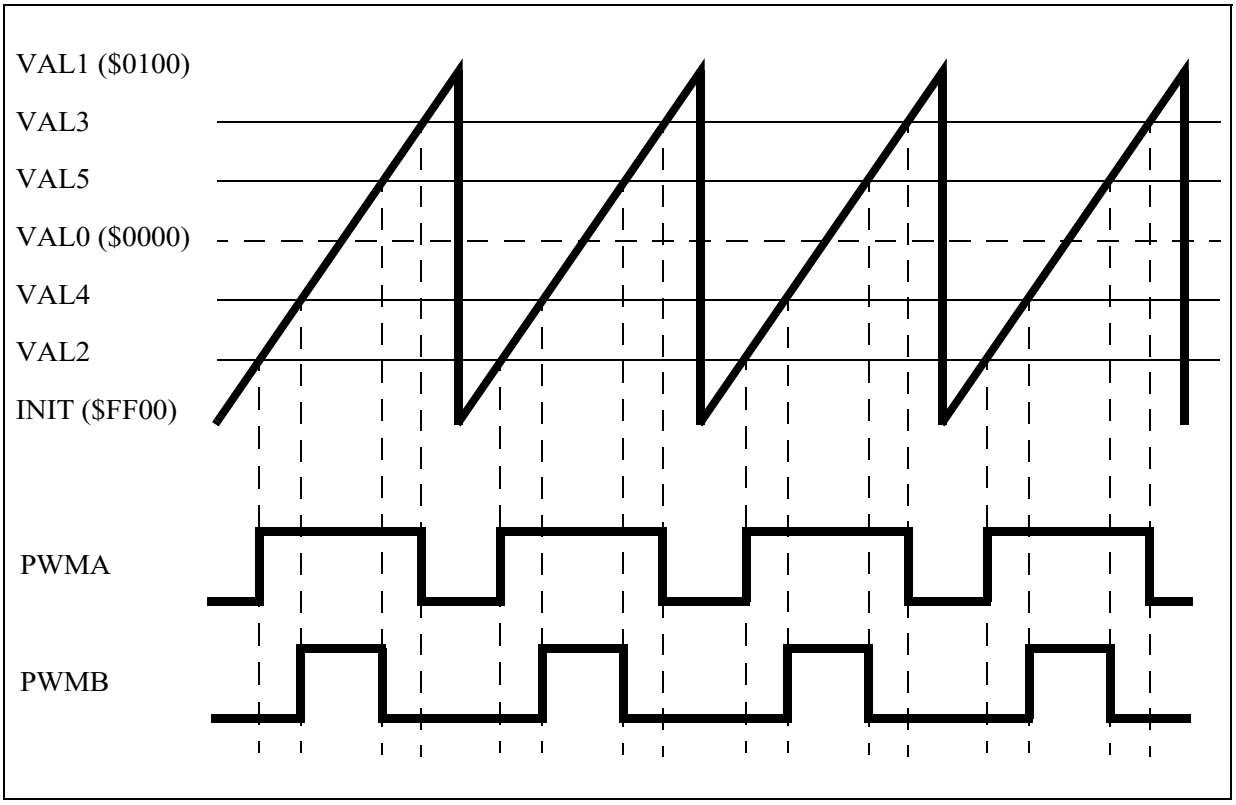
36.4.2 PWM Capabilities

This section describes some capabilities of the PWM module.

36.4.2.1 Center Aligned PWMs

Each submodule has its own timer that is capable of generating PWM signals on two output pins. The edges of each of these signals are controlled independently as shown in *Figure 465*.

Figure 465. Center Aligned Example



The submodule timers only count in the up direction and then reset to the INIT value. Instead of having a single value that determines pulse width, there are two values that must be specified: the turn on edge and the turn off edge. This double action edge generation not only gives the user control over the pulse width, but over the relative alignment of the signal as well. As a result, there is no need to support separate PWM alignment modes since the PWM alignment mode is inherently a function of the turn on and turn off edge values.

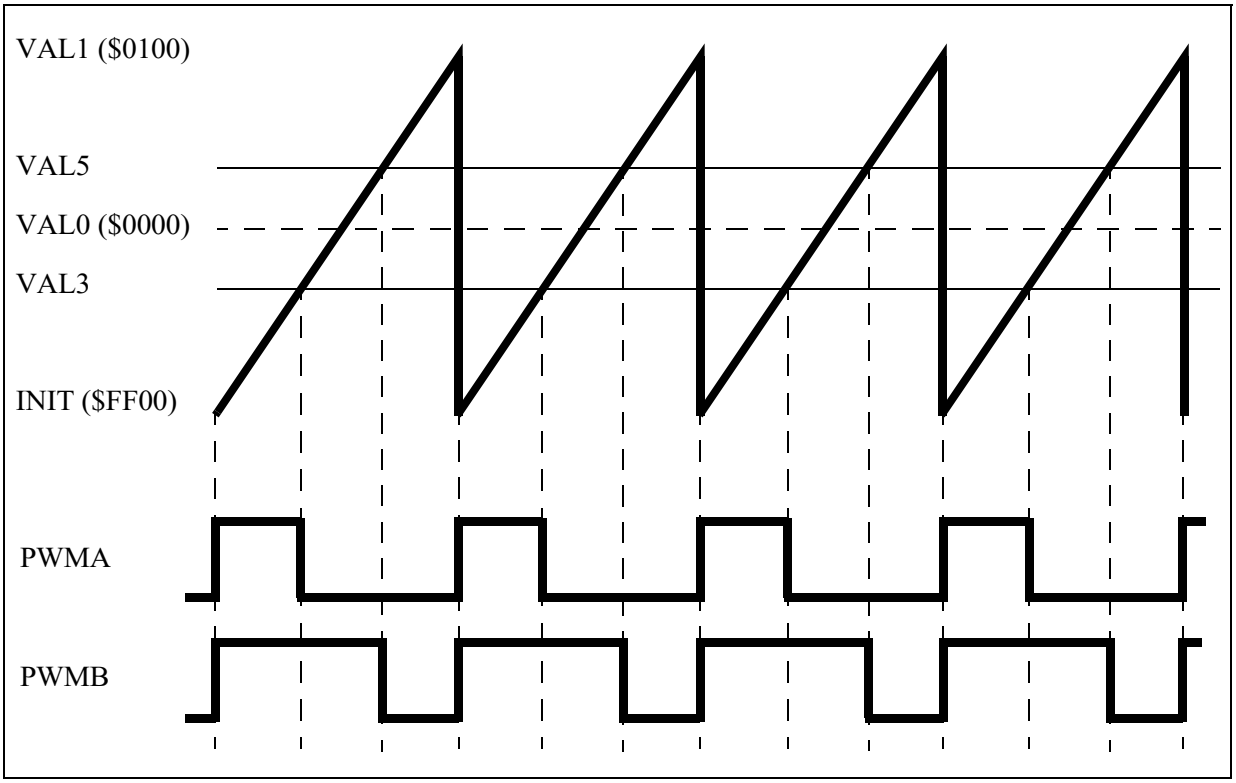
Figure 465 also illustrates an additional enhancement to the PWM generation process. When the counter resets, it is reloaded with a user specified value, which may or may not be zero. If the value chosen happens to be the 2's complement of the modulus value, then the PWM generator operates in "signed" mode. This means that if each PWM's turn on and turn off edge values are also the same number but only different in their sign, the "on" portion of the output signal will be centered around a count value of zero. Therefore, only one PWM value needs to be calculated in software and then this value and its negative are provided to the submodule as the turn off and turn on edges respectively. This technique will result in a pulse width that always consists of an odd number of timer counts. If all PWM signal edge calculations follow this same convention, then the signals will be center aligned with respect to each other, which is the goal. Of course, center alignment between the signals is not restricted to symmetry around the zero count value, as any other number

would also work. However, centering on zero provides the greatest range in signed mode and also simplifies the calculations.

36.4.2.2 Edge Aligned PWMs

When the turn on edge for each pulse is specified to be the INIT value, then edge aligned operation results, as illustrated in *Figure 466*. Therefore, only the turn off edge value needs to be periodically updated to change the pulse width.

Figure 466. Edge Aligned Example (INIT=VAL2=VAL4)



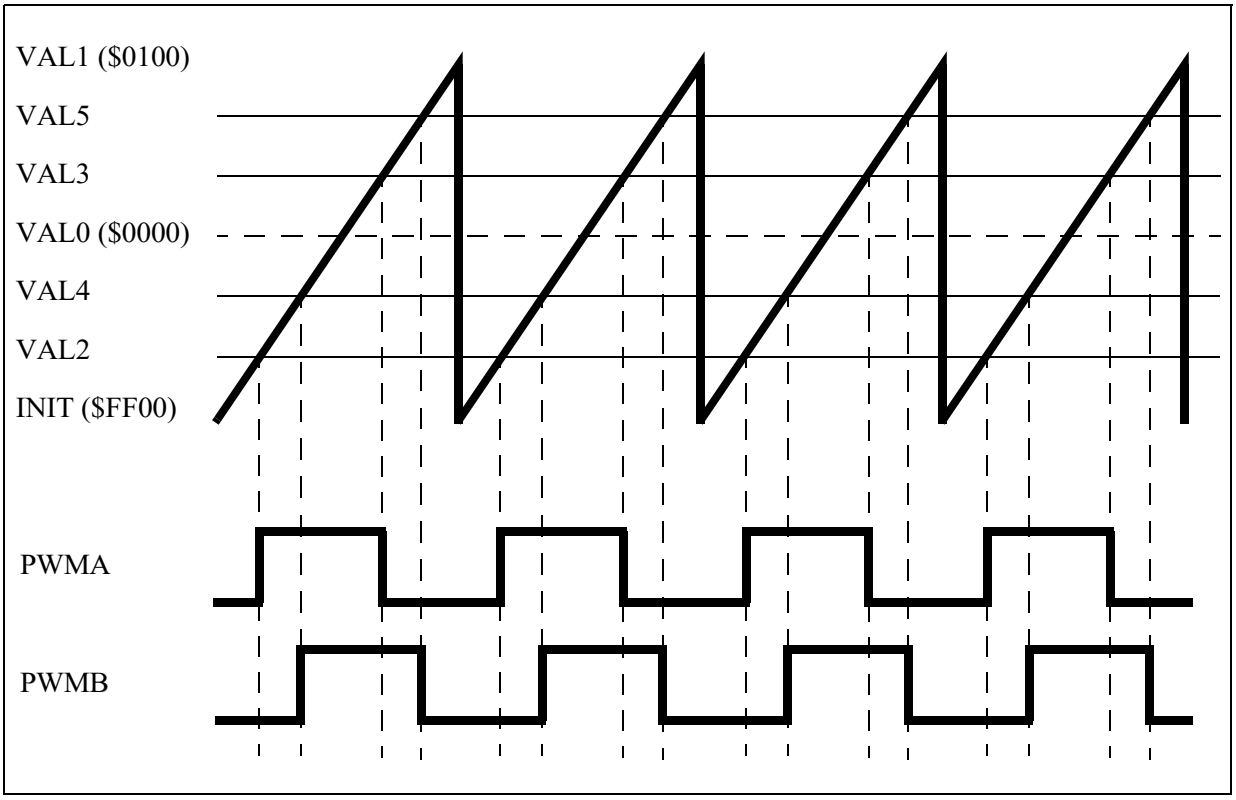
With edge aligned PWMs, another example of the benefits of signed mode can be seen. A common way to drive an H-bridge is to use a technique called "bipolar" PWMs where a 50% duty cycle results in zero volts on the load. Duty cycles less than 50% result in negative load voltages and duty cycles greater than 50% generate positive load voltages. If the module is set to signed mode operation (the INIT and VAL1 values are the same number with opposite signs), then there is a direct proportionality between the PWM turn off edge value and the motor voltage, INCLUDING the sign. So once again, signed mode of operation simplifies the software interface to the PWM module since no offset calculations are required to translate the output variable control algorithm to the voltage on an H-Bridge load.

36.4.2.3 Phase Shifted PWMs

In the previous sections, the benefits of signed mode of operation were discussed in the context of simplifying the required software calculations by eliminating the requirement to bias up signed variables before applying them to the module. However, if numerical biases ARE applied to the turn on and turn off edges of different PWM signal, the signals will be phase shifted with respect to each other, as illustrated in *Figure 467*. This results in certain

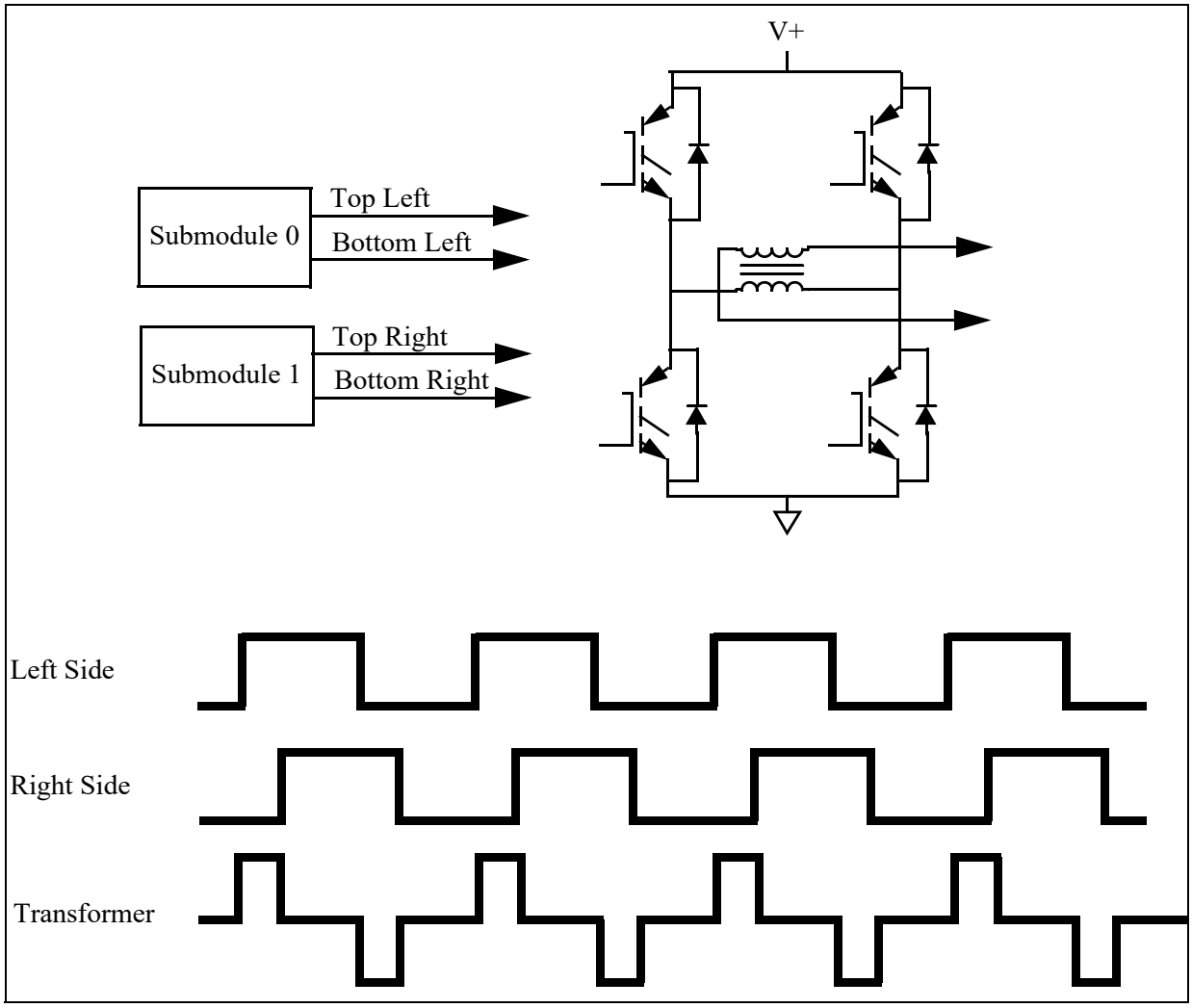
advantages when applied to a power stage. For example, when operating a multi-phase inverter at a low modulation index, all of the PWM switching edges from the different phases occur at nearly the same time. This can be troublesome from a noise standpoint, especially if ADC readings of the inverter must be scheduled near those times. Phase shifting the PWM signals can open up timing windows between the switching edges to allow a signal to be sampled by the ADC. However, phase shifting does NOT affect the duty cycle so average load voltage is not affected.

Figure 467. Phase Shifted Outputs Example



An additional benefit of phase shifted PWMs can be seen in *Figure 468*. In this case, an H-Bridge circuit is driven by 4 PWM signals to control the voltage waveform on the primary of a transformer. Both left and right side PWMs are configured to always generate a square wave with 50% duty cycle. This works out nicely for the H-Bridge since no narrow pulse widths are generated reducing the high frequency switching requirements of the transistors. Notice that the square wave on the right side of the H-Bridge is phase shifted compared to the left side of the H-Bridge. As a result, the transformer primary sees the bottom waveform across its terminals. The RMS value of this waveform is directly controlled by the amount of phase shift of the square waves. Regardless of the phase shift, no DC component appears in the load voltage as long as the duty cycle of each square wave remains at 50% making this technique ideally suited for transformer loads. As a result, this topology is frequently used in industrial welders to adjust the amount of energy delivered to the weld arc.

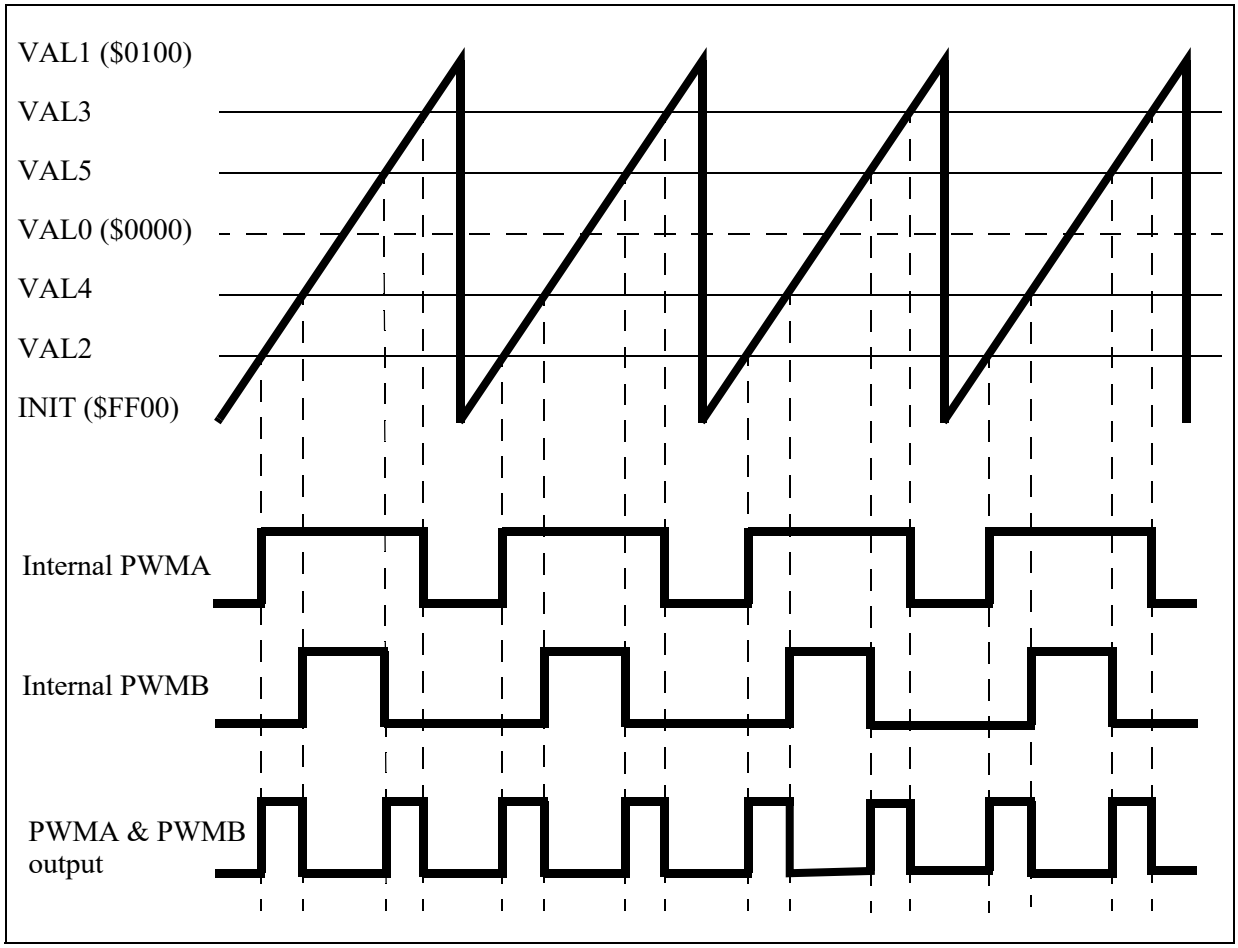
Figure 468. Phase Shifted PWMs Applied to a Transformer Primary



36.4.2.4 Double Switching PWMs

Double switching PWM output is supported to aid in single shunt current measurement and three phase reconstruction. This method support two independent rising edges and two independent falling edges per PWM cycle. The VAL2 and VAL3 registers are used to generate the even channel (labelled as PWMA in the figure) while VAL4 and VAL5 are used to generate the odd channel. The two channels are combined using XOR logic (see *Figure 479*) as shown in *Figure 469*. The DBLPWM signal can be run through the deadtime insertion logic.

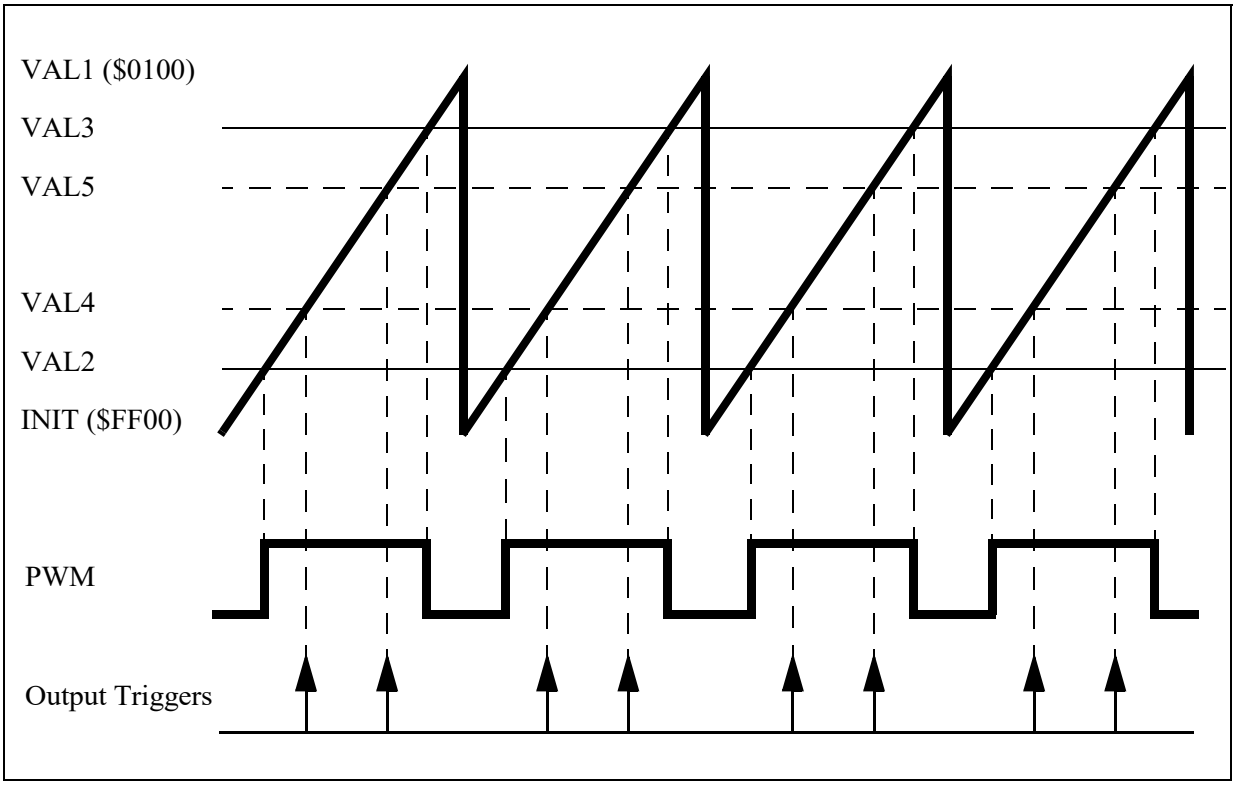
Figure 469. Double Switching Output Example



36.4.2.5 ADC Triggering

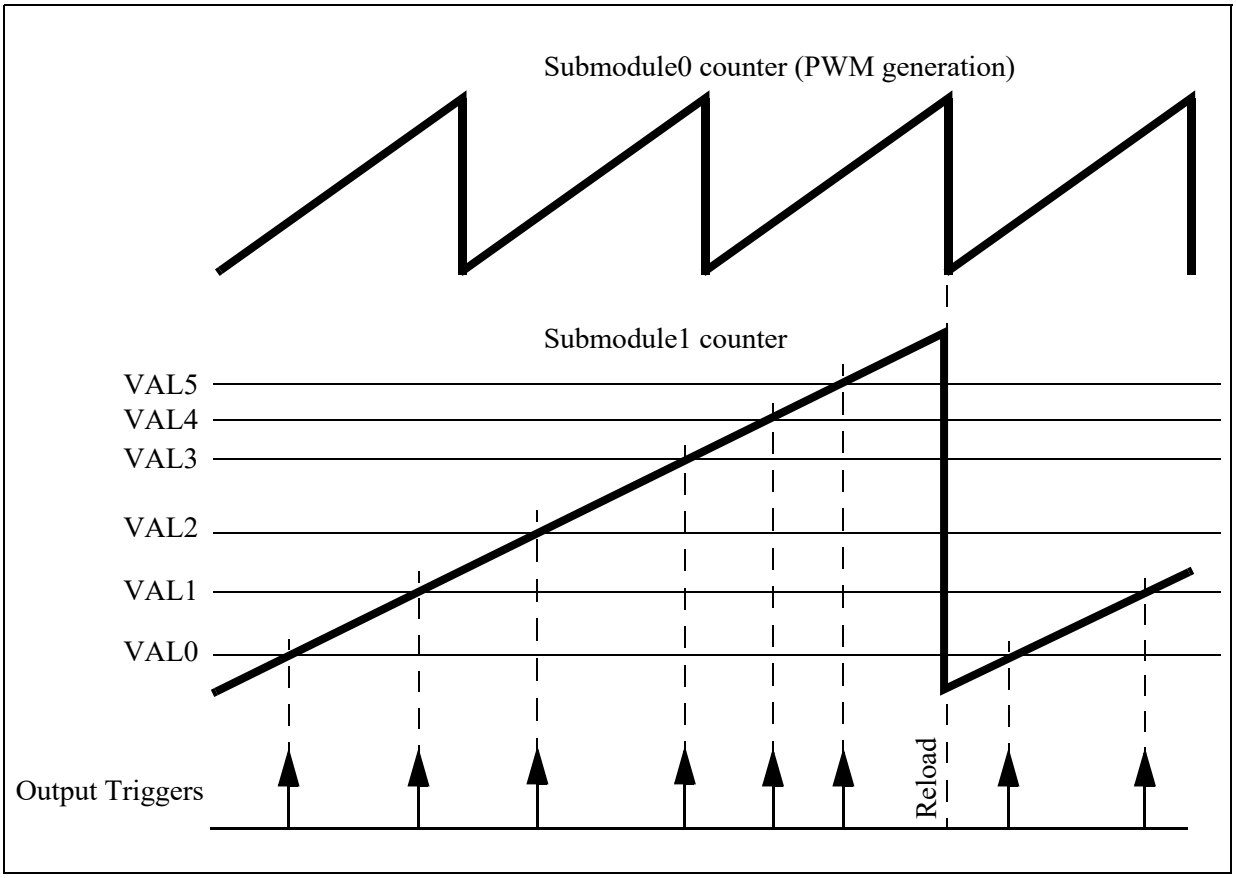
In cases where the timing of the ADC triggering is critical, it must be scheduled as a hardware event instead of software activated. With this PWM module, multiple ADC triggers can be generated in hardware per PWM cycle without the requirement of another timer module. *Figure 470* shows how this is accomplished. When specifying complimentary mode of operation, only two edge comparators are required to generate the output PWM signals for a given submodule. This means that the other comparators are free to perform other functions. In this example, the software doesn't have to quickly respond after the first conversion to set up other conversions that must occur in the same PWM cycle.

Figure 470. Multiple Output Trigger Generation in Hardware



Since each submodule has its own timer, it is possible for each submodule to run at a different frequency. One of the options possible with this PWM module is to have one or more submodules running at a lower frequency, but still synchronized to the timer in submodule0. *Figure 471* shows how this feature can be used to schedule ADC triggers over multiple PWM cycles. A suggested use for this configuration would be to use the lower frequency submodule to control the sampling frequency of the software control algorithm where multiple ADC triggers can now be scheduled over the entire sampling period. In *Figure 471*, ALL submodule comparators are shown being used for ADC trigger generation.

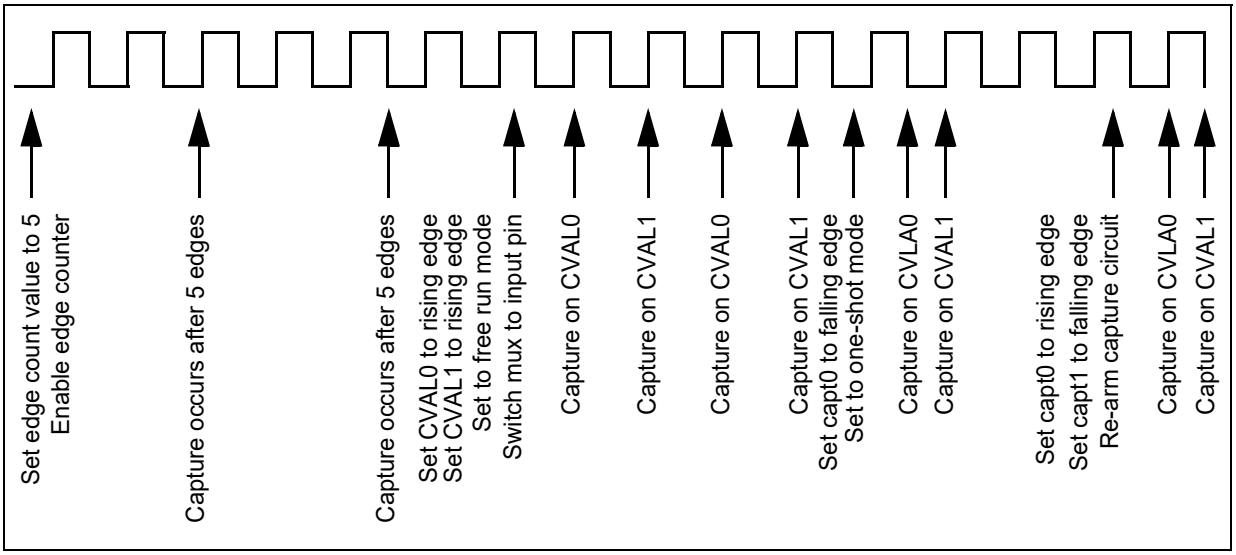
Figure 471. Multiple Output Triggers Over Several PWM Cycles



36.4.2.6 Enhanced Capture Capabilities (E-Capture)

When a PWM pin is not being used for PWM generation, it can be used to perform input captures. Recall that for PWM generation BOTH edges of the PWM signal are specified via separate compare register values. When programmed for input capture, both of these registers work on the same pin to capture multiple edges, toggling from one to the other in either a free running or one-shot fashion. By simply programming the desired edge of each capture circuit, period and pulse width of an input signal can easily be measured without the requirement to re-arm the circuit. In addition, each edge of the input signal can clock an 8 bit counter where the counter output is compared to a user specified value (EDGCMP). When the counter output equals EDGCMP, the value of the submodule timer is captured and the counter is automatically reset. This feature allows the module to count a specified number of edge events and then perform a capture and interrupt. *Figure 472* illustrates some of the functionality of the E-Capture circuit.

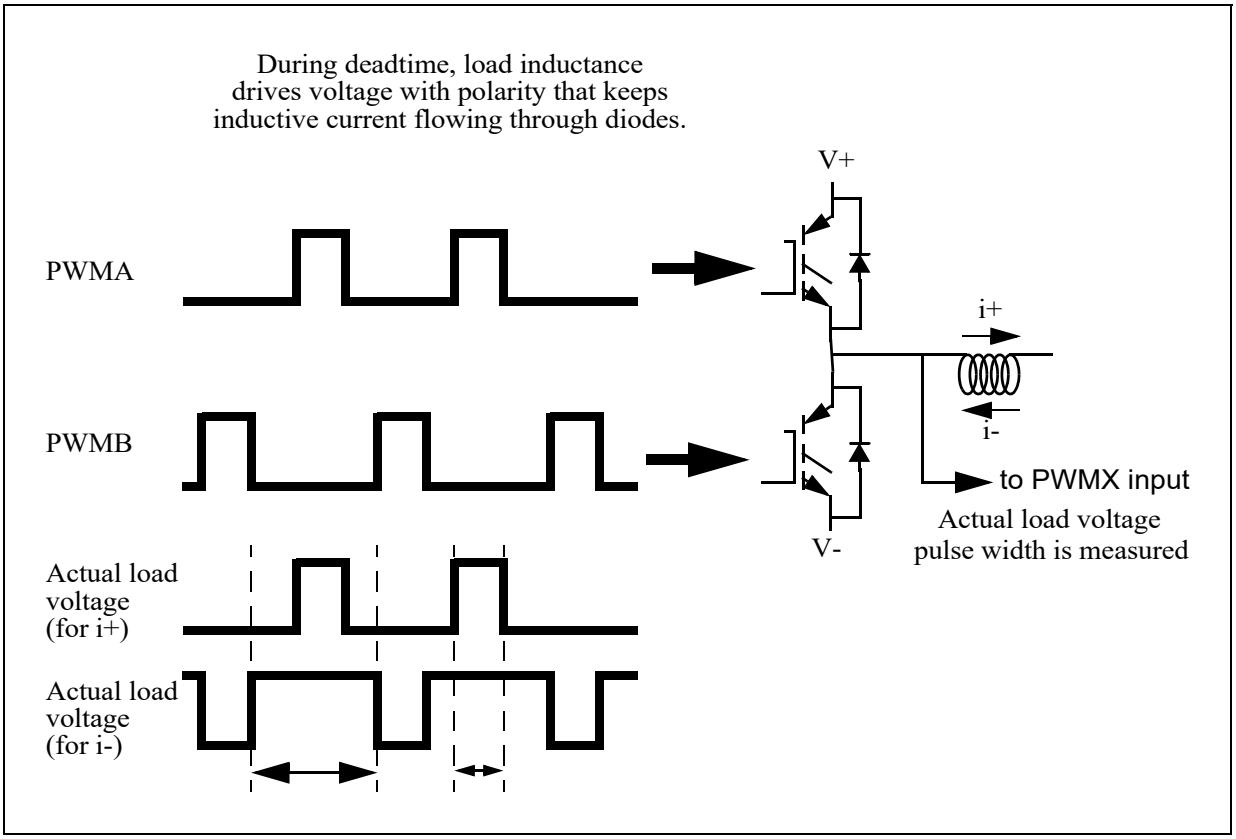
Figure 472. Capture Capabilities of the E-Capture Circuit



When a submodule is being used for PWM generation, its timer counts up to the modulus value used to specify the PWM frequency and then is re-initialized. Therefore, using this timer for input captures on one of the other pins (for example PWMX) has limited utility since it does not count through all of the numbers and the timer reset represents a discontinuity in the 16 bit number range. However, when measuring a signal that is synchronous to the PWM frequency, the timer modulus range is perfectly suited for the application. As an example, consider *Figure 473*. In this application the output of a PWM power stage is connected to the PWMX pin that is configured for free running input captures. Specifically, the CVAL0 capture circuitry is programmed for rising edges and the CVAL1 capture circuitry is set for falling edges. This will result in new load pulse width data being acquired every PWM cycle. To calculate the pulse width, simply subtract the CVAL0 register value from the CVAL1 register value. This measurement is extremely beneficial when performing dead-time distortion correction on a half bridge circuit driving an inductive load. Also, these values can be directly compared to the VALx registers responsible for generating the PWM outputs to obtain a measurement of system propagation delays.

Note: CVAL0-CVAL1 are used for PWMx, CVAL2 and CVAL3 are used for PWMA and finally, VAL4 and CVAL5 are used for PWMB.

Figure 473. Output Pulse Width Measurement Possible with the E-Capture Circuit



36.4.2.7 Synchronous Switching of Multiple Outputs

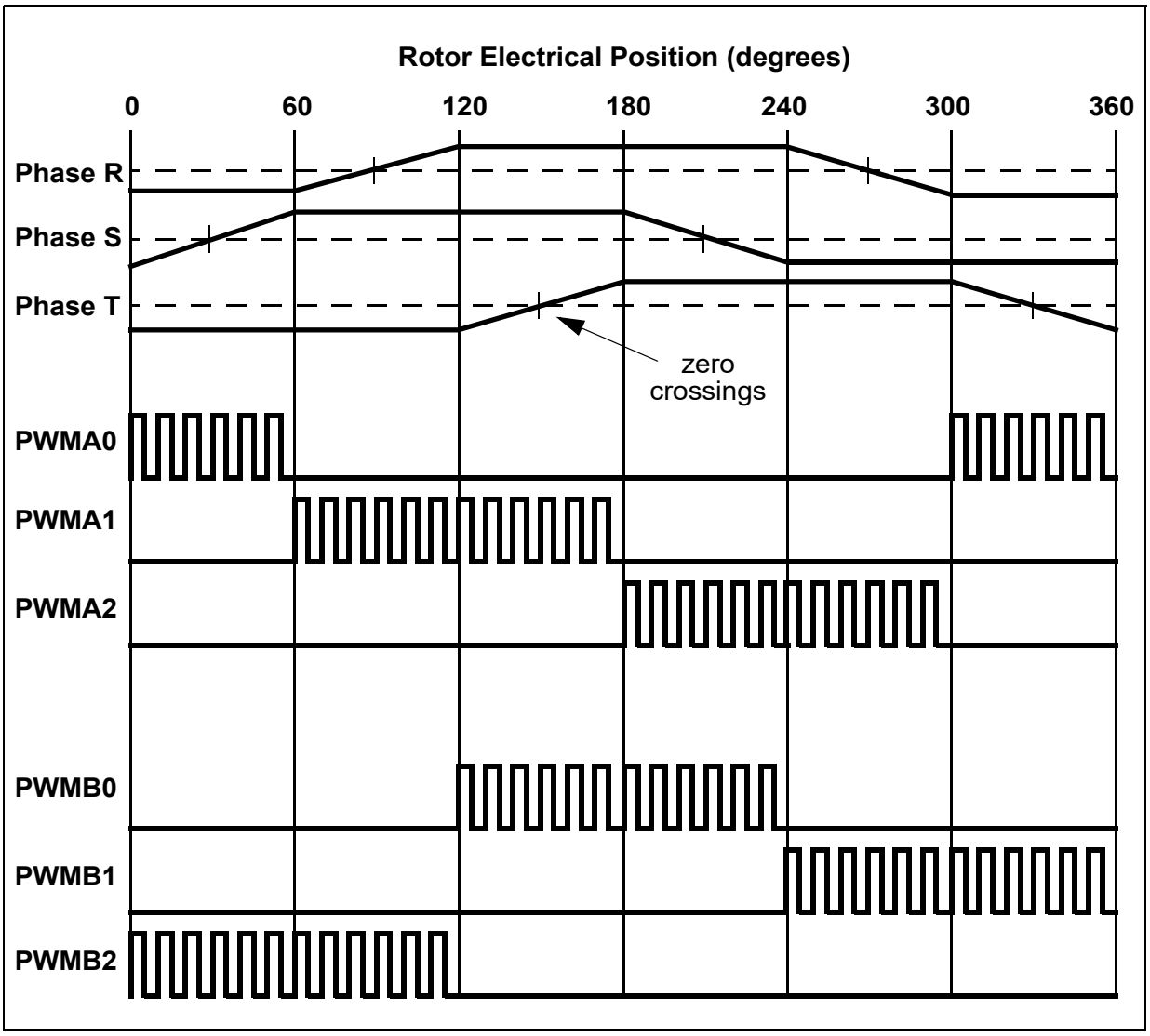
Before the PWM signals are routed to the output pins, they are processed by a hardware block that permits all submodule outputs to be switched synchronously. This feature can be extremely useful in commutated motor applications where the next commutation state can be laid in ahead of time and then immediately switched to the outputs when the appropriate condition or time is reached. Not only do all the changes occur synchronously on all submodule outputs, but they occur IMMEDIATELY after the trigger event occurs eliminating any interrupt latency.

The synchronous output switching is accomplished via a signal called FORCE_OUT. This signal originates from the local FORCE bit within the submodule, from submodule0, or from external to the PWM module and, in most cases, is supplied from an external timer channel configured for output compare. In a typical application, software sets up the desired states of the output pins in preparation for the next FORCE_OUT event using the SWCOUT and DTSRCSEL registers. This selection lays dormant until the FORCE_OUT signal transitions and then all outputs are switched simultaneously. The signal switching is performed upstream from the deadtime generator so that any abrupt changes that might occur do not violate deadtime on the power stage when in complementary mode.

Figure 474 shows a popular application that can benefit from this feature. On a brushless DC motor it is desirable on many cases to spin the motor without need of hall-effect sensor feedback. Instead, the back EMF of the motor phases is monitored and this information is used to schedule the next commutation event. The top waveforms of *Figure 474* are a simplistic representation of these back EMF signals. Timer compare events (represented by the long vertical lines in the diagram) are scheduled based on the zero crossings of the

back-EMF waveforms. The PWM module is configured via software ahead of time with the next state of the PWM pins in anticipation of the compare event. When it happens, the output compare of the timer drives the FORCE_OUT signal which immediately changes the state of the PWM pins to the next commutation state with no software latency.

Figure 474. Sensorless BLDC Commutation Using the Force Out Function



36.4.2.8 Phase shifted full bridge for DC/DC converters

Use CTRL1[PHSSHFT] to allow the EXT_SWITCH input to control the output of PWMA0/PWMB0. The EXT_SWITCH input switches the source of the submodule0 output from the VAL* registers within submodule0 to the VAL* registers in submodule1. This muxing can be seen in *Figure 478*.

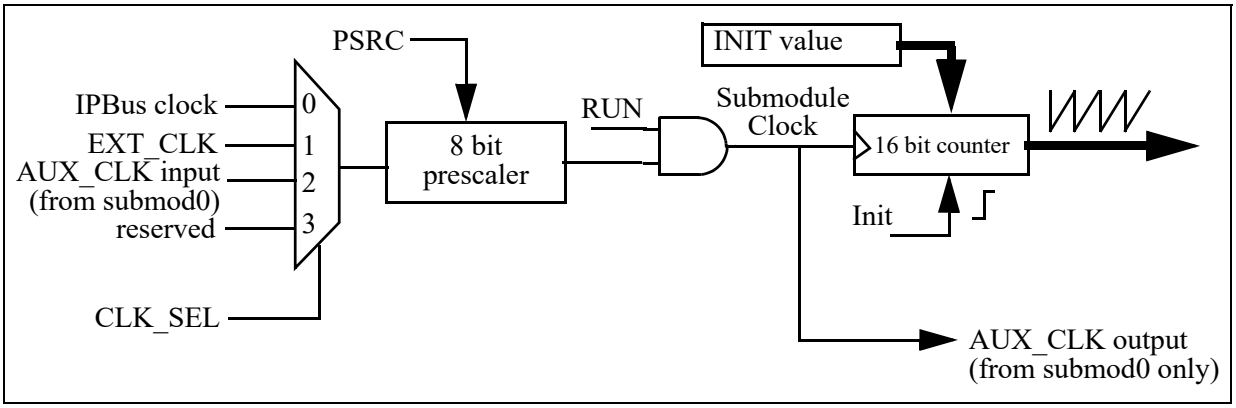
36.4.3 Functional Details

This section describes the implementation of various features of the PWM in greater detail.

36.4.3.1 PWM Clocking

Figure 475 shows the logic used to generate the main counter clock. Each submodule can select between three clock signals: the IPBus clock, EXT_CLK, and AUX_CLK. The EXT_CLK is generated by an on-chip resource such as a Timer module and goes to all of the submodules. The AUX_CLK signal is broadcast from submodule0 and can be selected as the clock source by other submodules so that the 8 bit prescaler and RUN bit from submodule0 can control all of the submodules. When AUX_CLK is selected as the source for the submodule clock, the RUN bit from submodule0 is used instead of the local RUN bit from this submodule.

Figure 475. Clocking Block Diagram for Each PWM Submodule



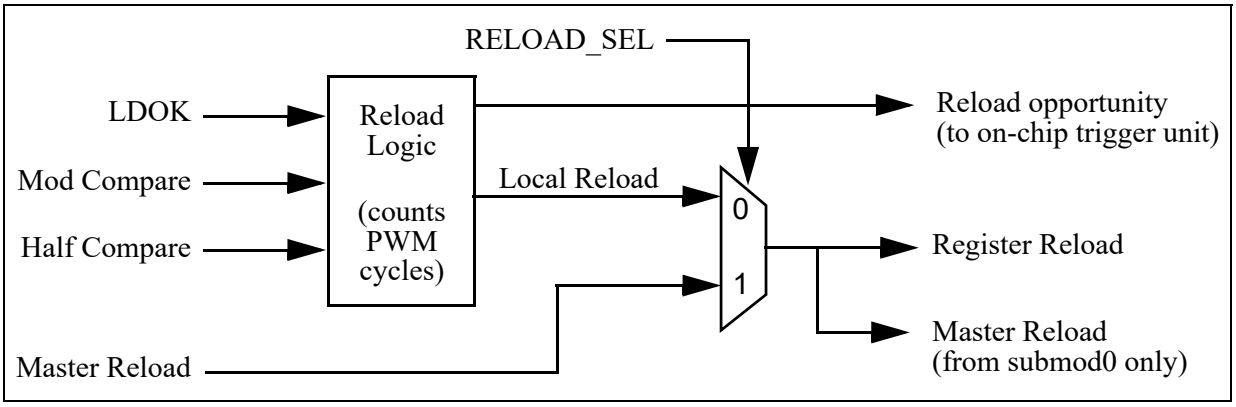
To permit lower PWM frequencies, the prescaler produces the PWM clock frequency by dividing the IPBus clock frequency by 1-128. The prescaler bits, PRSC, in the control register (CTRL1), select the prescaler divisor. This prescaler is buffered and will not be used by the PWM generator until the LDOK bit is set and a new PWM reload cycle begins or LDMOD is set.

36.4.3.2 Register Reload Logic

The register reload logic is used to determine when the outer set of registers for all double buffered register pairs will be transferred to the inner set of registers. The register reload event can be scheduled to occur every "n" PWM cycles using the LDFQ bits and the FULL bit. A half cycle reload option is also supported (HALF) where the reload can take place in the middle of a PWM cycle. The half cycle point is defined by the VAL0 register and does not have to be exactly in the middle of the PWM cycle.

As illustrated in *Figure 476* the reload signal from submodule0 can be broadcast as the Master Reload signal allowing the reload logic from submodule0 to control the reload of registers in other submodules.

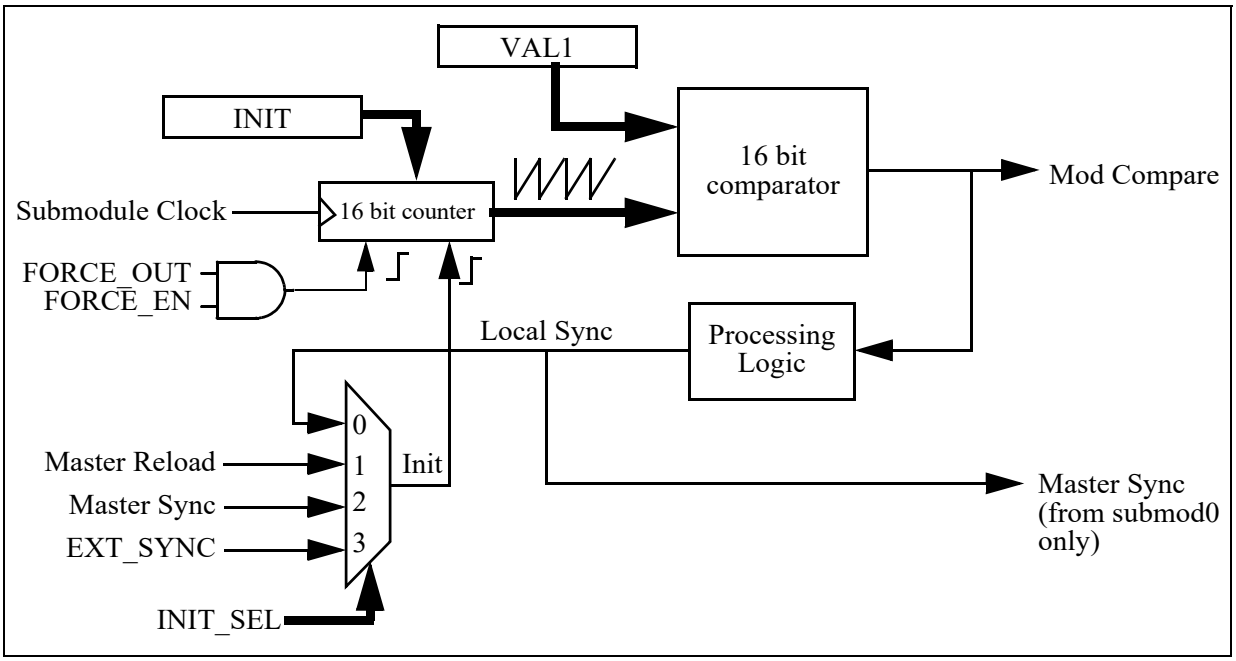
Figure 476. Register Reload Logic



36.4.3.3 Counter Synchronization

Referring to *Figure 477*, the 16 bit counter will count up until its output equals VAL1 which is used to specify the counter modulus value. The resulting compare causes a rising edge to occur on the Local Sync signal which is one of four possible sources used to cause the 16 bit counter to be initialized with INIT. If Local Sync is selected as the counter initialization signal, then VAL1 within the submodule effectively controls the timer period (and thus the PWM frequency generated by that submodule) and everything works on a local level.

Figure 477. Submodule Timer Synchronization



The Master Sync signal originates as the Local Sync from submodule0. If configured to do so, the timer period of any submodule can be locked to the period of the timer in submodule0.

The EXT_SYNC signal originates on chip or off chip depending on the system architecture. This signal may be selected as the source for counter initialization so that an external source can control the period of all submodules.

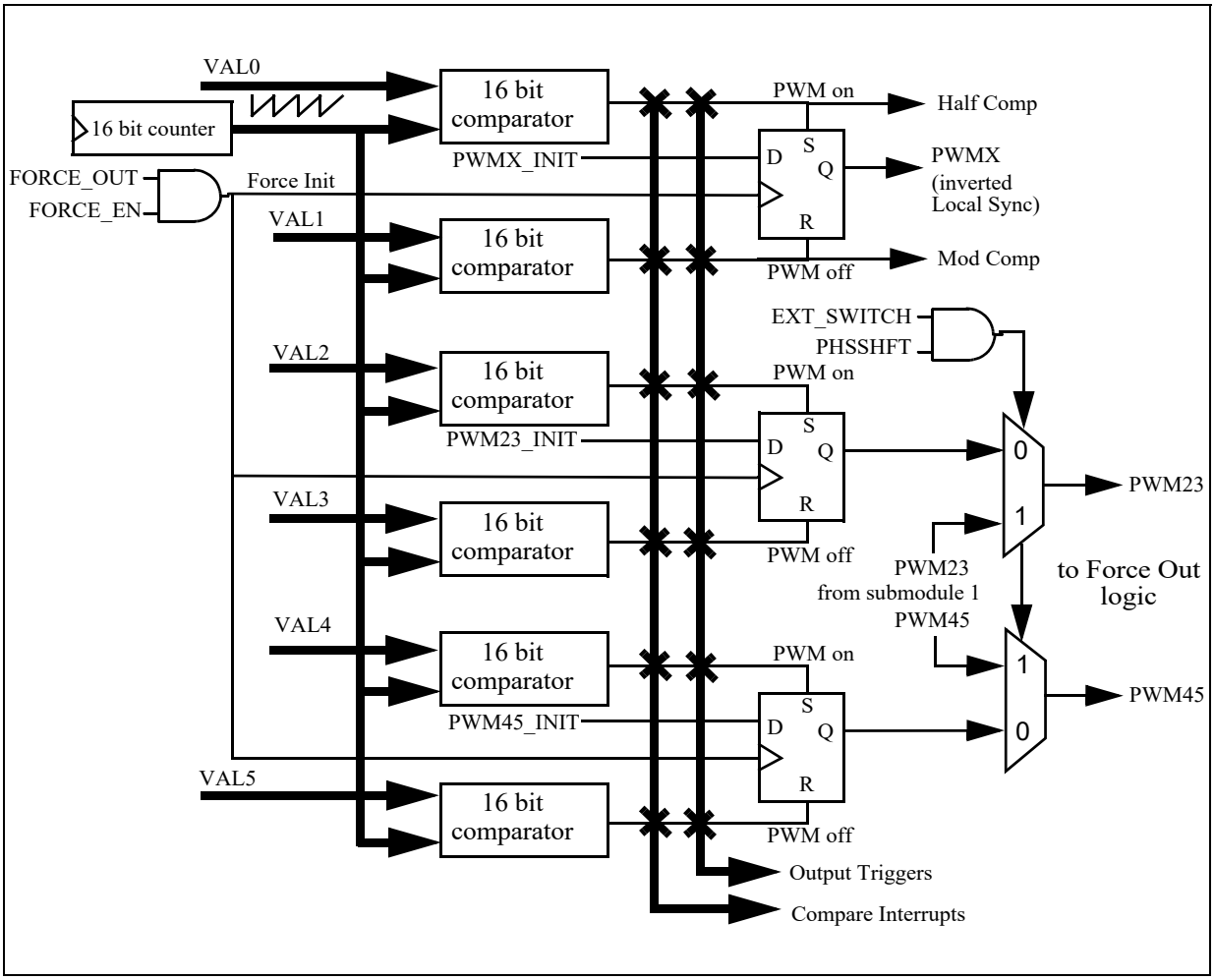
If the Master Reload signal is selected as the source for counter initialization, then the period of the counter will be locked to the register reload frequency of submodule0. Since the reload frequency is usually commensurate to the sampling frequency of the software control algorithm, the submodule counter period will therefore equal the sampling period. As a result, this timer can be used to generate output compares or output triggers over the entire sampling period which may consist of several PWM cycles. The Master Reload signal can only originate from submodule0.

The counter can optionally initialize upon the assertion of the FORCE_OUT signal assuming that the FORCE_EN bit is set. As indicated by *Figure 477*, this constitutes a second init input into the counter which will cause the counter to initialize regardless of which signal is selected as the counter init signal. The FORCE_OUT signal is provided mainly for commutated applications. When PWM signals are commutated on an inverter controlling a brushless DC motor, it is necessary to restart the PWM cycle at the beginning of the commutation interval. This action effectively resynchronizes the PWM waveform to the commutation timing. Otherwise, the average voltage applied to a motor winding integrated over the entire commutation interval will be a function of the timing between the asynchronous commutation event with respect to the PWM cycle. The effect is more critical at higher motor speeds where each commutation interval may consist of only a few PWM cycles. If the counter is not initialized at the start of each commutation interval, the result will be an oscillation caused by the beating between the PWM frequency and the commutation frequency.

36.4.3.4 PWM Generation

Figure 478 illustrates how PWM generation is accomplished in each submodule. In each case, two comparators and associated VALx registers are utilized for each PWM output signal. One comparator and VALx register is used to control the turn on edge while a second comparator and VALx register control the turn off edge.

Figure 478. PWM Generation Hardware



The generation of the Local Sync signal is performed exactly the same way as the other PWM signals in the submodule. While comparator 0 causes a rising edge of the Local Sync signal, comparator 1 generates a falling edge. Comparator 1 is also hardwired to the reload logic to generate the half cycle reload indicator.

If VAL1 is controlling the modulus of the counter and VAL0 is half of the VAL1 register minus the INIT value, then the half cycle reload pulse will occur exactly half way through the timer count period and the Local Sync will have a 50% duty cycle. On the other hand, if the VAL1 and VAL0 registers are not required for register reloading or counter initialization, they can be used to modulate the duty cycle of the Local Sync signal effectively turning it into an auxiliary PWM signal (PWMX) assuming that the PWMX pin is not being used for another function such as input capture or deadtime distortion correction. Including the Local Sync signal, each submodule is capable of generating 3 PWM signals where software has complete control over each edge of each of the signals.

If the comparators and edge value registers are not required for PWM generation, they can also be used for other functions such as output compares, generating output triggers, or generating interrupts at timed intervals.

The muxes for PWM23 and PWM45 shown in *Figure 478* are only in submodule0.

If the comparators assert both the set and reset of the flip-flop at the same time, then the flop output goes to 0.

36.4.3.5 Output Compare Capabilities

By using the VALx registers in conjunction with the submodule timer and 16 bit comparators, buffered output compare functionality can be achieved with no additional hardware required. Specifically, the following output compare functions are possible:

- An output compare sets the output high.
- An output compare sets the output low.
- An output compare generates an interrupt.
- An output compare generates an output trigger.

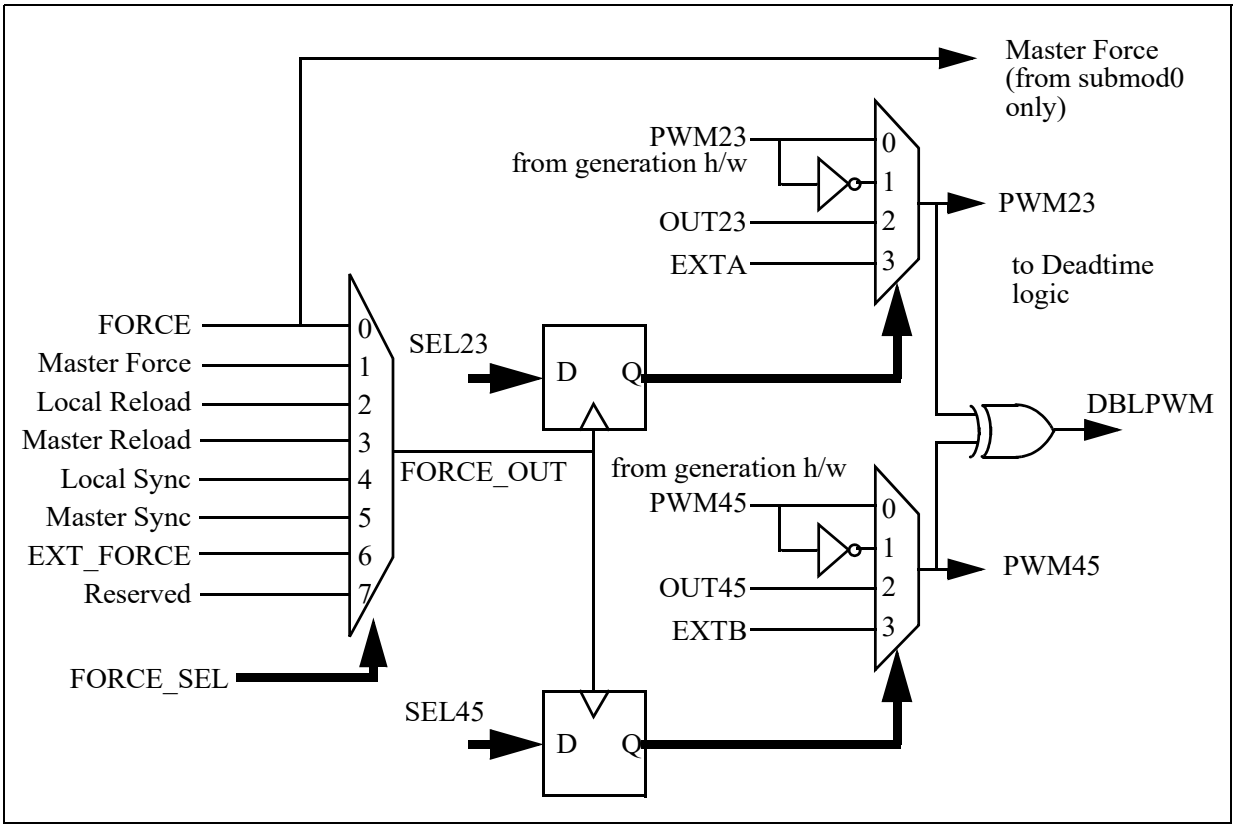
Referring again to *Figure 478*, an output compare is initiated by programming a VALx register for a timer compare which in turn causes the output of the D flip-flop to either set or reset. For example, if an output compare is desired on the PWMA signal that sets it high, VAL2 would be programmed with the counter value where the output compare should take place. However, to prevent the D flip-flop from being reset again after the compare has occurred, the VAL3 register must be programmed to a value outside of the modulus range of the counter. Therefore, a compare that would result in resetting the D flip-flop output would never occur. Conversely, if an output compare is desired on the PWMA signal that sets it low, the VAL3 register is programmed with the appropriate count value and the VAL2 register is programmed with a value outside the counter modulus range. Regardless of whether a high compare or low compare is programmed, an interrupt or output trigger can be generated when the compare event occurs.

36.4.3.6 Force Out Logic

For each submodule software can select between seven signal sources for the FORCE_OUT signal: the local FORCE bit, the Master Force signal from submodule0, the local Reload signal, the Master Reload signal from submodule0, the Local Sync signal, the Master Sync signal from submodule0, or the EXT_FORCE signal from on or off chip depending on the chip architecture. The local signals are used when the user simply wants to change the signals on the output pins of the submodule without regard for synchronization with other submodules. However, if it is required that all signals on all submodule outputs change at the same time, the Master signals or EXT_FORCE signal should be selected.

Figure 479 illustrates the Force logic. The SEL23 and SEL45 fields each choose from one of four signals that can be supplied to the submodule outputs: the PWM signal, the inverted PWM signal, a binary level specified by software via the OUT23 and OUT45 bits, or the EXTA or EXTB alternate external control signals. The selection can be determined ahead of time and, when a FORCE_OUT event occurs, these values are presented to the signal selection mux which immediately switches the requested signal to the output of the mux for further processing downstream.

Figure 479. Force Out Logic



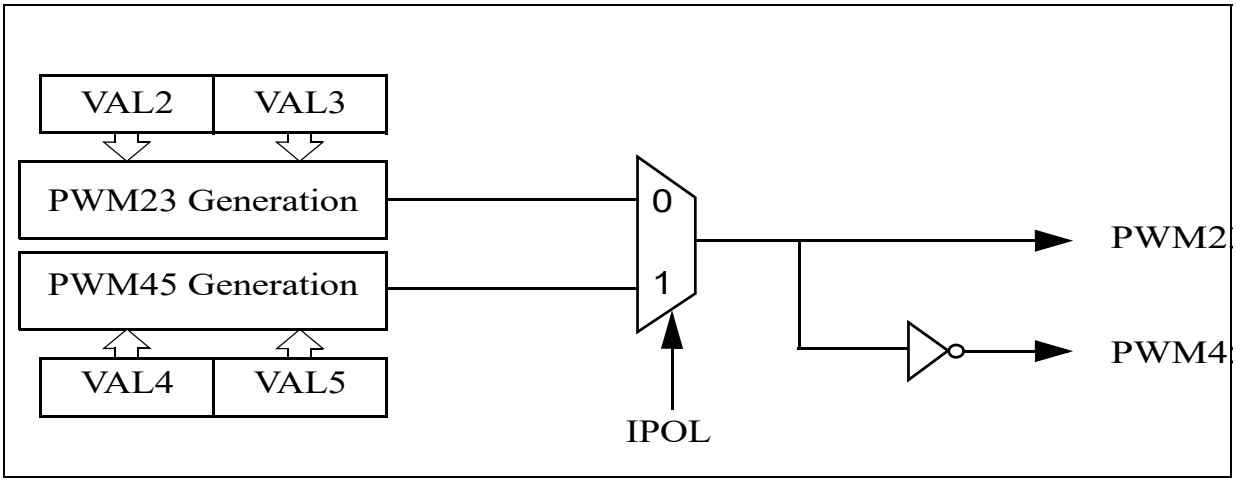
The local FORCE signal of submodule0 can be broadcast as the Master Force signal to other submodules. This feature allows the local FORCE bit of submodule0 to synchronously update all of the submodule outputs at the same time. The EXT_FORCE signal originates from outside the PWM module from a source such as a timer or digital comparators in the Analog-to-Digital Converter.

36.4.3.7 Independent or Complementary Channel Operation

Writing a logic one to the INDEP bit of the CNFG register configures the pair of PWM outputs as two independent PWM channels. Each PWM output is controlled by its own VALx pair operating independently of the other output.

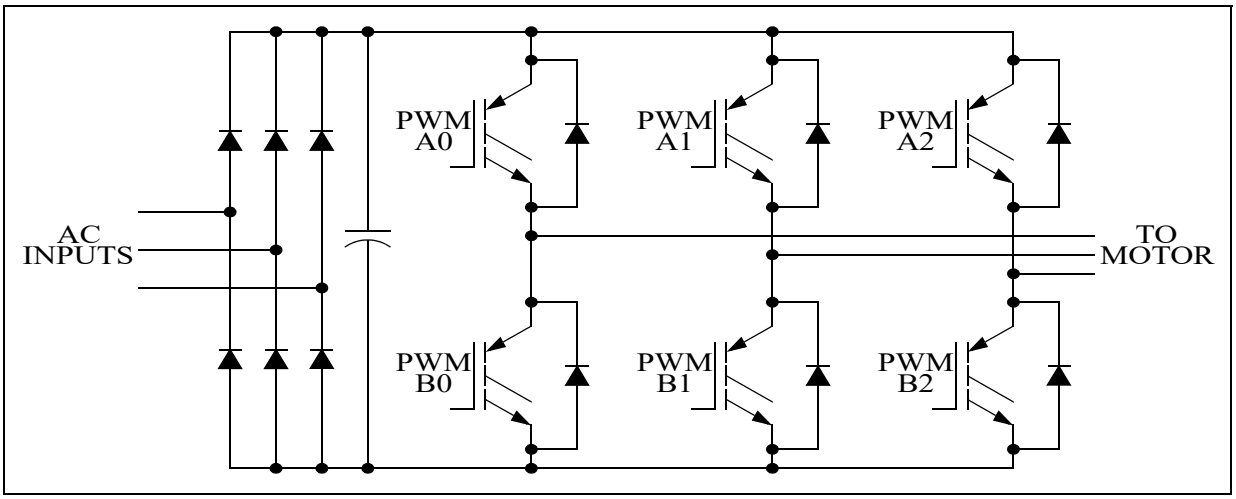
Writing a logic zero to the INDEP bit configures the PWM output as a pair of complementary channels. The PWM pins are paired as shown in *Figure 480* in complementary channel operation. Which signal is connected to the output pin (PWM23 or PWM45) is determined by the IPOL bit.

Figure 480. Complementary Channel Pair



The complementary channel operation is for driving top and bottom transistors in a motor drive circuit, such as the one in *Figure 481*.

Figure 481. Typical 3 Phase AC Motor Drive

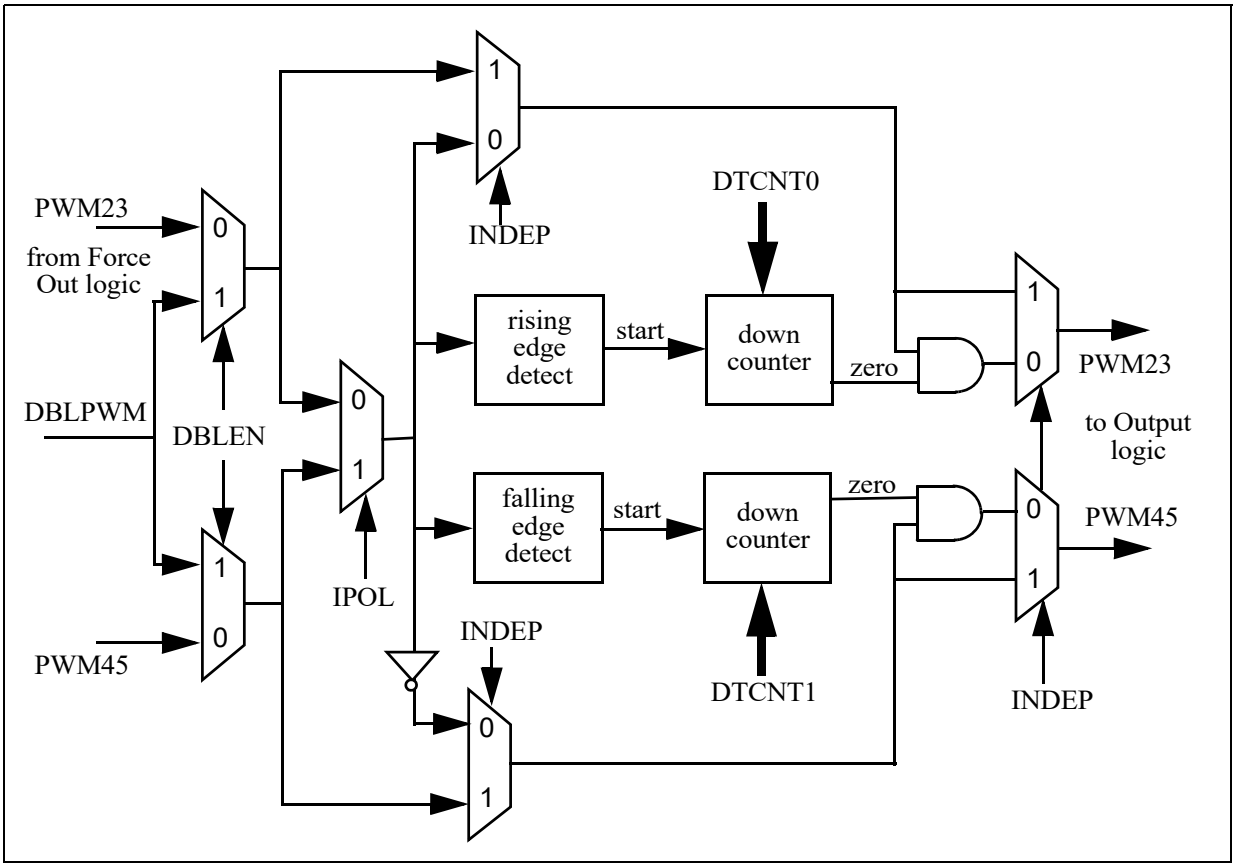


Complementary operation allows the use of the deadtime insertion feature.

36.4.3.8 Deadtime Insertion Logic

Figure 482 shows the deadtime insertion logic of each submodule which is used to create non-overlapping complementary signals when not in independent mode.

Figure 482. Deadtime Insertion and Fine Control Logic



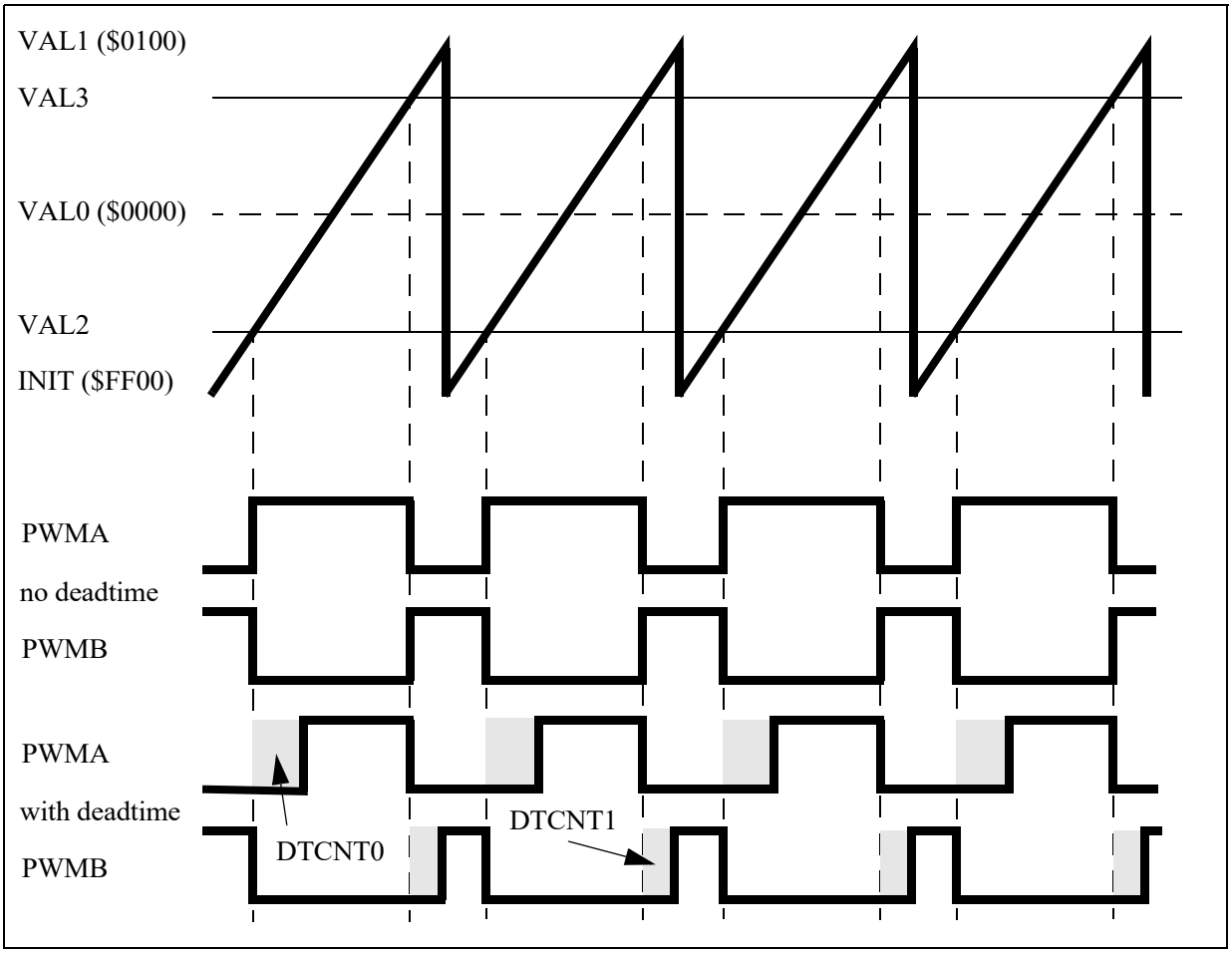
While in the complementary mode, a PWM pair can be used to drive top/bottom transistors, as shown in *Figure 482*. When the top PWM channel is active, the bottom PWM channel is inactive, and vice versa.

Note: To avoid short circuiting the DC bus and endangering the transistor, there must be no overlap of conducting intervals between top and bottom transistor. However, the transistor's characteristics may cause its switching-off time to be longer than its switching-on time. To avoid the conducting overlap of top and bottom transistors, deadtime needs to be inserted in the switching period, as illustrated in Figure 483

The deadtime generators automatically insert software-selectable activation delays into the pair of PWM outputs. The deadtime registers (DTCNT0 and DTCNT1) specify the number of IPBus clock cycles to use for deadtime delay. Every time the deadtime generator inputs change state, deadtime is inserted. Deadtime forces both PWM outputs in the pair to the inactive state.

When deadtime is inserted in complementary PWM signals connected to an inverter driving an inductive load, the PWM waveform on the inverter output will have a different duty cycle than what appears on the output pins of the PWM module. This results in a distortion in the voltage applied to the load. A method of correcting this, adding to or subtracting from the PWM value used, is discussed next.

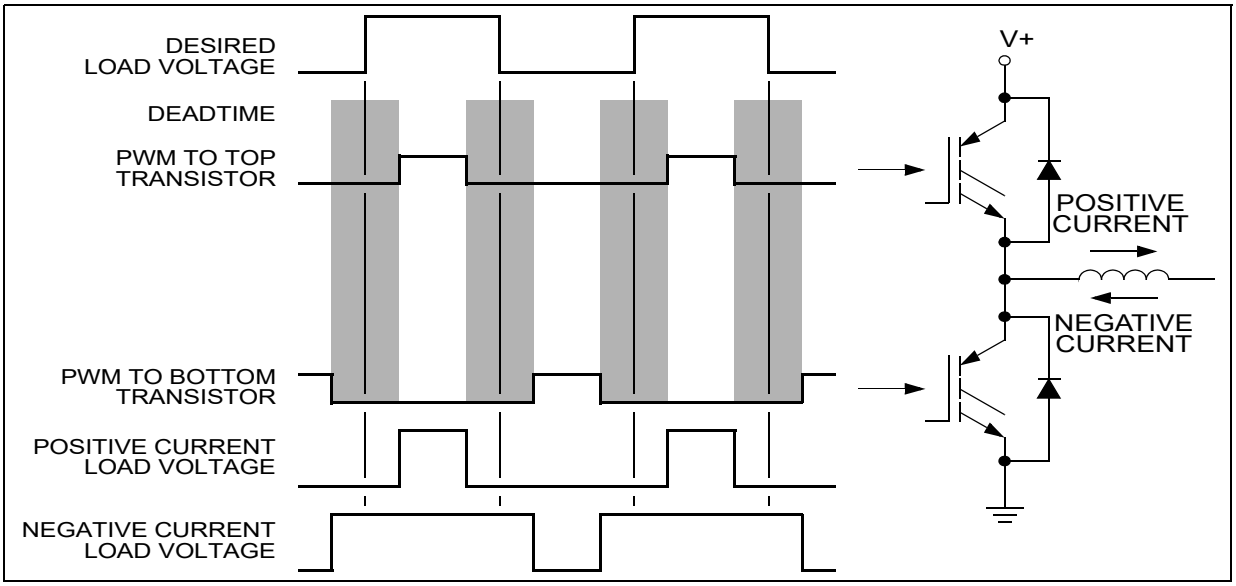
Figure 483. Deadtime Insertion



36.4.3.8.1 Top/Bottom Correction

In complementary mode, either the top or the bottom transistor controls the output voltage. However, deadtime has to be inserted to avoid overlap of conducting interval between the top and bottom transistor. Both transistors in complementary mode are off during deadtime, allowing the output voltage to be determined by the current status of load and introduce distortion in the output voltage. See *Figure 484*. On AC induction motors running open-loop, the distortion typically manifests itself as poor low-speed performance, such as torque ripple and rough operation.

Figure 484. Deadtime Distortion



During deadtime, load inductance distorts output voltage by keeping current flowing through the diodes. This deadtime current flow creates a load voltage that varies with current direction. With a positive current flow, the load voltage during deadtime is equal to the bottom supply, putting the top transistor in control. With a negative current flow, the load voltage during deadtime is equal to the top supply putting the bottom transistor in control.

Remembering that the original PWM pulse widths where shortened by deadtime insertion, the averaged sinusoidal output will be less than the desired value. However, when deadtime is inserted, it creates a distortion in the motor current waveform. This distortion is aggravated by dissimilar turn-on and turn-off delays of each of the transistors. By giving the PWM module information on which transistor is controlling at a given time this distortion can be corrected.

For a typical circuit in complementary channel operation, only one of the transistors will be effective in controlling the output voltage at any given time. This depends on the direction of the motor current for that pair. See *Figure 484*. To correct distortion one of two different factors must be added to the desired PWM value, depending on whether the top or bottom transistor is controlling the output voltage. Therefore, the software is responsible for calculating both compensated PWM values prior to placing them in the VALx registers. Either the VAL2/VAL3 or the VAL4/VAL5 register pair controls the pulse width at any given time. For a given PWM pair, whether the VAL2/VAL3 or VAL4/VAL5 pair is active depends on either:

- The state of the current status pin, PWMx, for that driver.
- The state of the odd/even correction bit, IPOL, for that driver.

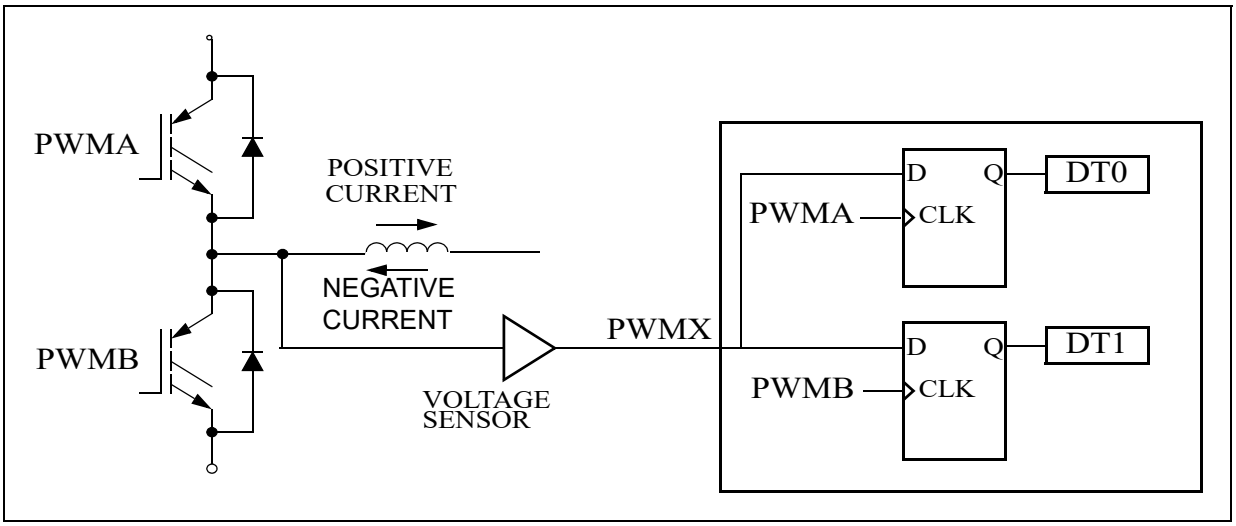
To correct deadtime distortion, software can decrease or increase the value in the appropriate VALx register.

- In edge-aligned operation, decreasing or increasing the PWM value by a correction value equal to the deadtime typically compensates for deadtime distortion.
- In center-aligned operation, decreasing or increasing the PWM value by a correction value equal to one-half the deadtime typically compensates for deadtime distortion.

36.4.3.8.2 Manual Correction

To detect the current status, the voltage on each PWMx pin is sampled twice in a PWM period, at the end of each deadtime. The value is stored in the DTx bits in the CTRL1 register. The DTx bits are a timing marker especially indicating when to toggle between PWM value registers. Software can then set the IPOL bit to switch between VAL2/VAL3 and VAL4/VAL5 register pairs according to DTx values.

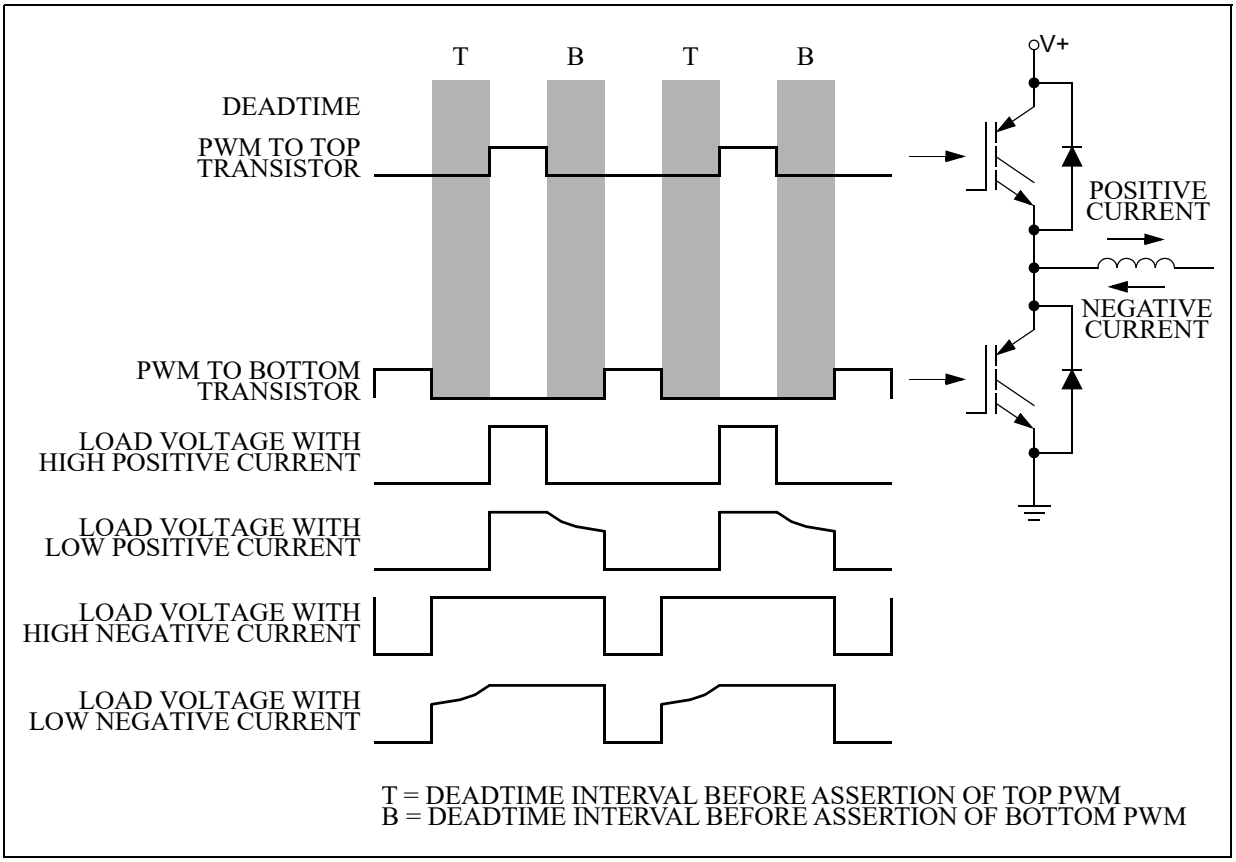
Figure 485. Current-status Sense Scheme for Deadtime Correction



Both D flip-flops latch low, DT0 = 0, DT1 = 0, during deadtime periods if current is large and flowing out of the complementary circuit. See *Figure 485*. Both D flip-flops latch the high, DT0 = 1, DT1 = 1, during deadtime periods if current is also large and flowing into the complementary circuit.

However, under low-current, the output voltage of the complementary circuit during deadtime is somewhere between the high and low levels. The current cannot free-wheel through the opposition anti-body diode, regardless of polarity, giving additional distortion when the current crosses zero. Sampled results will be DT0 = 0 and DT1 = 1. Thus, the best time to change one PWM value register to another is just before the current zero crossing.

Figure 486. Output Voltage Waveforms

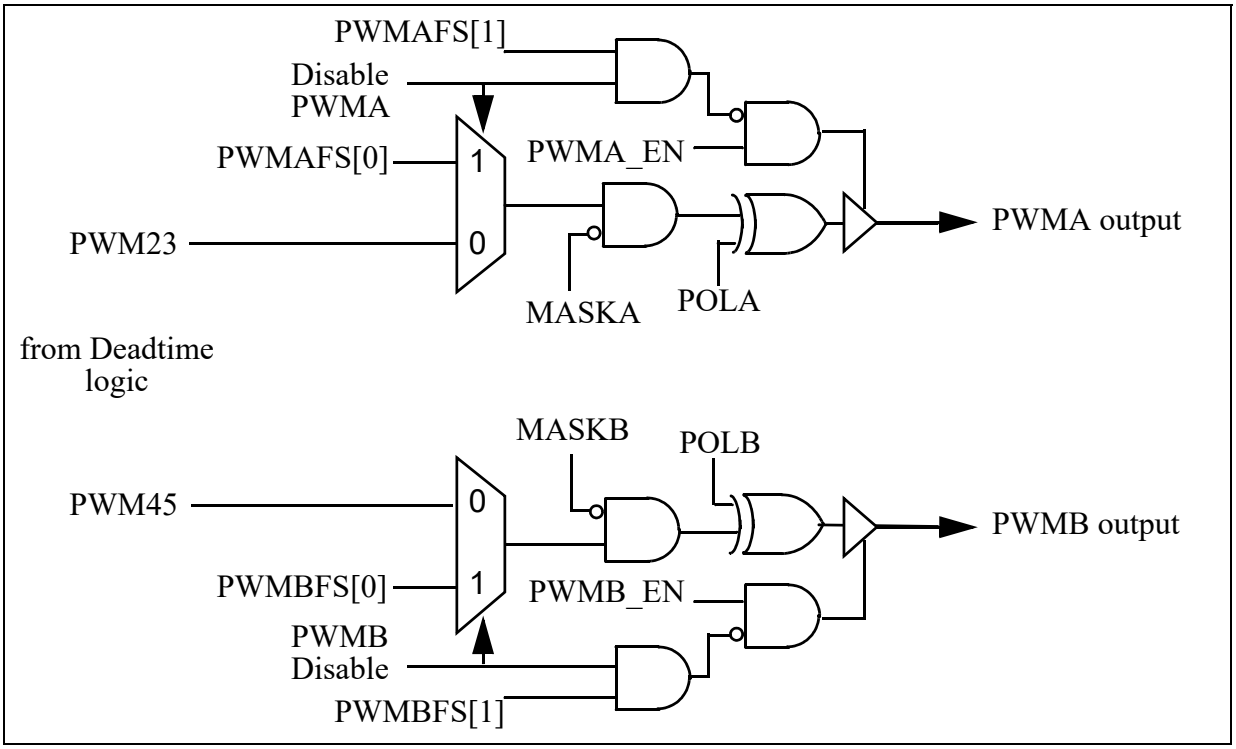


36.4.3.9 Output Logic

Figure 487 shows the output logic of each submodule including how each PWM output has individual fault disabling, polarity control, and output enable. This allows for maximum flexibility when interfacing to the external circuitry.

The PWM23 and PWM45 signals which are output from the deadtime logic in *Figure 487* are positive true signals. In other words, a high level on these signals should result in the corresponding transistor in the PWM inverter being turned ON. The voltage level required at the PWM output pin to turn the transistor ON or OFF is a function of the logic between the pin and the transistor. Therefore, it is imperative that the user program the POLA and POLB bits before enabling the output pins. A fault condition can result in the PWM output being tristated, forced to a logic 1, or forced to a logic 0 depending on the values programmed into the PWMxFS fields.

Figure 487. Output Logic Section

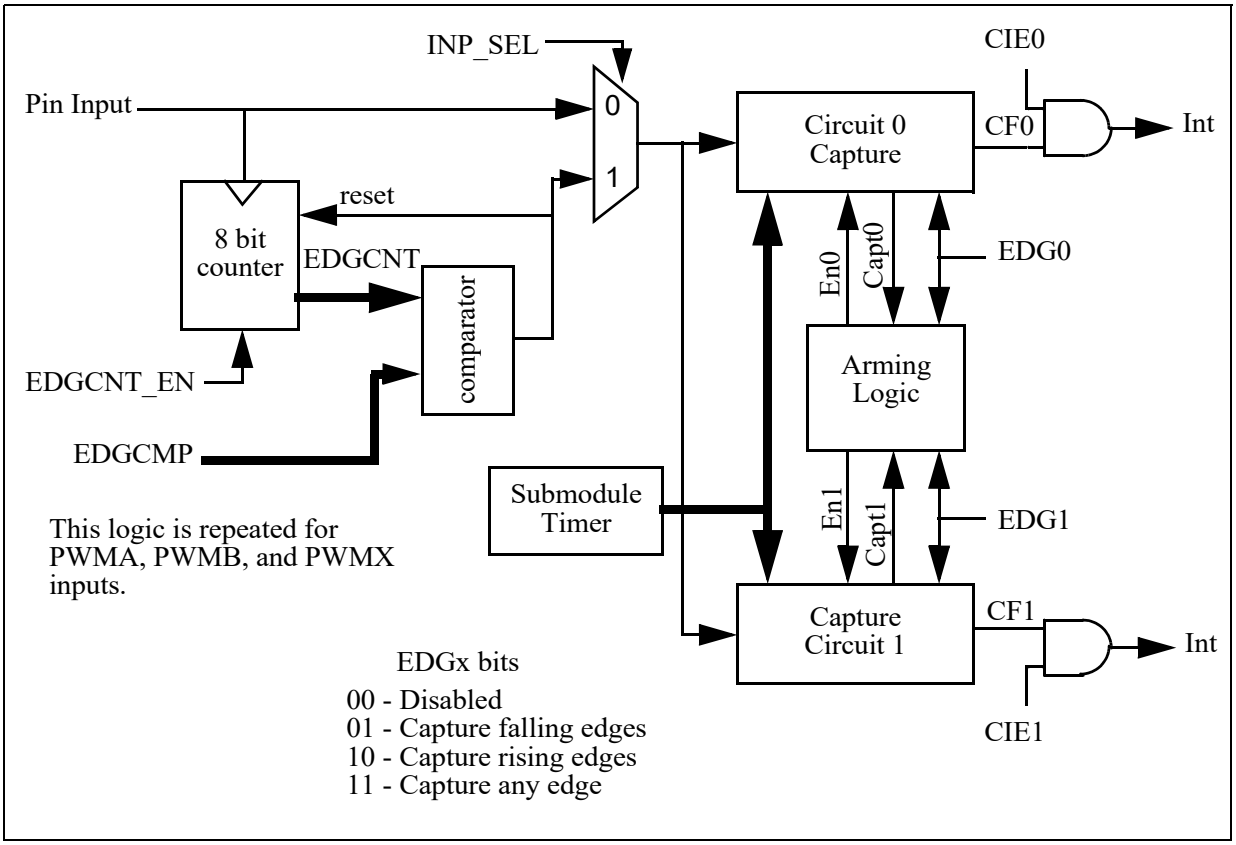


36.4.3.10 E-Capture

Commensurate with the idea of controlling both edges of an output signal, the Enhanced Capture (E-Capture) logic is designed to measure both edges of an input signal. As a result, when a submodule pin is configured for input capture, the CVALx registers associated with that pin are used to record the edge values.

Figure 488 illustrates the block diagram of the E-Capture circuit. Upon entering the pin input, the signal is split into two paths. One goes straight to a mux input where software can select to pass the signal directly to the capture logic for processing. The other path connects the signal to an 8 bit counter which counts both the rising and falling edges of the signal. The output of this counter is compared to an 8 bit value that is specified by the user (EDGCMPx) and when the two values are equal, the comparator generates a pulse that resets the counter. This pulse is also supplied to the mux input where software can select it to be processed by the capture logic. This feature permits the E-Capture circuit to count up to 256 edge events before initiating a capture event. this feature is useful for dividing down high frequency signals for capture processing so that capture interrupts don't overwhelm the CPU. Also, this feature can be used to simply generate an interrupt after "n" events have been counted.

Figure 488. Enhanced Capture (E-Capture) Logic



Based on the mode selection, the mux selects either the pin input or the compare output from the count/compare circuit to be processed by the capture logic. The selected signal is routed to two separate capture circuits which work in tandem to capture sequential edges of the signal. The type of edge to be captured by each circuit is determined by the EDGx1 and EDGx0 bits whose functionality is listed in *Figure 488*. Also, controlling the operation of the capture circuits is the arming logic which allows captures to be performed in a free running (continuous) or one shot fashion. In free running mode, the capture sequences will be performed indefinitely. If both capture circuits are enabled, they will work together in a ping-pong style where a capture event from one circuit leads to the arming of the other and vice versa. In one shot mode, only one capture sequence will be performed. If both capture circuits are enabled, capture circuit 0 is first armed and when a capture event occurs, capture circuit 1 is armed. Once the second capture occurs, further captures are disabled until another capture sequence is initiated. Both capture circuits are also capable of generating an interrupt to the CPU.

36.4.3.11 Fault Protection

Fault protection can control any combination of PWM output pins. Faults are generated by a logic one on any of the FAULTx pins. This polarity can be changed via the FLVL bits. Each FAULTx pin can be mapped arbitrarily to any of the PWM outputs. When fault protection hardware disables PWM outputs, the PWM generator continues to run, only the output pins are forced to logic 0, logic 1, or tristated depending the values of the PWMxFS bits.

The fault decoder disables PWM pins selected by the fault logic and the disable mapping register (DISMAP). See *Figure 489* for an example of the fault disable logic. Each bank of bits in DISMAP control the mapping for a single PWM pin. Refer to *Table 566*.

The fault protection is enabled even when the PWM module is not enabled; therefore, a fault will be latched in and must be cleared in order to prevent an interrupt when the PWM is enabled.

Figure 489. Fault Decoder for PWMA

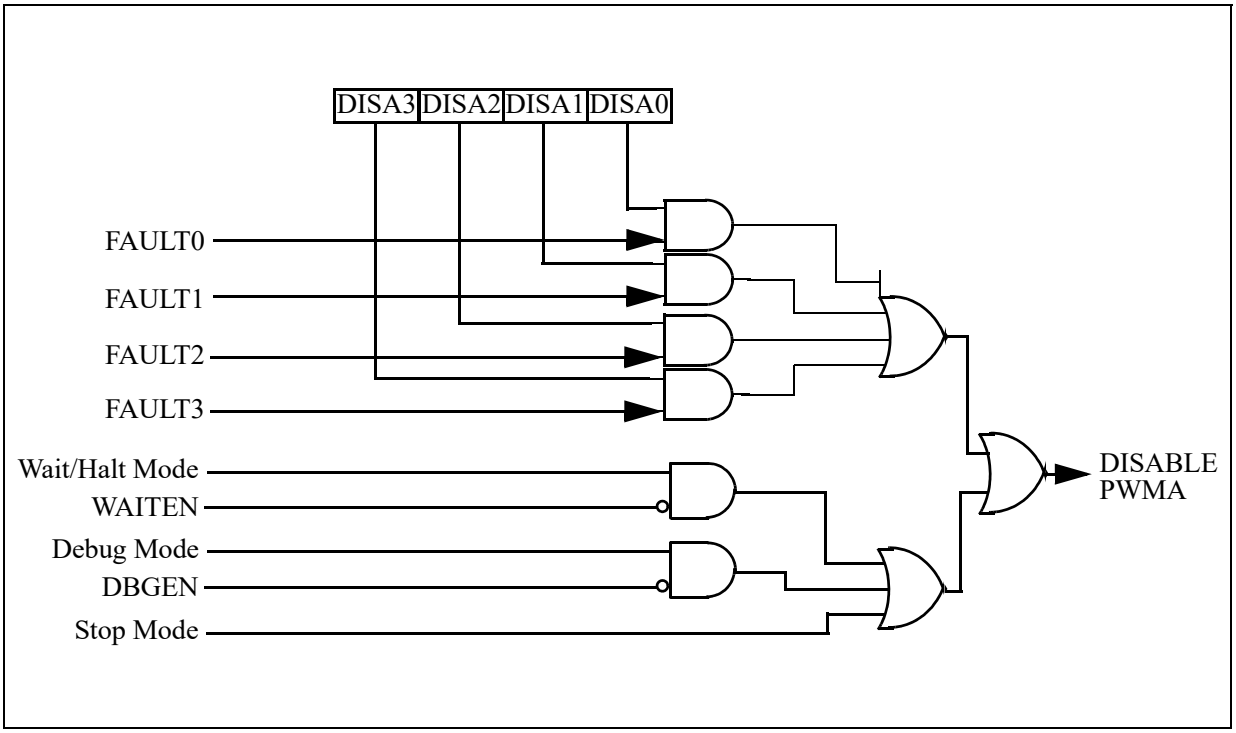


Table 566. Fault Mapping

PWM Pin	Controlling Register Bits
PWMA	DISA[3:0]
PWMB	DISB[3:0]
PWMX	DISX[3:0]

36.4.3.11.1 Fault Pin Filter

Each fault pin has a programmable filter that can be bypassed. The sampling period of the filter can be adjusted with the FILT_PER field of the FFILTx register. The number of consecutive samples that must agree before an input transition is recognized can be adjusted using the FILT_CNT field of the same register. Setting FILT_PER to all 0 disables the input filter for a given FAULTx pin.

Upon detecting a logic 0 on the filtered FAULTx pin (or a logic 1 if FLVLx is set), the corresponding FFPINx and fault flag, FFLAGx, bits are set. The FFPINx bit remains set as long as the filtered FAULTx pin is zero. Clear FFLAGx by writing a logic 1 to FFLAGx.

If the FIEx, FAULTx pin interrupt enable bit is set, the FFLAGx flag generates a CPU interrupt request. The interrupt request latch remains set until:

- Software clears the FFLAGx flag by writing a logic one to the bit.
- Software clears the FIEx bit by writing a logic zero to it.
- A reset occurs.

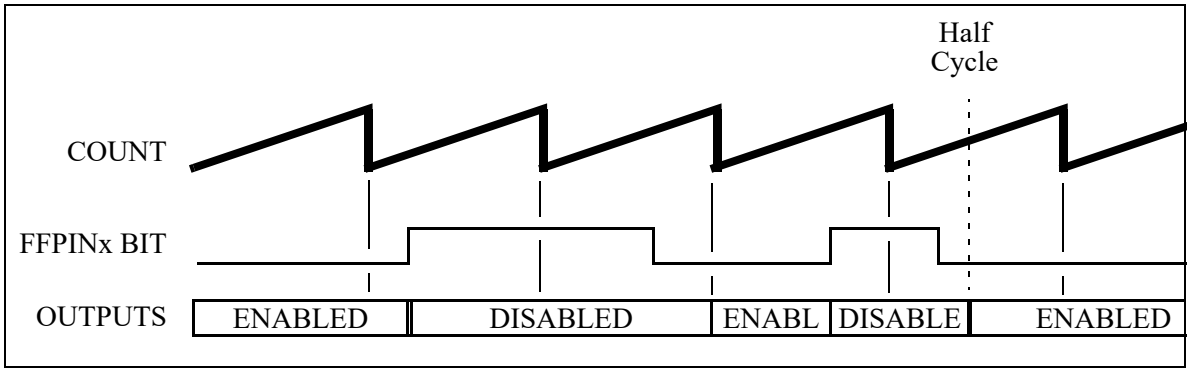
Even with the filter enabled, there is a combinational path from the FAULTx inputs to the PWM pins. This logic is also capable of holding a fault condition in the event of loss of clock to the PWM module.

36.4.3.11.2 Automatic Fault Clearing

Setting an automatic clearing mode bit, FAUTOx, configures faults from the FAULTx pin for automatic clearing.

When FAUTOx is set, disabled PWM pins are enabled when the FAULTx pin returns to logic one and a new PWM full or half cycle begins. See *Figure 490*. If FFULLx is set, then the disabled PWM pins are enabled only at the start of a full cycle and not at the half cycle. Clearing the FFLAGx flag does not affect disabled PWM pins when FAUTOx is set.

Figure 490. Automatic Fault Clearing



36.4.3.11.3 Manual Fault Clearing

Clearing the automatic clearing mode bit, FAUTOx, configures faults from the FAULTx pin for manual clearing:

- If the fault safety mode bits, FSAFEx, are clear, then PWM pins disabled by the FAULTx pins are enabled when:
  - Software clears the corresponding FFLAGx flag.
  - The pins are enabled when the next PWM full or half cycle begins regardless of the logic level detected by the filter at the FAULTx pin. See *Figure 491*. If FFULLx is set, then the disabled PWM pins are enabled only at the start of a full cycle and not at the half cycle.
- If the fault safety mode bits, FSAFEx, are set, then PWM pins disabled by the FAULTx pins are enabled when:
  - Software clears the corresponding FFLAGx flag.
  - The filter detects a logic one on the FAULTx pin at the start of the next PWM full or half cycle boundary. See *Figure 492*. If FFULLx is set, then the disabled PWM pins are enabled only at the start of a full cycle and not at the half cycle.

Figure 491. Manual Fault Clearing (FSAFE=0)

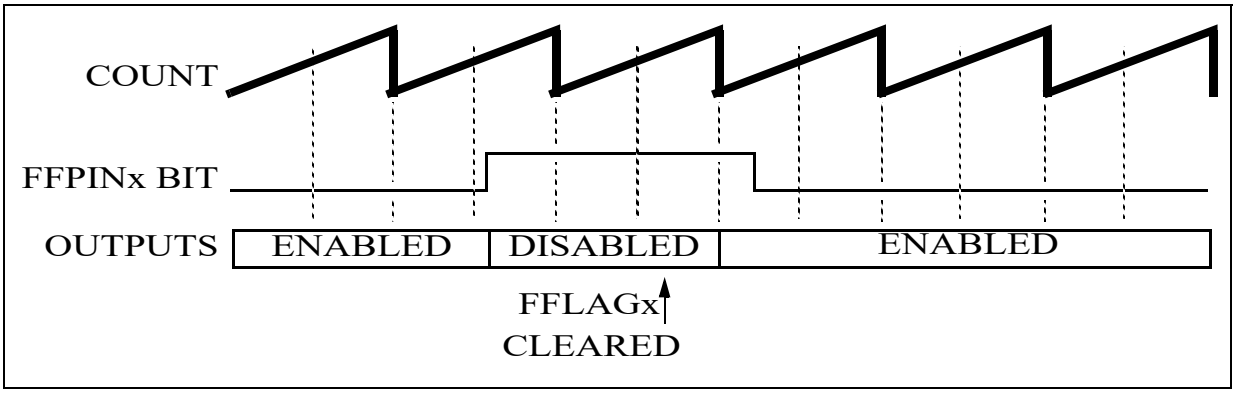
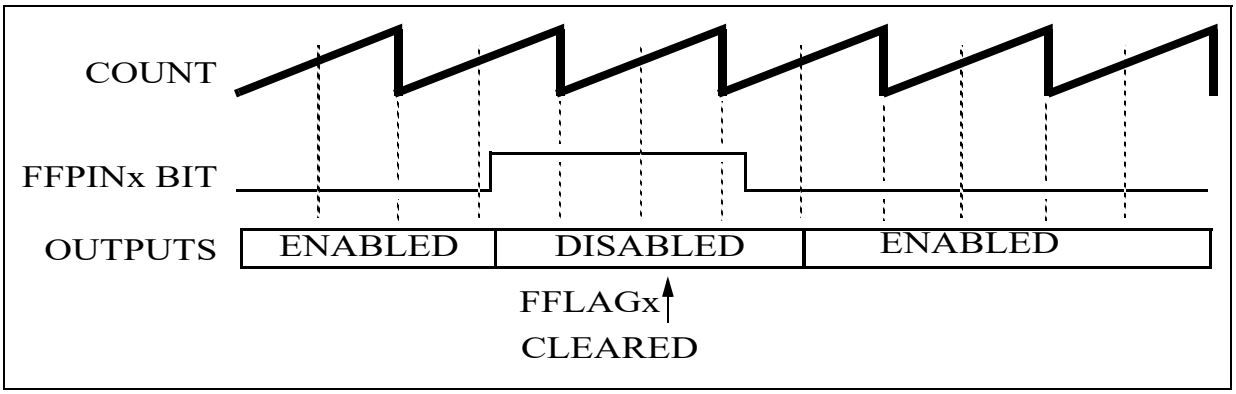


Figure 492. Manual Fault Clearing (FSAFE=1)



Note: Fault protection also applies during software output control when the SEL23 and SEL45 fields are set to select OUT23 and OUT45 bits or EXTA and EXTB. Fault clearing still occurs at half PWM cycle boundaries while the PWM generator is engaged, RUN equals one. But the OUTx bits can control the PWM pins while the PWM generator is off, RUN equals zero. Thus, fault clearing occurs at IPBus cycles while the PWM generator is off and at the start of PWM cycles when the generator is engaged.

36.4.3.11.4 Fault Testing

The FTEST bit is used to simulate a fault condition on each of the fault inputs.

36.4.4 PWM Generator Loading

36.4.4.1 Load Enable

The LDOK bit enables loading of the following PWM generator parameters:

- The prescaler divisor: from the PRSC bits in the CTRL1 register.
- The PWM period and pulse width: from the INIT and VALx registers.

LDOK allows software to finish calculating all of these PWM parameters so they can be synchronously updated. The PSRC, INIT, and VALx registers are loaded by software into a set of outer buffers. When LDOK is set, these values are transferred to an inner set of registers at the beginning of the next PWM reload cycle to be used by the PWM generator. These values can be transfered to the inner set of registers immediately upon setting LDOK

if LDMOD is set. Set LDOK by reading it when it is a logic zero and then writing a logic one to it. After loading, LDOK is automatically cleared.

36.4.4.2 Load Frequency

The LDFQ bits in the CTRL1 register select an integral loading frequency of one to 16 PWM reload opportunities. The LDFQ bits take effect at every PWM reload opportunity, regardless the state of the LDOK bit. The HALF and FULL bits in the CTRL1 register control reload timing. If FULL is set, a reload opportunity occurs at the end of every PWM cycle when the count equals VAL1. If HALF is set, a reload opportunity occurs at the half cycle when the count equals VAL0. If both HALF and FULL are set, a reload opportunity occurs twice per PWM cycle when the count equals VAL1 and when it equals VAL0.

Figure 493. Full Cycle Reload Frequency Change

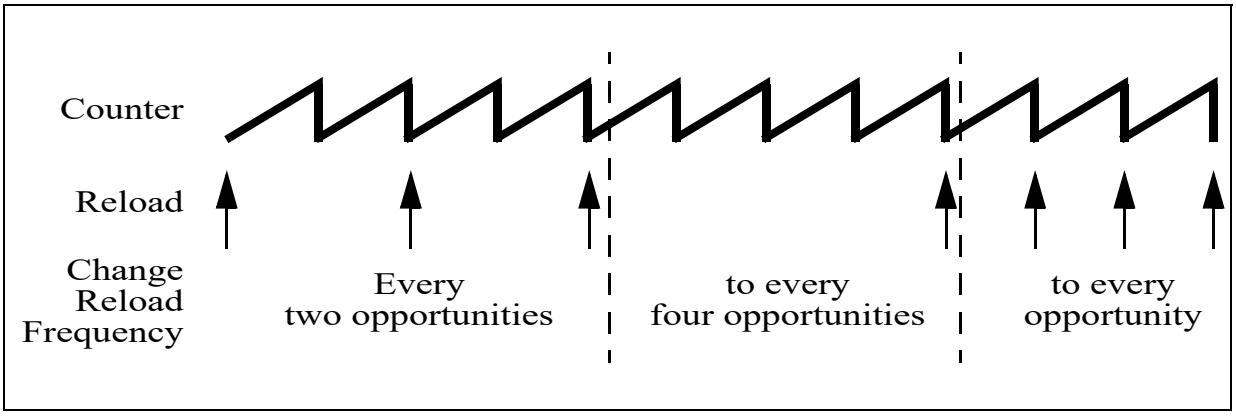


Figure 494. Half Cycle Reload Frequency Change

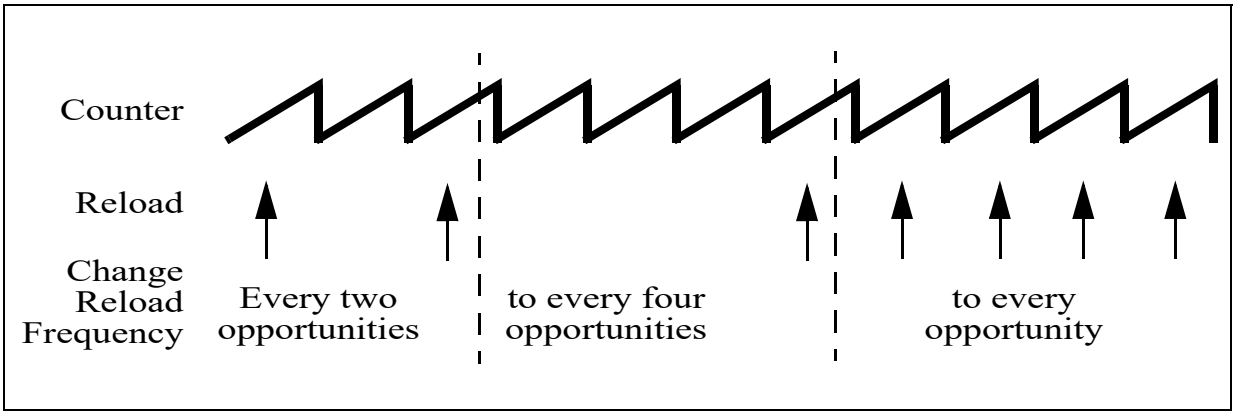
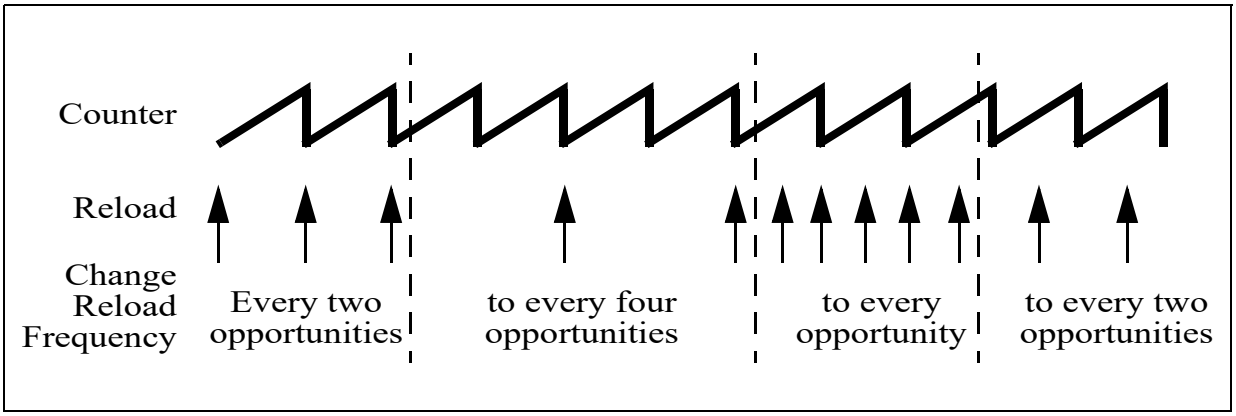


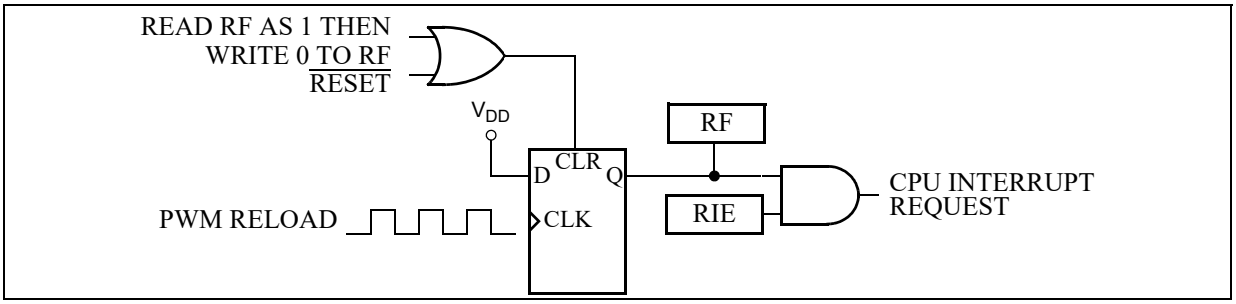
Figure 495. Full and Half Cycle Reload Frequency Change



36.4.4.3 Reload Flag

At every reload opportunity the PWM Reload Flag (RF) in the CTRL1 register is set. Setting RF happens even if an actual reload is prevented by the LDOK bit. If the PWM reload interrupt enable bit, RIE is set, the RF flag generates CPU interrupt requests allowing software to calculate new PWM parameters in real time. When RIE is not set, reloads still occur at the selected reload rate without generating CPU interrupt requests.

Figure 496. PWMF Reload Interrupt Request



36.4.4.4 Reload Errors

Whenever one of the VAL or PSRC registers is updated, the RUF flag is set to indicate that the data is not coherent. RUF will be cleared by a successful reload which consists of the reload signal while LDOK is set. If RUF is set and LDOK is clear when the reload signal occurs, a reload error has taken place and the REF bit is set. If RUF is clear when a reload signal asserts, then the data is coherent and no error will be flagged.

36.4.4.5 Initialization

Initialize all registers and set the LDOK bit before setting the RUN bit.

Note: Even if LDOK is not set, setting RUN also sets the RF flag. To prevent a CPU interrupt request, clear the RIE bit before setting RUN.

The PWM generator uses the last values loaded if RUN is cleared and then set while LDOK equals zero.

When the RUN bit is cleared:

- The RF flag and pending CPU interrupt requests are not cleared.
- All fault circuitry remains active.
- Software/external output control remains active.
- Deadtime insertion continues during software/external output control.

36.5 Resets

The mcPWM module can only be reset by the IPG_HARD_ASYNC_RESET_B signal. This forces all registers to their reset states and tri-states the PWM outputs.

36.6 Clocks

Each PWM submodule receives it's own copy of the IPBus Clock. This allows for better power management by turning off the clock to unused submodules. The fault logic also has its own version of the IPBus clock.

Table 567. Clock Summary

Clock	Used by
ipg_clk_sub0	Submodule 0
ipg_clk_sub1	Submodule 1
ipg_clk_sub2	Submodule 2
ipg_clk_sub3	Submodule 3
ipg_clk_flt	Fault logic

36.7 Interrupts

Each of the submodules within the mcPWM can generate an interrupt from several sources. The fault logic can also generate interrupts. The interrupt service routine (ISR) must check the related interrupt enables and interrupt flags to determine the actual cause of the interrupt.

Table 568. Interrupt Summary

Core Interrupt	Interrupt Flag	Interrupt Enable	Name	Description
ipi_int_comp0	CMPF_0	CMPIE_0	Submodule 0 compare interrupt	Compare event has occurred
ipi_int_capt0	CFA1_0, CFA0_0, CFB1_0, CFB0_0, CFX1_0, CFX0_0	CFA1IE_0, CFA0IE_0, CFB1IE_0, CFB0IE_0, CFX1IE_0, CFX0IE_0	Submodule 0 input capture interrupt	Input capture event has occurred

Table 568. Interrupt Summary (continued)

Core Interrupt	Interrupt Flag	Interrupt Enable	Name	Description
ipi_int_reload0	RF_0	RIE_0	Submodule 0 reload interrupt	Reload event has occurred
ipi_int_comp1	CMPF_1	CMPIE_1	Submodule 1 compare interrupt	Compare event has occurred
ipi_int_capt1	CFA1_1, CFA0_1, CFB1_1, CFB0_1, CFX1_1, CFX0_1	CFA1IE_1, CFA0IE_1, CFB1IE_1, CFB0IE_1, CFX1IE_1, CFX0IE_1	Submodule 1 input capture interrupt	Input capture event has occurred
ipi_int_reload1	RF_1	RIE_1	Submodule 1 reload interrupt	Reload event has occurred
ipi_int_comp2	CMPF_2	CMPIE_2	Submodule 2 compare interrupt	Compare event has occurred
ipi_int_capt2	CFA1_2, CFA0_2, CFB1_2, CFB0_2, CFX1_2, CFX0_2	CFA1IE_2, CFA0IE_2, CFB1IE_2, CFB0IE_2, CFX1IE_2, CFX0IE_2	Submodule 2 input capture interrupt	Input capture event has occurred
ipi_int_reload2	RF_2	RIE_2	Submodule 2 reload interrupt	Reload event has occurred
ipi_int_comp3	CMPF_3	CMPIE_3	Submodule 3 compare interrupt	Compare event has occurred
ipi_int_capt3	CFA1_3, CFA0_3, CFB1_3, CFB0_3, CFX1_3, CFX0_3	CFA1IE_3, CFA0IE_3, CFB1IE_3, CFB0IE_3, CFX1IE_3, CFX0IE_3	Submodule 3 input capture interrupt	Input capture event has occurred
ipi_int_reload3	RF_3	RIE_3	Submodule 3 reload interrupt	Reload event has occurred
ipi_int_rerr	REF_0	REIE_0	Submodule 0 reload error interrupt	Reload error has occurred
	REF_1	REIE_1	Submodule 1 reload error interrupt	
	REF_2	REIE_2	Submodule 2 reload error interrupt	
	REF_3	REIE_3	Submodule 3 reload error interrupt	
ipi_int_fault	FFLAG	FIE	Fault input interrupt	Fault condition has been detected

36.8 DMA

Each submodule can request a DMA read access for its capture FIFOs and a DMA write request for its double buffered VALx registers.

Table 569. DMA Summary

DMA Request	DMA Enable	Name	Description
Submodule 0 read request	CX0DE_0	Capture FIFO X0 read request	CVAL0 contains a value to be read
	CX1DE_0	Capture FIFO X1 read request	CVAL1 contains a value to be read
	CA0DE_0	Capture FIFO A0 read request	CVAL2 contains a value to be read
	CA1DE_0	Capture FIFO A1 read request	CVAL3 contains a value to be read
	CB0DE_0	Capture FIFO B0 read request	CVAL4 contains a value to be read
	CB1DE_0	Capture FIFO B1 read request	CVAL5 contains a value to be read
	CAPTDE_0	Capture FIFO read request source select	Selects source of read DMA request
Submodule 0 write request	VALDE_0	VALx write request	VALx registers need to be updated
Submodule 1 read request	CX0DE_1	Capture FIFO X0 read request	CVAL0 contains a value to be read
	CX1DE_1	Capture FIFO X1 read request	CVAL1 contains a value to be read
	CA0DE_1	Capture FIFO A0 read request	CVAL2 contains a value to be read
	CA1DE_1	Capture FIFO A1 read request	CVAL3 contains a value to be read
	CB0DE_1	Capture FIFO B0 read request	CVAL4 contains a value to be read
	CB1DE_1	Capture FIFO B1 read request	CVAL5 contains a value to be read
	CAPTDE_1	Capture FIFO read request source select	Selects source of read DMA request
Submodule 1 write request	VALDE_1	VALx write request	VALx registers need to be updated
Submodule 2 read request	CX0DE_2	Capture FIFO X0 read request	CVAL0 contains a value to be read
	CX1DE_2	Capture FIFO X1 read request	CVAL1 contains a value to be read
	CA0DE_2	Capture FIFO A0 read request	CVAL2 contains a value to be read
	CA1DE_2	Capture FIFO A1 read request	CVAL3 contains a value to be read
	CB0DE_2	Capture FIFO B0 read request	CVAL4 contains a value to be read
	CB1DE_2	Capture FIFO B1 read request	CVAL5 contains a value to be read
	CAPTDE_2	Capture FIFO read request source select	Selects source of read DMA request
Submodule 2 write request	VALDE_2	VALx write request	VALx registers need to be updated

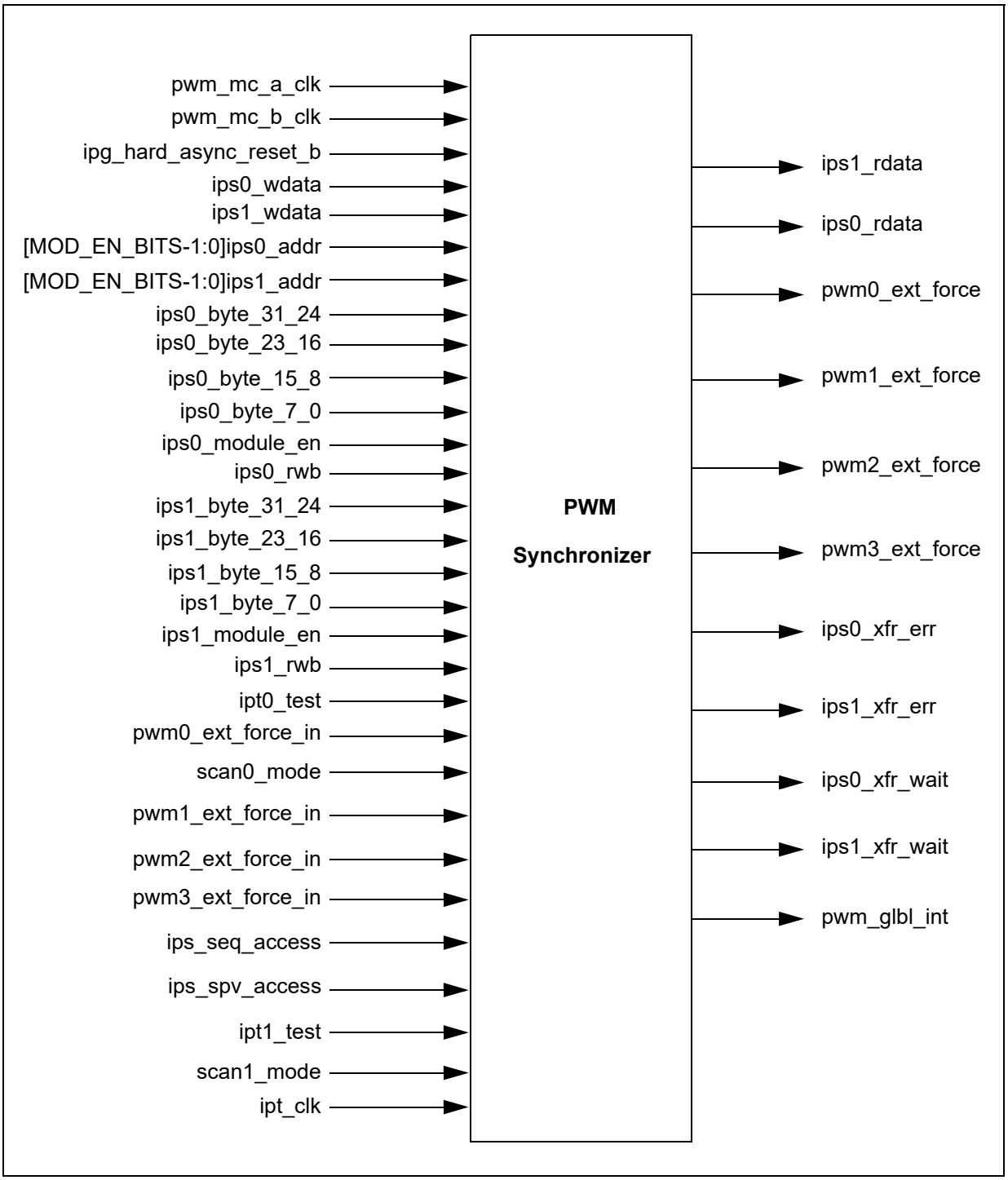
Table 569. DMA Summary (continued)

DMA Request	DMA Enable	Name	Description
Submodule 3 read request	CX0DE_3	Capture FIFO X0 read request	CVAL0 contains a value to be read
	CX1DE_3	Capture FIFO X1 read request	CVAL1 contains a value to be read
	CA0DE_3	Capture FIFO A0 read request	CVAL2 contains a value to be read
	CA1DE_3	Capture FIFO A1 read request	CVAL3 contains a value to be read
	CB0DE_3	Capture FIFO B0 read request	CVAL4 contains a value to be read
	CB1DE_3	Capture FIFO B1 read request	CVAL5 contains a value to be read
	CAPTDE_3	Capture FIFO read request source select	Selects source of read DMA request
Submodule 3 write request	VALDE_3	VALx write request	VALx registers need to be updated

37 PWM Synchronizer

37.1 Introduction

Figure 497. Block Diagram



37.2 Main Features

- Start synchronously the triggers for PWM0, PWM1, PWM2, PWM3 in full sync option.
- Start synchronously triggers for PWM0, PWM1 and PWM2, PWM3 independently in the half sync option.
- Two independent interfaces (PBRIDGE_0, PBRIDGE_1).
- Allows Read back of the 4 PWM triggers (to check if triggers are sent or not).
- Clocked by the 2 motor control clocks.
- Four bits wide pre-scaler for the two clocks.
- 12 bits delay counters (max delay (34 µs to 546 µs)).
- One global IRQ interrupt to notify the start of either of PWMx trigger and misconfiguration error.
- All Registers are byte-wise, half-word and word writable.

37.3 Memory map and register description

This section provides a detailed description of all registers accessible in the PWM synchronizer.

37.3.1 Memory map

Table 570 gives an overview of the implemented PWM synchronizer registers.

Table 570. PWM synchronizer memory map

Address offset	Register	Location
PBRIDGE_0		
0x0000	PWM Synchronizer trigger readback 0 register (PWM_SYNCHR_TRIG_READBACK0)	*Section 37.3.2.1*
0x0004	PWM Synchronizer delay control 0 (PWM_SYNCHR_DLY_CNTR0)	*Section 37.3.2.2*
0x0008	PWM Synchronizer start 0 register (PWM_SYNCHR_STRT0)	*Section 37.3.2.3*
0x000C	PWM Synchronizer configuration 0 register (PWM_SYNCHR_CONFIG0)	*Section 37.3.2.4*
0x0010	PWM Synchronizer enable 0 register (PWM_SYNCHR_INT_EN0)	*Section 37.3.2.5*
0x0014	PWM Synchronizer prescalar 0 register(PWM_SYNCHR_PRE_SCALAR0)	*Section 37.3.2.6*
0x0018	PWM Synchronizer interrupt status 0 register (PWM_SYNCHR_INT_STAT0)	*Section 37.3.2.7*
PBRIDGE_1		
0x0000	PWM Synchronizer trigger readback 1 register (PWM_SYNCHR_TRIG_READBACK1)	*Section 37.3.2.8*
0x0004	PWM Synchronizer delay control 1 (PWM_SYNCHR_DLY_CNTR1)	*Section 37.3.2.9*
0x0008	PWM Synchronizer start 1 register (PWM_SYNCHR_STRT1)	*Section 37.3.2.10*
0x000C	PWM Synchronizer configuration 1 register (PWM_SYNCHR_CONFIG1)	*Section 37.3.2.11*
0x0010	PWM Synchronizer enable 1 register (PWM_SYNCHR_INT_EN1)	*Section 37.3.2.12*

Table 570. PWM synchronizer memory map (continued)

Address offset	Register	Location
0x0014	PWM Synchronizer prescalar 1 register(PWM_SYNCHR_PRE_SCALAR1)	*Section 37.3.2.13*
0x0018	PWM Synchronizer interrupt status 1 register (PWM_SYNCHR_INT_STAT1)	*Section 37.3.2.14*

37.3.2 Register descriptions

37.3.2.1 PWM_SYNCHR_TRIG_READBACK0

pwmx_ext_force is to be shorted with pwmx_ext_force_in.

Offset: 0x0000 Access: read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	PWM1_EXT_FORCE_IN	PWM0_EXT_FORCE_IN
W															w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 498. PWM_SYNCHR_TRIG_READBACK0

Table 571. PWM_SYNCHR_TRIG_READBACK0 field descriptions

Field	Description
30 PWM1_EXT_FORCE_IN	0 PWM1_EXT_FORCE trigger output pulse was not generated 1 PWM1_EXT_FORCE trigger output pulse was generated
31 PWM0_EXT_FORCE_IN	0 PWM0_EXT_FORCE trigger output pulse was not generated 1 PWM0_EXT_FORCE trigger output pulse was generated

37.3.2.2 PWM_SYNCHR_DLY_CNTR0

This register works on PBRIDGE_0.

Offset: 0x0004 Access: read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DLY_CNT_PWM1												0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	DLY_CNT_PWM0												0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 499. PWM_SYNCHR_DLY_CNTR0

Table 572. PWM_SYNCHR_DLY_CNTR0 field descriptions

Field	Description
0:11 DLY_CNT_PWM1	Start delay value to trigger PWM1 in number of prescaled clock cycles - 1. Note that predicting exact delay not possible as registers work in motor clock domain and trigger generating logic works in another domain(divided clock).
16:27 DLY_CNT_PWM0	Start delay value to trigger PWM0 in number of prescaled clock cycles - 1. Note that predicting exact delay not possible as registers work in motor clock domain and trigger generating logic works in another domain (divided clock).

37.3.2.3 PWM_SYNCHR_STRT0

This register is written by PBRIDGE_0 only.

Offset: 0x0008 Access: write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																FH0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 500. PWM_SYNCHR_STRT0

Table 573. PWM_SYNCHR_STRT0 field descriptions

Field	Description
31 FH0	0 No action for write 0 1 Half/Full sync mode start These bits are write once and read zero always. The full sync mode starts after both the FH0 and FH1 have been written.

37.3.2.4 PWM_SYNCHR_CONFIG0

This register is written by PBRIDGE_0 only.

Offset: 0x000C Access: read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	HALF_FULL0	0	PWM10_EN	
W														SR0		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 501. PWM_SYNCHR_CONFIG0

Table 574. PWM_SYNCHR_CONFIG0 field descriptions

Field	Description
28 HALF_FULL0	0 Half sync mode selected 1 Full sync mode selected
29 SR0	1 Soft reset all the PBRIDGE_0 status logic registers, internal counters and outputs. 0 No effect This bit is write once, its always reads as zero.
30:31 PWM10_EN	00 PWM0 disabled, PWM1 disabled 01 PWM0 enabled, PWM1 disabled 10 PWM1 enabled, PWM0 disabled 11 PWM1 enabled, PWM0 enabled

37.3.2.5 PWM_SYNCHR_INT_EN0

This register is written by PBRIDGE_0 only.

Offset: 0x0010 Access: read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	ERR_INT_EN	INT_EN1	INT_EN0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 502. PWM_SYNCHR_INT_EN0

Table 575. PWM_SYNCHR_INT_EN0 field descriptions

Field	Description
29 ERR_INT_EN	1 Enable global interrupt generation for misconfiguration error occurrence 0 Disable global interrupt generation for misconfiguration error occurrence
30 INT_EN1	1 Enables the interrupt generation for PWM1 0 Disables the interrupt generation for PWM1
31 INT_EN0	1 Enables the interrupt generation for PWM0 0 Disables the interrupt generation for PWM0

37.3.2.6 PWM_SYNCHR_PRE_SCALAR0

This register is written by PBRIDGE_0 only.

Offset: 0x0014 Access: read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	PRE_SCALE0			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 503. PWM_SYNCHR_PRE_SCALAR0

Table 576. PWM_SYNCHR_PRE_SCALAR0 field descriptions

Field	Description
28:31 PRE_SCALE0	PRE_SCALE0 + 1 gives the clock division factor This value should be the same as PRE_SCALE1 in full sync mode

37.3.2.7 PWM_SYNCHR_INT_STAT0

This register is written by PBRIDGE_0 only.

Offset: 0x0018 Access: read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	ERR0	INT1	INT0
W														w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 504. PWM_SYNCHR_INT_STAT0

Table 577. PWM_SYNCHR_INT_STAT0 field descriptions

Field	Description
29 ERR0	0 No Configuration mismatch between PBRIDGE_0 and PBRIDGE_1 registers 1 Configuration mismatch seen between PBRIDGE_0 and PBRIDGE_1 registers Configuration Registers compared are PWM_SYNCHR_PRESCALE0 and PWM_SYNCHR_PRESCALE1, HALF_FULL0 and HALF_FULL1 bits of PWM_SYNCHR_CONFIGx registers (Comparison is done when the trigger start bits, FH0 or FH1, are written)
30 INT1	0 No interrupt was generated. 1 PWM1 interrupt was generated.
31 INT0	0 No interrupt was generated. 1 PWM0 interrupt was generated.

37.3.2.8 PWM_SYNCHR_TRIG_READBACK1

pwmx_ext_force is to be shorted with pwmx_ext_force_in.

Offset: 0x0000 Access: read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	PWM3_EXT_FORCE_IN	PWM2_EXT_FORCE_IN
W															w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 505. PWM_SYNCHR_TRIG_READBACK1

Table 578. PWM_SYNCHR_TRIG_READBACK1 field descriptions

Field	Description
30 PWM3_EXT_FORCE_IN	0 PWM3_EXT_FORCE trigger output pulse was not generated 1 PWM3_EXT_FORCE trigger output pulse was generated
31 PWM2_EXT_FORCE_IN	0 PWM2_EXT_FORCE trigger output pulse was not generated 1 PWM2_EXT_FORCE trigger output pulse was generated

37.3.2.9 PWM_SYNCHR_DLY_CNTR1

This register works on PBRIDGE_1.

Offset: 0x0004 Access: read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DLY_CNT_PWM3												0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	DLY_CNT_PWM2												0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 506. PWM_SYNCHR_DLY_CNTR1

Table 579. PWM_SYNCHR_DLY_CNTR1 field descriptions

Field	Description
0:11 DLY_CNT_PWM3	Start delay value to trigger PWM3 in number of prescaled clock cycles - 1. Note that predicting exact delay not possible as registers work in motor clock domain and trigger generating logic works in another domain (divided clock).
16:27 DLY_CNT_PWM2	Start delay value to trigger PWM2 in number of prescaled clock cycles - 1. Note that predicting exact delay not possible as registers work in motor clock domain and trigger generating logic works in another domain (divided clock).

37.3.2.10 PWM_SYNCHR_STRT1

This register is written by PBRIDGE_1 only.

Offset: 0x0008 Access: write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																FH1
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 507. PWM_SYNCHR_STRT1

Table 580. PWM_SYNCHR_DLY_CNTR1 field descriptions

Field	Description
31 FH1	0 No action for write 0 1 Half/Full sync mode start These bits are write once and read zero always. The full sync mode starts after both FH0 and FH1 have been written.

37.3.2.11 PWM_SYNCHR_CONFIG1

This register is written by PBRIDGE_1 only.

Offset: 0x000C Access: read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	HALF_FULL1	0	PWM32_EN	
W														SR1		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 508. PWM_SYNCHR_CONFIG1

Table 581. PWM_SYNCHR_CONFIG1 field descriptions

Field	Description
28 HALF_FULL1	0 Half sync mode selected 1 Full sync mode selected
29 SR1	1 Soft reset all the PBRIDGE_1 status logic registers, internal counters and outputs. 0 No effect This bit is write once, its always reads as zero.
30:31 PWM32_EN	00 PWM2 disabled, PWM3 disabled 01 PWM2 enabled, PWM3 disabled 10 PWM3 enabled, PWM2 disabled 11 PWM3 enabled, PWM2 enabled

37.3.2.12 PWM_SYNCHR_INT_EN1

This register is written by PBRIDGE_1 only.

Offset: 0x0010 Access: read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	INT_EN3	INT_EN2
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 509. PWM_SYNCHR_INT_EN1

Table 582. PWM_SYNCHR_INT_EN1 field descriptions

Field	Description
30 INT_EN3	1 Enables the interrupt generation for PWM3 0 Disables the interrupt generation
31 INT_EN2	1 Enables the interrupt generation for PWM2 0 Disables the interrupt generation

37.3.2.13 PWM_SYNCHR_PRE_SCALAR1

This register is written by PBRIDGE_1 only.

Offset: 0x0014 Access: read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	PRE_SCALE1			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 510. PWM_SYNCHR_PRE_SCALAR1

Table 583. PWM_SYNCHR_PRE_SCALAR1 field descriptions

Field	Description
28:31 PRE_SCALE1	PRE_SCALAR1 + 1 gives the value for the clock division factor This value should be the same as PRE_SCALE0 for the full sync mode

37.3.2.14 PWM_SYNCHR_INT_STAT1

This register is written by PBRIDGE_1 only.

Offset: 0x0018 Access: read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	ERR1	INT3	INT2
W														w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 511. PWM_SYNCHR_INT_STAT1

Table 584. PWM_SYNCHR_INT_STAT1 field descriptions

Field	Description
29 ERR1	0 No Configuration mismatch between PBRIDGE_0 and PBRIDGE_1 registers 1 Configuration mismatch seen between PBRIDGE_0 and PBRIDGE_1 registers Configuration Registers compared are PWM_SYNCHR_PRESCALE0 and PWM_SYNCHR_PRESCALE1, HALF_FULL0 and HALF_FULL1 bits of PWM_SYNCHR_CONFIGx registers (Comparison is done when the trigger start bits, FH0 or FH1, are written)
30 INT3	0 No interrupt was generated. 1 PWM3 interrupt was generated.
31 INT2	0 No interrupt was generated. 1 PWM2 interrupt was generated.

37.4 Programming Sequence

For half sync mode

1. Program the relevant prescalar register first (PWM_SYNCHR_PRE_SCALAR0/1) to divide the clock.
2. Then the remaining configuration registers (such as PWM_SYNCHR_CONFIG0/1 and PWM_SYNCHR_INT_EN0/1), in any order.
3. Finally the relevant start register, PWM_SYNCHR_STRT0/1.

For full sync mode

1. Program PBRIDGE_0 and PBRIDGE_1 prescalar registers first (PWM_SYNCHR_PRE_SCALAR1 and PWM_SYNCHR_PRE_SCALAR0) with the same value.
2. Configure registers PWM_SYNCHR_CONFIG0 and PWM_SYNCHR_CONFIG1 with the same value, 1, for HALF_FULL0 and HALF_FULL1 fields.
3. PWM_SYNCHR_DLY_CNTR0 and PWM_SYNCHR_DLY_CNTR1 with the required delay values and optionally the PWM_SYNCHR_INT_EN0 and PWM_SYNCHR_INT_EN1 interrupt enable registers.
4. Finally the PWM_SYNCHR_STRT0 and PWM_SYNCHR_STRT1, in any order.

37.5 Functional description

The PWM synchronizer has two clocks and two separate PBRIDGE interfaces to guarantee the separation and redundancy of motor control sub-system 0 (made by eTimer0/1, PWM0/1, ADC0/1/2/B) and motor control sub-system 1 (made by eTimer2/3, PWM2/3, ADC3/4/5/B1).

The synchronizer is built in such a way that it is split in the two different LBISTpartitions, having few signals crossing the partitions. The idea is to have two sub-modules in PWM synchronizer, one for managing PBRIDGE_0 and triggers for PWM0/1 and a similar one for managing PBRIDGE_1 and triggers for PWM2/3. These two sub-modules are interconnected with glue logic in between them.

The two PBRIDGE interfaces work independently. There are two modes to generate triggers namely full sync *Figure 512* and half sync *Figure 513* mode. In full sync mode, all the four trigger outputs are generated at the same time after the programmed delays, based on the respective trigger enables, after sync start bits(FH0, FH1) are written, for both of the interface, PBRIDGE_0 and PBRIDGE_1.

In half sync mode, the respective two triggers are generated, after the programmed delay, depending on which trigger is enabled. Writing FH0 may generate both pwm0_ext_force0 and pwm1_ext_force and writing FH1 may generate both pwm2_force_ext and pwm3_ext_force outputs, as shown in *Figure 513*. The four programmed delays are independent. At the time of generation of trigger outputs, the interrupts are also generated and their status latched, based on their respective enable values.

Figure 512. PWM synchronizer (full sync mode)

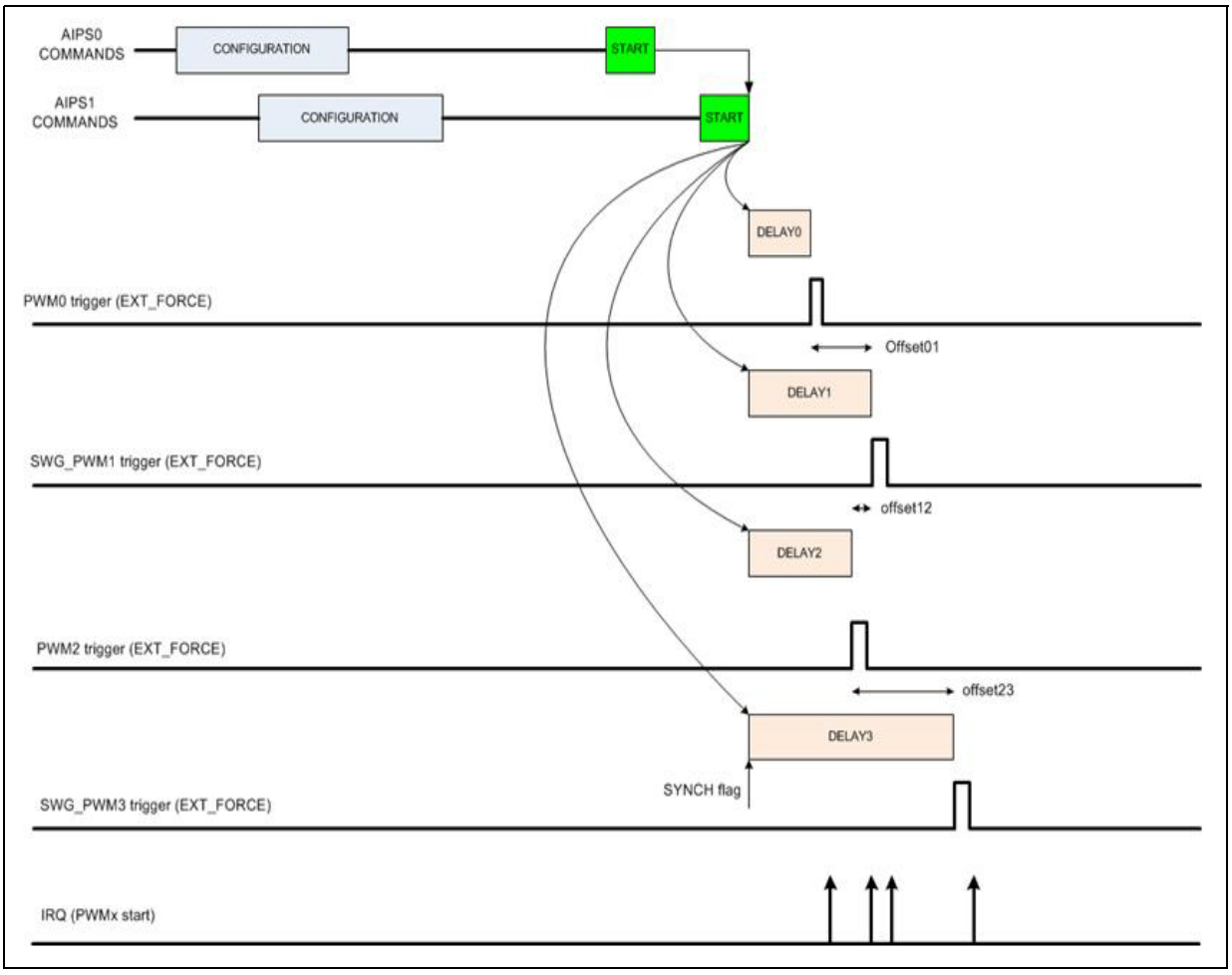


Figure 513. PWM synchronizer (half sync mode)

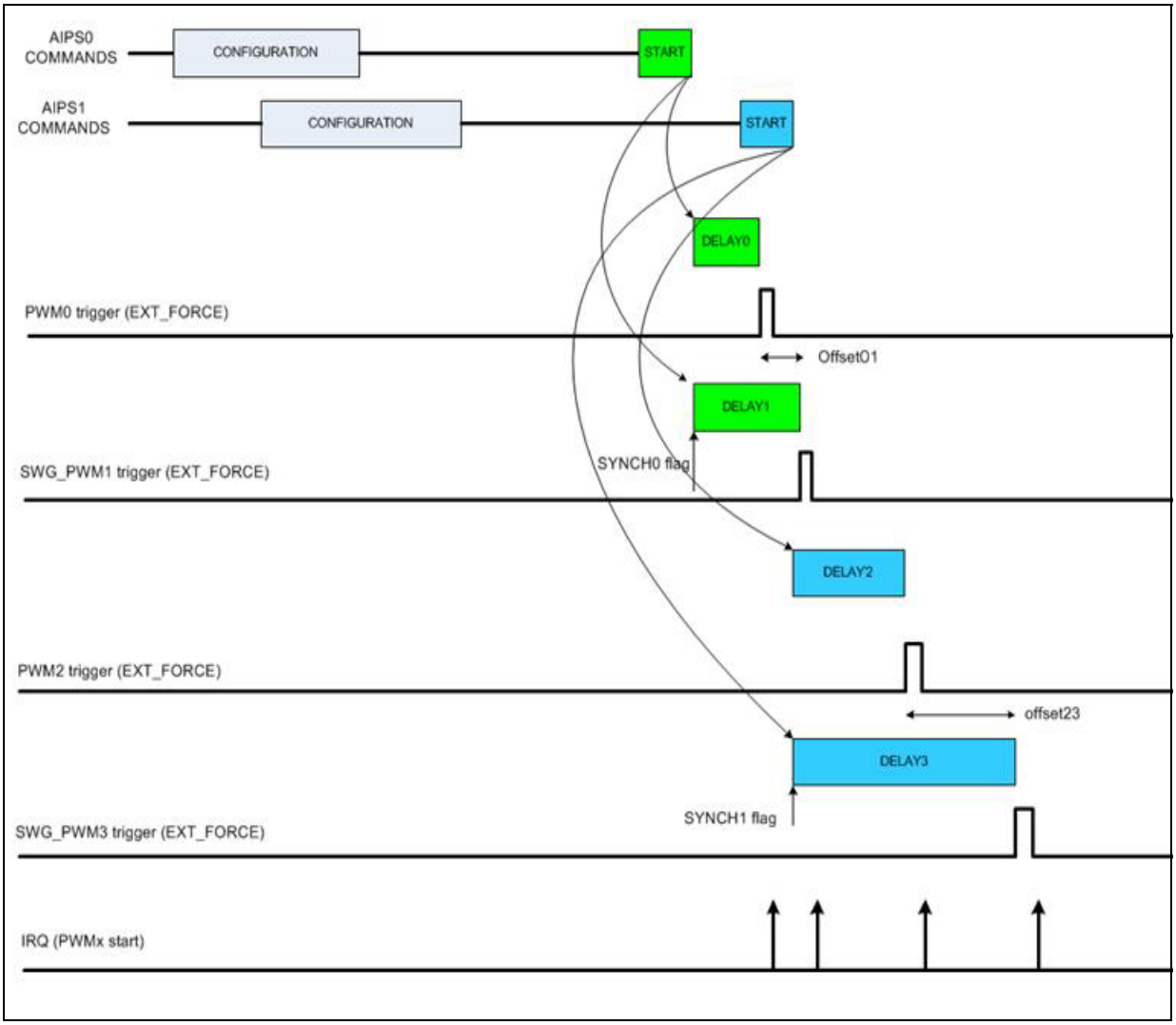
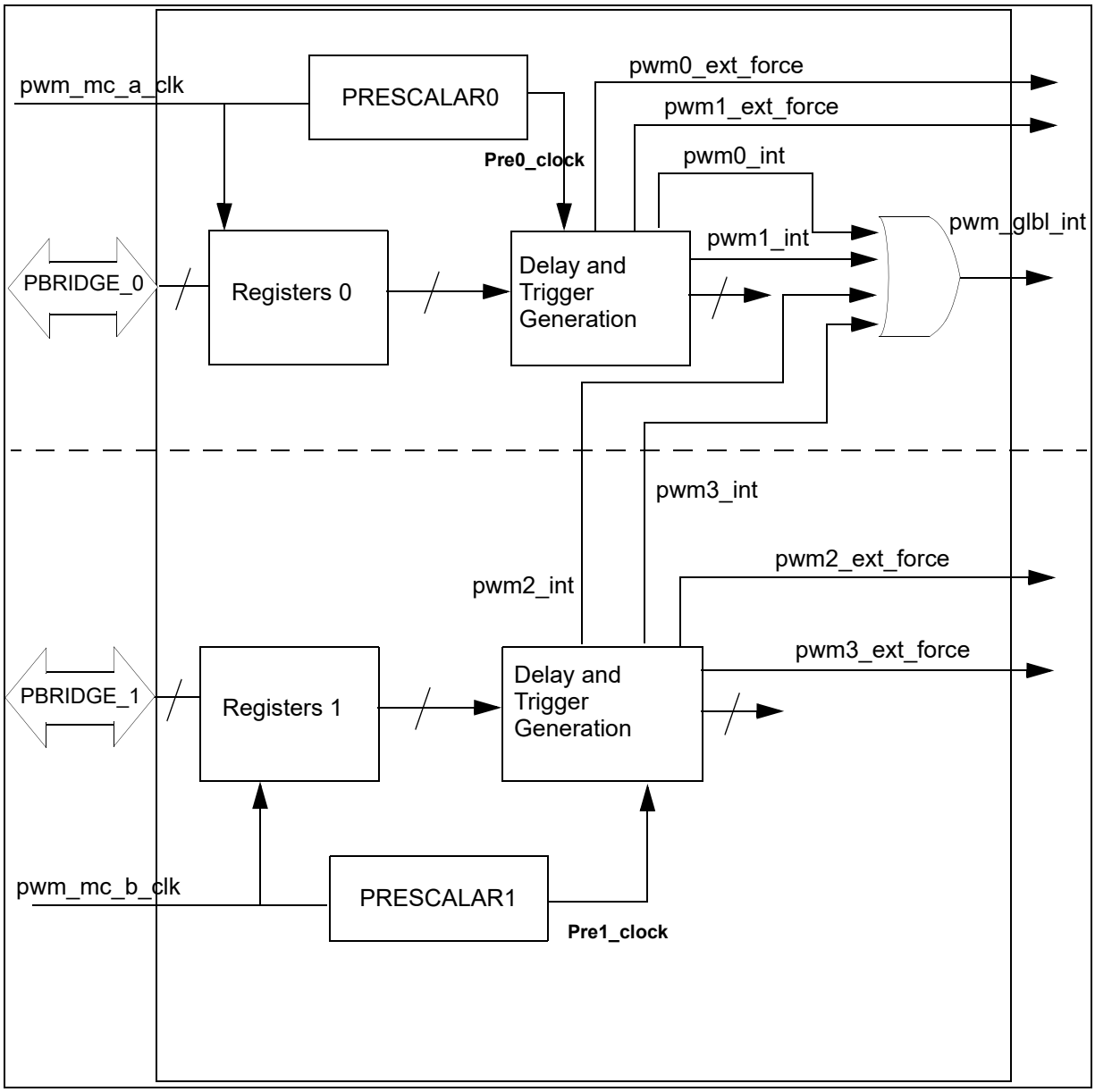


Figure 514. PWM SYNCHR



38 Enhanced Motor Control Timer (eTimer)

38.1 Introduction

38.1.1 Overview

An eTimer module provides:

- Six identical counter/timer channels
- One watchdog timer function (might not be available on every eTimer module instance)

Each 16-bit counter/timer channel contains a prescaler, a counter, a load register, a hold register, two queued capture registers, two compare registers, two compare preload registers, and four control registers.

Note: This document uses the terms "Timer" and "Counter" interchangeably because the counter/timers may perform either or both tasks.

The Load register provides the initialization value to the counter when the counter's terminal value has been reached. For true modulo counting the counter can also be initialized by the CMPLD1 or CMPLD2 registers.

The Hold register captures the counter's value when other counters are being read. This feature supports the reading of cascaded counters coherently.

The Capture registers enable an external signal to take a "snap shot" of the counter's current value.

The COMP1 and COMP2 registers provide the values to which the counter is compared. If a match occurs, the OFLAG signal can be set, cleared, or toggled. At match time, an interrupt is generated if enabled, and the new compare value is loaded into the COMP1 or COMP2 registers from CMPLD1 and CMPLD2 if enabled.

The Prescaler provides different time bases useful for clocking the counter/timer.

The Counter provides the ability to count internal or external events.

Within the eTimer module, the input pins are shareable.

38.1.2 Features

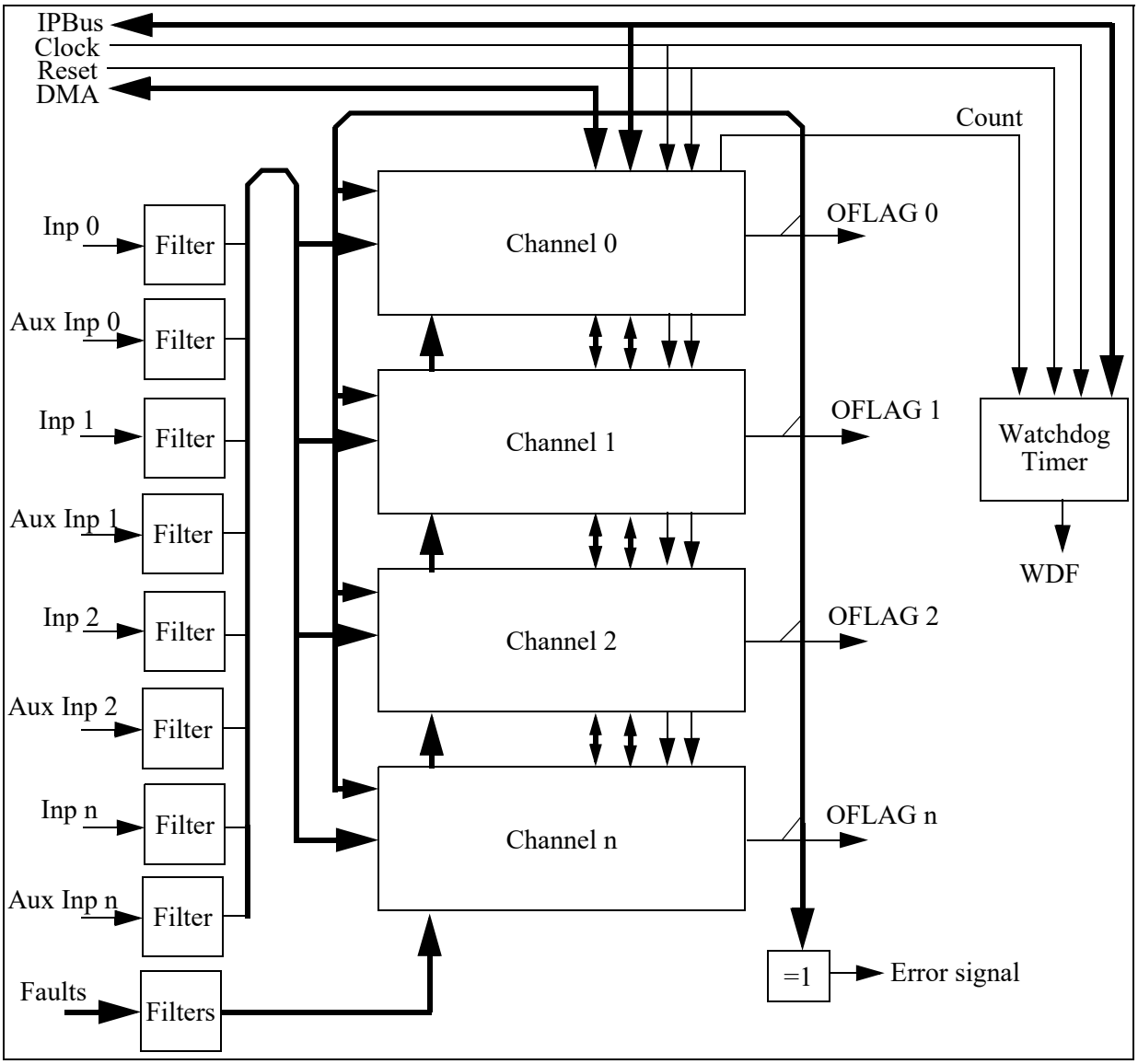
The eTimer module design includes these distinctive features:

- 16-bit counters/timers: 6
- Count up/down
- Counters are cascadable
- Enhanced programmable up/down modulo counting
- Max count rate equals peripheral clock/2 for external clocks
- Max count rate equals peripheral clock for internal clocks
- Count once or repeatedly
- Counters are preloadable
- Compare registers are preloadable
- Counters can share available input pins
- Separate prescaler for each counter
- Each counter has capture and compare capability
- Continuous and single shot capture for enhanced speed measurement
- DMA support of capture registers and compare registers
- 32-bit watchdog capability to detect stalled quadrature counting (might not be available on every eTimer module instance)
- OFLAG comparison for safety critical applications
- Programmable operation during debug mode and stop mode
- Programmable input filter
- Counting start can be synchronized across counters.

38.1.3 Module block diagram

The eTimer block diagram is shown in *Figure 515*.

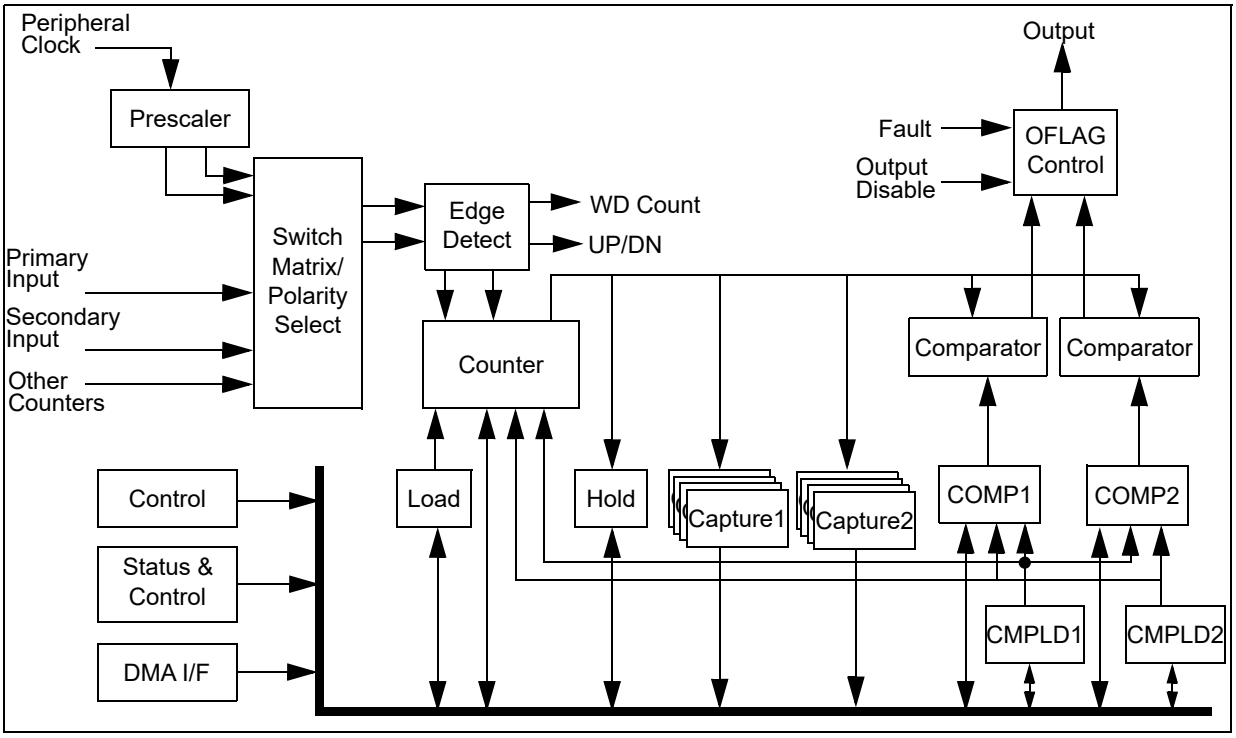
Figure 515. eTimer Block Diagram



38.1.4 Channel block diagram

Each of the timer/counter channels within the eTimer are shown in *Figure 516*.

Figure 516. eTimer channel block diagram



38.2 External signal descriptions

The eTimer module has:

- Six external signals that can be used as either inputs or outputs
- Interfaces to the Peripheral Bus
- Two auxiliary inputs

38.2.1 TIO[*n*:0] - Timer Input/Outputs

These pins can be independently configured to be either timer input sources or output flags. In *Figure 516*, the TIO signals are shown as the Primary Inputs and the Outputs from the OFLAG Control block

38.2.2 TAI[*n*:0] - Timer Auxiliary Inputs

These pins act as alternate input choices for the timer channels. The TAI signals are shown as Aux Inp 1..n in *Figure 515*.

38.3 Functional Description

38.3.1 General

Each channel has two basic modes of operation: it can count internal or external events, or it can count an internal clock source while an external input signal is asserted, thus timing the width of the external input signal.

- The counter can count the rising, falling, or both edges of the selected input pin.
- The counter can decode and count quadrature encoded input signals.
- The counter can count up and down using dual inputs in a “count with direction” format.
- The counter’s terminal count value (modulo) is programmable.
  - The value that is loaded into the counter after reaching its terminal count is programmable.
- The counter can count repeatedly, or it can stop after completing one count cycle.
- The counter can be programmed to count to a programmed value and then immediately reinitialize, or it can count through the compare value until the count “rolls over” to zero.

The external inputs to each counter/timer are shareable among each of the channels within the module. The external inputs can be used as:

- Count commands
- Timer commands
- They can trigger the current counter value to be “captured”
- They can be used to generate interrupt requests

The polarity of the external inputs is selectable.

The primary output of each channel is the output signal OFLAG. The OFLAG output signal can be:

- Set, cleared, or toggled when the counter reaches the programmed value.
- The OFLAG output signal may be output to an external pin instead of having that pin serve as a timer input.
- The OFLAG output signal enables each counter to generate square waves, PWM, or pulse stream outputs.
- The polarity of the OFLAG output signal is programmable.

Any channel can be assigned as a “Master”. A master’s compare signal can be broadcasted to the other channels within the module. The other channels can be configured to reinitialize their counters and/or force their OFLAG output signals to predetermined values when a Master channel’s compare event occurs.

38.3.2 Counting Modes

The selected external signals are sampled at the eTimer’s base clock rate and then run through a transition detector. The maximum count rate is one-half of the eTimer’s base clock rate when using an external signal. Internal clock sources can be used to clock the counters at the eTimer’s base clock rate.

If a counter is programmed to count to a specific value and then stop, the CNTMODE field in the CTRL1 register is cleared when the count terminates.

38.3.2.1 STOP Mode

If the CNTMODE field is set to ‘000’, the counter is inert. No counting will occur. Stop mode will also disable the interrupts caused by input transitions on a selected input pin.

38.3.2.2 COUNT Mode

If the CNTMODE field is set to ‘001’, the counter will count the rising edges of the selected clock source. This mode is useful for generating periodic interrupts for timing purposes, or counting external events such as “widgets” on a conveyor belt passing a sensor. If the selected input is inverted by setting the PIPS bit, then the negative edge of the selected external input signal is counted.

See *Section 38.3.2.9: CASCADE-COUNT Mode*, through *Section 38.3.2.12: VARIABLE-FREQUENCY PWM Mode*, for additional capabilities of this operating mode.

38.3.2.3 EDGE-COUNT Mode

If the CNTMODE field is set to ‘010’, the counter will count both edges of the selected external clock source. This mode is useful for counting the changes in the external environment such as a simple encoder wheel.

38.3.2.4 GATED-COUNT Mode

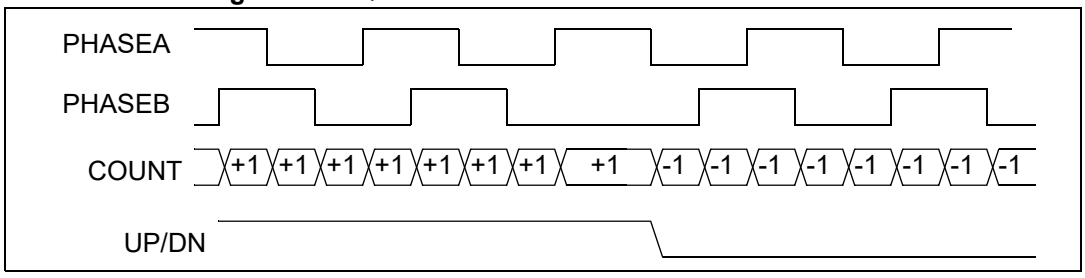
If the CNTMODE field is set to ‘011’, the counter will count while the selected secondary input signal is high. This mode is used to time the duration of external events. If the selected input is inverted by setting the SIPS bit, then the counter will count while the selected secondary input is low.

38.3.2.5 QUADRATURE-COUNT Mode

If the CNTMODE field is set to ‘100’, the counter will decode the primary and secondary external inputs as quadrature encoded signals. Quadrature signals are usually generated by rotary or linear sensors used to monitor movement of motor shafts or mechanical equipment. The quadrature signals are square waves that are 90 degrees out of phase. The decoding of quadrature signal provides both count and direction information.

Figure 517 shows a timing diagram illustrating the basic operation of a quadrature incremental position encoder.

Figure 517. Quadrature Incremental Position Encoder



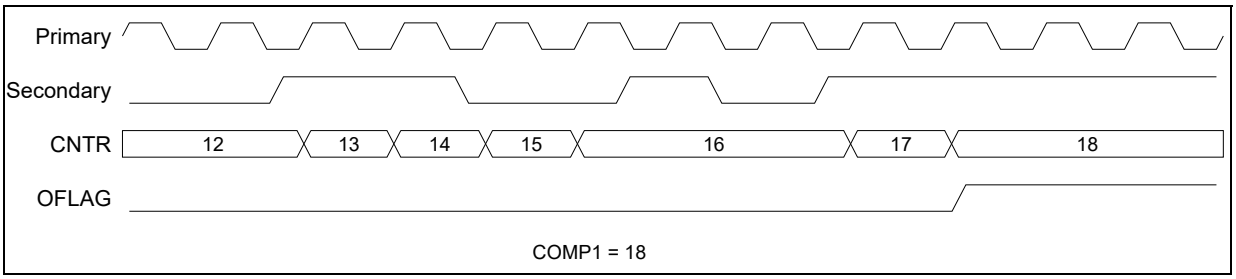
38.3.2.6 SIGNED-COUNT Mode

If the CNTMODE field is set to ‘101’, the counter counts the primary clock source while the selected secondary source provides the selected count direction (up/down).

38.3.2.7 TRIGGERED-COUNT Mode

If the CNTMODE field is set to ‘110’, the counter will begin counting the primary clock source after a positive transition (negative if SIPS = 1) of the secondary input occurs. The counting will continue until a compare event occurs or another positive input transition is detected. Subsequent secondary positive input transitions will continue to restart and stop the counting until a compare event occurs.

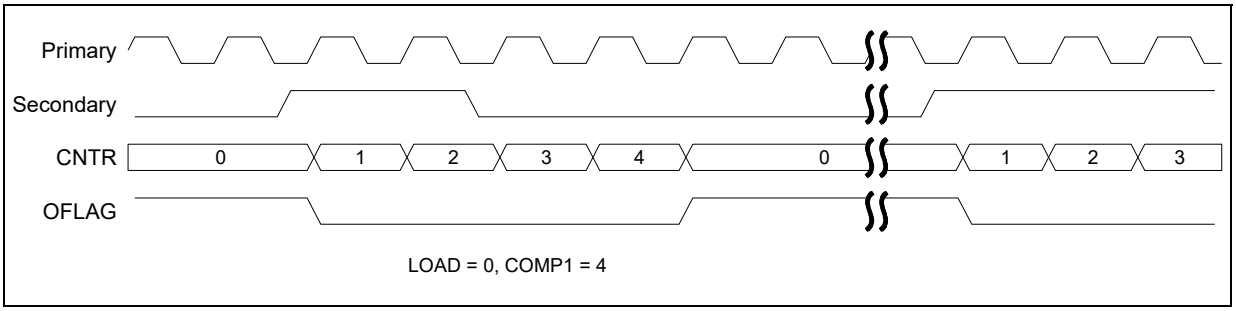
Figure 518. Triggered Count Mode (Length=1)



38.3.2.8 ONE-SHOT Mode

If the CNTMODE field is set to ‘110’, and the counter is set to reinitialize at a compare event (LENGTH =1), and the OFLAG OUTMODE is set to ‘0101’ (cleared on init, set on compare), the counter works in a “One-Shot Mode”. An external events causes the counter to count, when terminal count is reached, the output is asserted. This “delayed” output can be used to provide timing delays.

Figure 519. One-Shot Mode (Length=1)



38.3.2.9 CASCADE-COUNT Mode

If the CNTMODE field is set to ‘111’, the counter’s input is connected to the output of another selected counter. The counter will count up and down as compare events occur in the selected source counter. This “Cascade” or “Daisy-Chained” mode enables multiple counters to be cascaded to yield longer counter lengths. When operating in cascade mode, a special high speed signal path is used between modules rather than the OFLAG output signal. If the selected source counter is counting up and it experiences a compare event, the counter will be incremented. If the selected source counter is counting down and it experiences a compare event, the counter will be decremented.

Up to two counters may be cascaded to create a 32 bit wide synchronous counter.

Whenever any counter is read within a counter module, all of the counters’ values within the module are captured in their respective HOLD registers. This action supports the reading of a cascaded counter chain. First read any counter of a cascaded counter chain, then read

the HOLD registers of the other counters in the chain. The cascaded counter mode is synchronous.

Note: It is possible to connect counters together by using the other (non-cascade) counter modes and selecting the outputs of other counters as a clock source. In this case, the counters are operating in a "ripple" mode, where higher order counters will transition a clock later than a purely synchronous design.

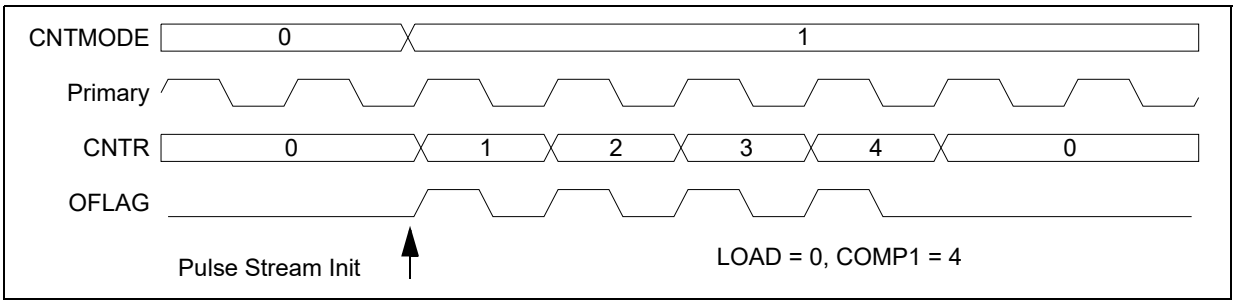
A channel can be cascaded with any other channel, but you can't cascade more than 2 channels together. You can create separate cascades of pairs of channels. For example, you can cascade channels 0 and 1 and separately cascade channels 6 and 5. You can't cascade channels 0, 1, and 5.

38.3.2.10 PULSE-OUTPUT Mode

If the counter is setup for CNTMODE = 001, and the OFLAG OUTMODE is set to '1111' (gated clock output), and the ONCE bit is set, then the counter will output a pulse stream of pulses that has the same frequency of the selected clock source, and the number of output pulses is equal to the compare value minus the init value. This mode is useful for driving step motor systems.

Note: This does not work if the PRISRC is set to '11000' (IP_bus/1).

Figure 520. Pulse Output Mode



38.3.2.11 FIXED-FREQUENCY PWM Mode

If the counter is setup for CNTMODE = 001, count through roll-over (LENGTH = 0), continuous count (ONCE = 0) and the OFLAG OUTMODE is '0111' (set on compare, cleared on counter roll-over) then the counter output yields a Pulse Width Modulated (PWM) signal with a frequency equal to the count clock frequency divided by 65,536 and a pulse width duty cycle equal to the compare value divided by 65,536. This mode of operation is often used to drive PWM amplifiers used to power motors and inverters.

38.3.2.12 VARIABLE-FREQUENCY PWM Mode

If the counter is setup for CNTMODE = 001, count till compare (LENGTH = 1), continuous count (ONCE = 0) and the OFLAG OUTMMODE is '0100' (toggle OFLAG and alternate compare registers) then the counter output yields a Pulse Width Modulated (PWM) signal whose frequency and pulse width is determined by the values programmed into the COMP1 and COMP2 registers, and the input clock frequency. This method of PWM generation has the advantage of allowing almost any desired PWM frequency and/or constant on or off periods. This mode of operation is often used to drive PWM amplifiers used to power motors and inverters. The CMPLD1 and CMPLD2 registers are especially useful for this mode, as they allow the programmer time to calculate values for the next PWM cycle while the PWM current cycle is underway.

Figure 521. Variable PWM Waveform

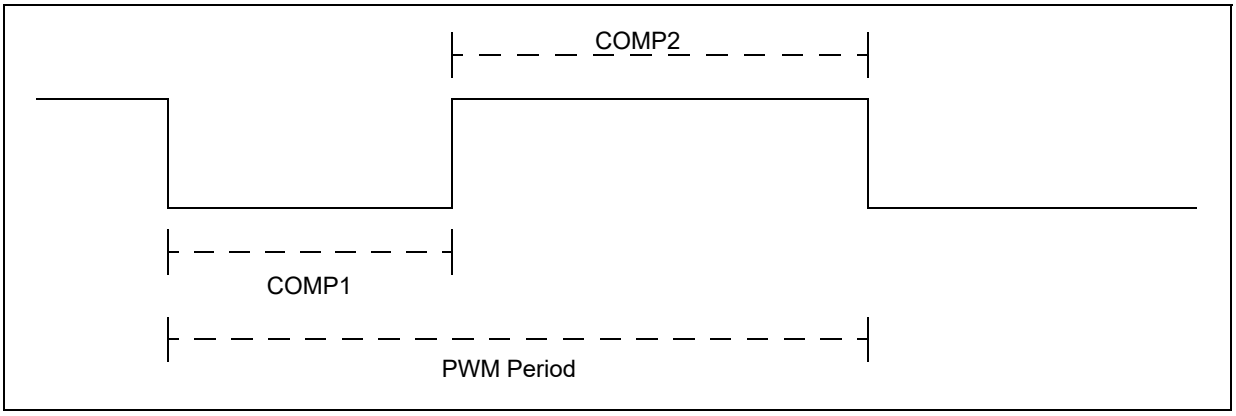
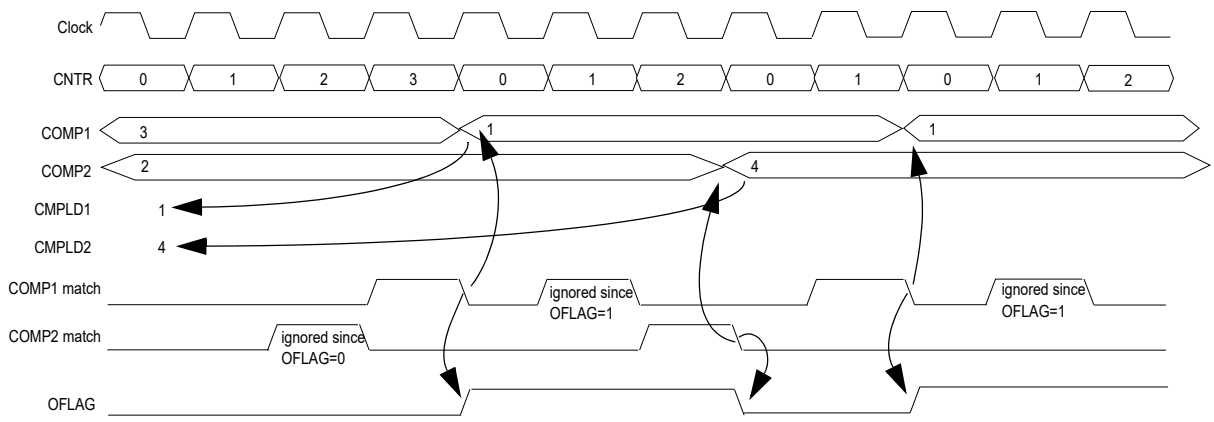


Figure 522. Variable Frequency PWM Mode Timing



CLC1=010, load COMP1 when CNTR=COMP1,
Load COMP1 with CMPLD1 upon successful compare with the value in COMP1

CLC2=101, load COMP2 when CNTR=COMP2,
Load COMP1 with CMPLD2 upon successful compare with the value in COMP2

38.3.2.13 Usage of Compare Registers

The dual compare registers (COMP1 and COMP2) provide a bidirectional modulo count capability.

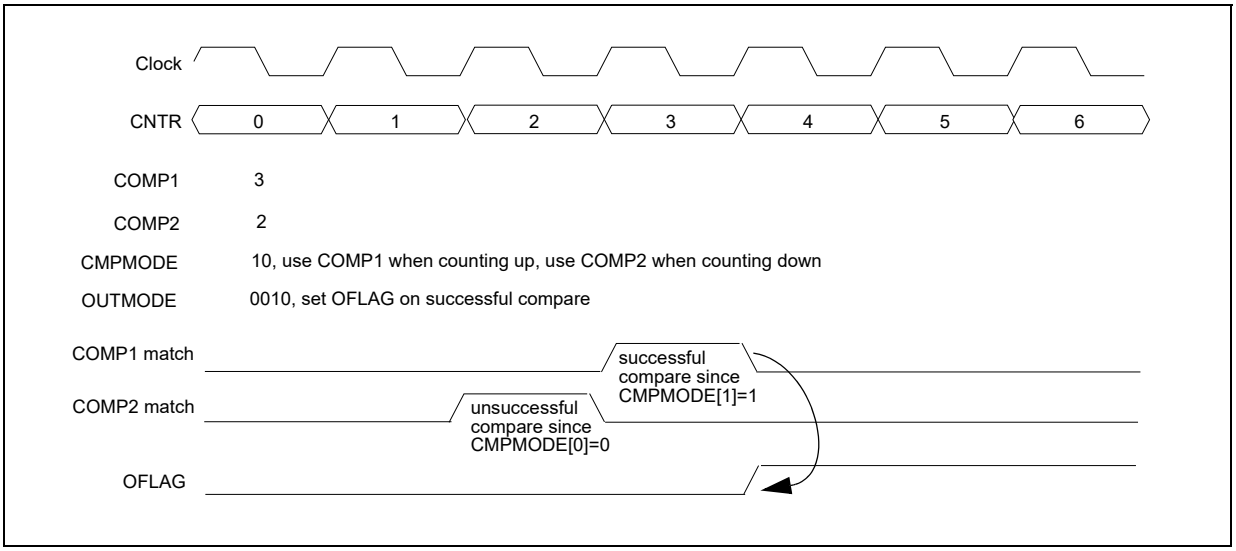
The COMP1 register should be set to the desired maximum count value or 0xFFFF to indicate the maximum unsigned value prior to roll-over, and the COMP2 register should be set to the minimum count value or 0x0000 to indicate the minimum unsigned value prior to roll-under.

If the output mode is set to 0100, the OFLAG will toggle while using alternating compare registers. In this variable frequency PWM mode, the COMP2 value defines the desired pulse width of the on time, and the COMP1 register defines the off time. COMP1 is used when OFLAG=0 and COMP2 is used when OFLAG=1. OFLAG can be forced to a value using CTRL2[FORCE] and CTRL2[VAL].

Use caution when changing COMP1 and COMP2 while the counter is active. If the counter has already passed the new value, it will count to 0xFFFF or 0x0000, roll over, then begin counting toward the new value. The check is: CNTR = COMPx, *not* CNTR > COMP1 or CNTR < COMP2.

The use of the CMPLD1 and CMPLD2 registers to preload compare values will help to minimize this problem.

Figure 523. Compare Register and OFLAG Timing



38.3.2.14 Usage of Compare Load Registers

The CMPLD1, CMPLD2 and CCCTRL registers offer a high degree of flexibility for loading compare registers with user-defined values on different compare events. To ensure correct functionality while using these registers we strongly suggest using the following method described in this section.

The purpose of the compare load feature is to allow quicker updating of the compare registers. A compare register can be updated using interrupts. However, because of the latency between an interrupt event occurring and the service of that interrupt, there is the possibility that the counter may have already counted past the new compare value by the time the compare register is updated by the interrupt service routine. The counter would then continue counting until it rolled over and reached the new compare value.

To address this, the compare registers are updated in hardware in the same way the counter register is re-initialized to the value stored in the LOAD register. The compare load feature allows the user to calculate new compare values and store them in to the comparator load registers. When a compare event occurs, the new compare values in the comparator load registers are written to the compare registers eliminating the use of software to do this.

The compare load feature is intended to be used in variable frequency PWM mode. The COMP1 register determines the pulse width for the logic low part of OFLAG and COMP2 determines the pulse width for the logic high part of OFLAG. The period of the waveform is determined by the COMP1 and COMP2 values and the frequency of the primary clock source. See *Figure 521*.

38.3.2.15 MODULO COUNTING Mode

To create a modulo counter using COMP1 and COMP2 as the counter boundaries (instead of 0x0000 and 0xFFFF), set the registers in the following manner. Set CNTMODE to either 100 (quadrature count mode) or 101 (count with direction mode). Use count through roll-over (LENGTH = 0) and continuous count (ONCE = 0). Set COMP1 and CMPLD1 to the upper boundary value. Set COMP2 and CMPLD2 to the lower boundary value. Set CMPMODE = 10 (COMP1 is used when counting up and COMP2 is used when counting down). Set CLC2 = 110 (load CNTR with value of CMPLD2 on COMP1 compare) and CLC1 = 111 (load CNTR with value of CMPLD1 on COMP2 compare).

38.3.2.16 Compare Register and OFLAG Operation

The compare flags and output flag are registered signals and can only change state after the counter reached the compare value. This means that, if the counter is continuously counting, then the compare flag is set on the count after the counter reaches the compare value. When counting is not continuous (such as when using a prescaler or when counting edges on the primary input), the compare flag or OFLAG are not updated until the counter is about to transition to the next count. This means that if the compare value is 4, then the flags will be changed on the same clock edge that transitions the counter from 4 to its next value.

Figure 524. OFLAG timing during non-continuous counting

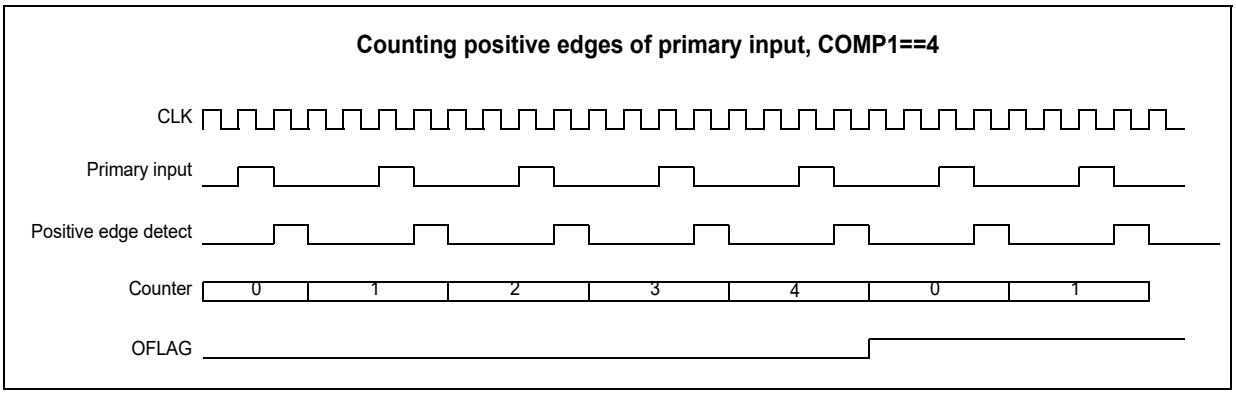
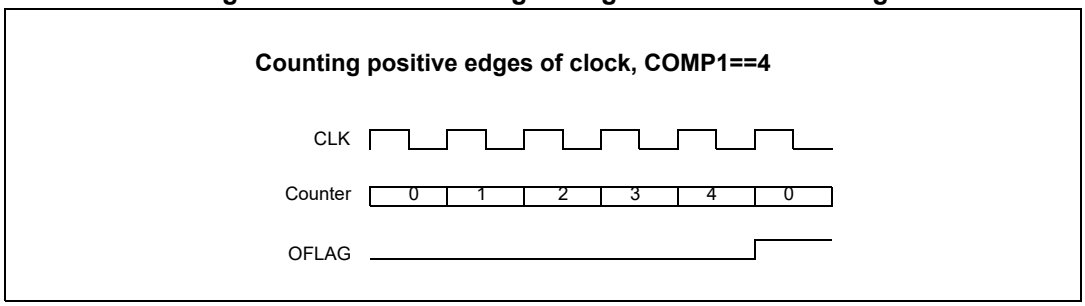


Figure 525. OFLAG timing during continuous counting



38.3.3 Other Features

38.3.3.1 Redundant OFLAG Checking

This mode allows the user to bundle two timer functions generating any pattern to compare their resulting OFLAG behaviors (output signal).

The redundant mode is used to support online checks for functional safety reasons. Whenever a mismatch between the two adjacent channels occurs, it is reported via an interrupt to the core and the two outputs are put into their inactive states. An error is flagged via the RCF flag.

This feature can be tested by forcing a transition on one of the OFLAGs using the VAL and FORCE bits of the channel.

38.3.3.2 Loopback Checking

This mode is always available in that one channel can generate an OFLAG while another channel uses the first channels OFLAG as its input to be measured and verified to be as expected.

38.3.3.3 Input Capture Mode

Input capture is used to measure pulse width (by capturing the counter value on two successive input edges) or waveform period (by capturing the counter value on two consecutive rising edges or two consecutive falling edges). The Capture Registers store a copy of the counter's value when an input edge (positive, negative, or both) is detected. The type of edge to be captured by each circuit is determined by the CPT1MODE and CPT2MODE bits whose functionality is listed in *Table 599*. Also, controlling the operation of the capture circuits is the arming logic which allows captures to be performed in a free running (continuous) or one shot fashion. In free running mode, the capture sequences will be performed indefinitely. If both capture circuits are enabled, they will work together in a ping-pong style where a capture event from one circuit leads to the arming of the other and vice versa. In one shot mode, only one capture sequence will be performed. If both capture circuits are enabled, capture circuit 0 is first armed and when a capture event occurs, capture circuit 1 is armed. Once the second capture occurs, further captures are disabled until another capture sequence is initiated. Both capture circuits are also capable of generating an interrupt to the CPU.

38.3.3.4 Master/Slave Mode

Any timer channel can be assigned as a Master (MSTR = 1). A Master's compare signal can be broadcasted to the other channels within the module. The other counters can be

configured to reinitialize their counters (COINIT = 1) and/or force their OFLAG output signals (COFRC = 1) to predetermined values when a Master counter compare event occurs.

38.3.3.5 Watchdog Timer

Note: The watchdog timer may not be implemented on each eTimer module instance.

The watchdog timer is used to monitor for a stalled count when channel 0 is in quadrature count mode. When the watchdog is enabled, it loads the time-out value into a down counter. The down counter counts as long as channel 0 is in quadrature decode count mode. If this down counter reaches zero, an interrupt is asserted. The down counter is reloaded to the time-out value each time the counter value from channel 0 changes. If the channel 0 count value is toggling between two values (indicating a possibly stalled encoder), then the down counter is not reloaded.

The watchdog function of the eTimer counts using the peripheral bus clock which is the same clock as used by the eTimer counter channels. The watchdog function looks for counting activity in counter #0.

38.4 Memory Map and Registers

38.4.1 Overview

The base address of the eTimer module will differ from chip to chip. All memory mapped registers described below have their location described in relation to the BASE Address.

Note: The actual availability, configuration and number of registers are device-specific. For this information, see Chapter 7: Device configuration.

38.4.2 Module Memory Map

Table 585. eTimer Memory Map

Address offset	Reg Name	Description	Location
Timer channel registers per channel (n=0 to 5)			
0x0000 + n*0x20	COMP1n	Compare Register 1	*Section 38.4.4.1*
0x0002 + n*0x20	COMP2n	Compare Register 2	*Section 38.4.4.2*
0x0004 + n*0x20	CAPT1n	Capture Register 1	*Section 38.4.4.3*
0x0006 + n*0x20	CAPT2n	Capture Register 2	*Section 38.4.4.4*
0x0008 + n*0x20	LOADn	Load Register	*Section 38.4.4.5*
0x000A + n*0x20	HOLDn	Hold Register	*Section 38.4.4.6*
0x000C + n*0x20	CNTRn	Counter Register	*Section 38.4.4.7*
0x000E + n*0x20	CTRL1n	Control Register 1	*Section 38.4.4.8*
0x0010 + n*0x20	CTRL2n	Control Register 2	*Section 38.4.4.9*
0x0012 + n*0x20	CTRL3n	Control Register 3	*Section 38.4.4.10*
0x0014 + n*0x20	STSn	Status Register	*Section 38.4.4.11*

Table 585. eTimer Memory Map (continued)

Address offset	Reg Name	Description	Location
0x0016 + n*0x20	INTDMAn	Interrupt and DMA Enable Register	*Section 38.4.4.12*
0x0018 + n*0x20	CMPLD1n	Comparator Load Register 1	*Section 38.4.4.13*
0x001A + n*0x20	CMPLD2n	Comparator Load Register 2	*Section 38.4.4.14*
0x001C + n*0x20	CCCTRLn	Compare and Capture Control Register	*Section 38.4.4.15*
0x001E + n*0x20	FILTn	Input Filter Register	*Section 38.4.4.16*
Watchdog timer registers(1)			
0x0100	WDTOL	Watchdog Time-out Low World Register	*Section 38.4.5.1*
0x0102	WDTOH	Watchdog Time-out High World Register	*Section 38.4.5.2*
Configuration Registers			
0x010C	ENBL	Channel Enable Register	*Section 38.4.6.1*
0x0110	DREQ0	DMA Request 0 Select Register	*Section 38.4.6.2*
0x0112	DREQ1	DMA Request 1 Select Register	*Section 38.4.6.3*

1. If applicable. Not all eTimer module instantiations implement a watchdog timer.

38.4.3 Register Descriptions

The address of a register is the sum of a base address and an address offset. The base address is defined at the chip level and the address offset is defined at the module level. There are a set of registers for each timer channel, a set for the watchdog timer, and a set of configuration registers. Some registers are labelled as not byte accessible, however, no error will be generated if they are byte accessed.

38.4.4 Timer Channel Registers

These registers are repeated for each timer channel. The base address of channel 0 is the same as the base address of the eTimer module as a whole. The base address of channel 1 is 0x20. This is the base address of the eTimer module plus an offset based on the number of bytes of registers in a timer channel. The base address of each subsequent timer channel is equal to the base address of the previous channel plus this same offset of 0x20.

38.4.4.1 Compare Register 1n (COMP1n)

This read/write register stores the value used for comparison with the counter value. This register is not byte accessible. More explanation on the use of COMP1 can be found in *Section 38.3.2.13: Usage of Compare Registers*.

Offset: 0x0000 + n*0x20 (n=0 to 5) Access: R/W

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	COMP1															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 526. Compare Register 1n (COMP1n)

Table 586. Compare Register 1n (COMP1n) field descriptions

Field	Description
0:15 COMP1	Compare Register1

38.4.4.2 Compare Register 2n (COMP2n)

This read/write register stores the value used for comparison with the counter value. This register is not byte accessible. More explanation on the use of COMP2 can be found in *Section 38.3.2.13: Usage of Compare Registers*.

Offset: 0x0002 + n*0x20 (n=0 to 5) Access: R/W

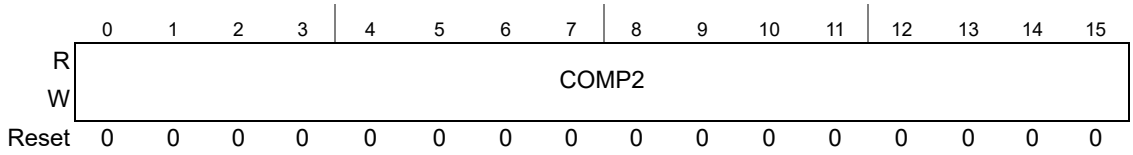


Figure 527. Compare Register 2n (COMP2n)

Table 587. Compare Register 2n (COMP2n) field descriptions

Field	Description
0:15 COMP2	Compare Register2

38.4.4.3 Capture Register 1n (CAPT1n)

This read-only register stores the value captured from the counter. Exactly when a capture occurs is defined by the CPT1MODE bits. This is actually a 2-deep FIFO and not a single register. This register is not byte accessible.

Offset: 0x0004 + n*0x20 (n=0 to 5) Access: Read-only

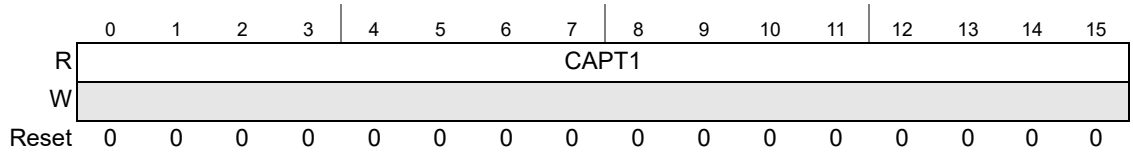


Figure 528. Capture Register 1n (CAPT1n)

Table 588. Capture Register 1n (CAPT1n) field descriptions

Field	Description
0:15 CAPT1	Capture Register1

38.4.4.4 Capture Register 2n (CAPT2n)

This read-only register stores the value captured from the counter. Exactly when a capture occurs is defined by the CPT2MODE bits. This is actually a 2-deep FIFO and not a single register. This register is not byte accessible.

Offset: 0x0006 + n*0x20 (n=0 to 5) Access: Read-only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	CAPT2															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 529. Capture Register 2n (CAPT2n)

Table 589. Capture Register 2n (CAPT2n) field descriptions

Field	Description
0:15 CAPT2	Capture Register2

38.4.4.5 Load Register n (LOADn)

This read/write register stores the value used to initialize the counter. This register is not byte accessible.

Offset: 0x0008 + n*0x20 (n=0 to 5) Access: R/W

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	LOAD															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 530. Load Register n (LOADn)

Figure 531. Load Register n (LOADn) field descriptions

Field	Description
0:15 LOAD	Load Register

38.4.4.6 Hold Register n (HOLDn)

This read-only register stores the counter's value whenever any of the other counters within a module are read. This is used to support coherent reading of cascaded counters.

Offset: 0x000A + n*0x20 (n=0 to 5) Access: Read-only

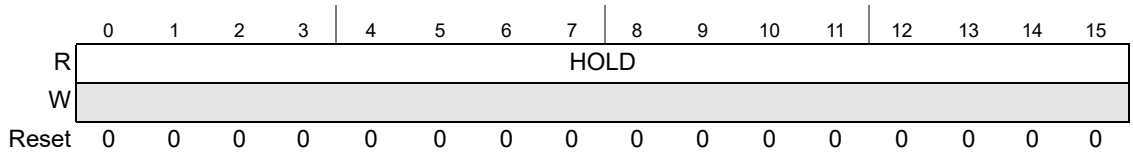


Figure 532. Hold Register n (HOLDn)

Table 590. Hold Register n (HOLDn) field descriptions

Field	Description
0:15 HOLD	Hold Register

38.4.4.7 Counter Register n (CNTRn)

This read/write register is the counter for this channel of the timer module. This register is not byte accessible.

Offset: 0x000C + n*0x20 (n=0 to 5) Access: R/W

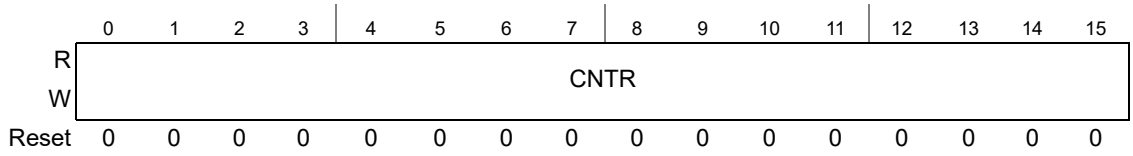


Figure 533. Counter Register n (CNTRn)

Table 591. Counter Register n (CNTRn) field descriptions

Field	Description
0:15 CNTR	Counter Register

38.4.4.8 Control Register 1n (CTRL1n)

Offset: 0x000E + n*0x20 (n=0 to 5) Access: R/W

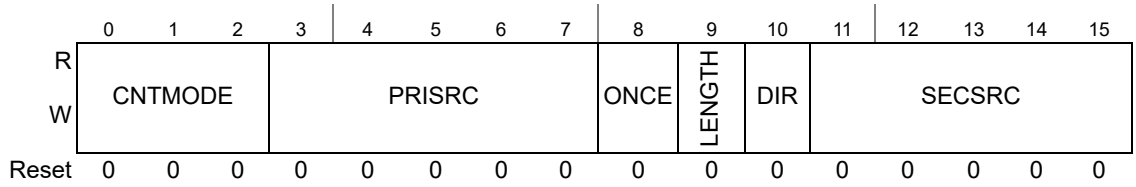


Figure 534. Control Register 1n (CTRL1n)

Table 592. Control Register 1n (CTRL1n) field descriptions

Field	Description
0:2 CNTMODE	Count Mode These bits control the basic counting and behavior of the counter. 000 No operation 001 Count rising edges of primary source(1) 010 Count rising and falling edges of primary source(2) 011 Count rising edges of primary source while secondary input high active 100 Quadrature count mode, uses primary and secondary sources 101 Count primary source rising edges, secondary source specifies direction (1 = minus)(3) 110 Edge of secondary source triggers primary count till compare 111 Cascaded counter mode, up/down(4)
3:7 PRISRC	Primary Count Source These bits select the primary count source. 00000 Counter #0 input pin 00001 Counter #1 input pin 00010 Counter #2 input pin 00011 Counter #3 input pin 00100 Counter #4 input pin 00101 Counter #5 input pin 00110 Reserved 00111 Reserved 01000 Auxiliary input #0 pin 01001 Auxiliary input #1 pin 01010 Reserved 01011 Reserved 01100 Reserved 01101 Reserved 01110 Reserved 01111 Reserved 10000 Counter #0 output 10001 Counter #1 output 10010 Counter #2 output 10011 Counter #3 output 10100 Counter #4 output 10101 Counter #5 output 10110 Reserved 10111 Reserved 11000 IP Bus clock divide by 1 prescaler 11001 IP Bus clock divide by 2 prescaler 11010 IP Bus clock divide by 4 prescaler 11011 IP Bus clock divide by 8 prescaler 11100 IP Bus clock divide by 16 prescaler 11101 IP Bus clock divide by 32 prescaler 11110 IP Bus clock divide by 64 prescaler 11111 IP Bus clock divide by 128 prescaler **Note:** A timer selecting its own output as its primary count source is not a legal choice. The result is no counting.
8 ONCE	Count Once This bit selects continuous or one shot counting mode 1 Count until compare and then stop. When output mode $4 is used, the counter re-initializes after reaching the COMP1 value and continues to count to the COMP2 value then stops. 0 Count repeatedly

Table 592. Control Register 1n (CTRL1n) field descriptions (continued)

Field	Description
9 LENGTH	Count Length This bit determines whether the counter counts to the compare value and then re-initializes itself to the value specified in the LOAD, CMPLD1, or CMPLD2 registers, or the counter continues counting past the compare value, to the binary roll over. 1 Count until compare, then reinitialize. The value that the counter is reinitialized with depends on the settings of CLC1 and CLC2. If neither of these indicates the counter is to be loaded from one of the CMPLD registers, then the LOAD register is used to reinitialize the counter upon matching either COMP register. If one of CLC1 or CLC2 indicates that the counter is to be loaded from one of the CMPLD registers, then the counter will reinitialize to the value in the appropriate CMPLD register upon a match with the appropriate COMP register. If both of the CLC1 and CLC2 fields indicate that the counter is to be loaded from the CMPLD registers, then CMPLD1 will have priority if both compares happen at the same value. When output mode $4 is used, alternating values of COMP1 and COMP2 are used to generate successful comparisons. For example, the counter counts until COMP1 value is reached, re-initializes, then counts until COMP2 value is reached, re-initializes, then counts until COMP1 value is reached, etc. 0 Continue counting to roll over.
10 DIR	Count Direction This bit selects either the normal count direction up, or the reverse direction, down. 1 Count down 0 Count up
11:15 SECSRC	Secondary Count Source These bits identify the source to be used as a count command or timer command. The selected input can trigger the timer to capture the current value of the CNTR. The selected input can also be used to specify the count direction. The polarity of the signal can be inverted by the bit CTRL2[SIPS]. 00000 Counter #0 input pin 00001 Counter #1 input pin 00010 Counter #2 input pin 00011 Counter #3 input pin 00100 Counter #4 input pin 00101 Counter #5 input pin 00110 Reserved 00111 Reserved 01000 Auxiliary input #0 pin 01001 Auxiliary input #1 pin 01010 Reserved 01011 Reserved 01100 Reserved 01101 Reserved 01110 Reserved 01111 Reserved 10000 Counter #0 output 10001 Counter #1 output 10010 Counter #2 output 10011 Counter #3 output 10100 Counter #4 output 10101 Counter #5 output 10110 Reserved 10111 Reserved 11000 - 11111 Reserved

1. Rising edges counted only when PIPS = 0. Falling edges counted when PIPS = 1. If primary count source is IP bus clock, only rising edges are counted regardless of PIPS value.
2. IP Bus clock divide by 1 can not be used as a primary count source in edge count mode.
3. Rising edges counted only when PIPS = 0. Falling edges counted when PIPS = 1.
4. Primary count source must be set to one of the counter outputs.

38.4.4.9 Control Register 2n (CTRL2n)

Offset: 0x0010 + n*0x20 (n=0 to 5) Access: R/W

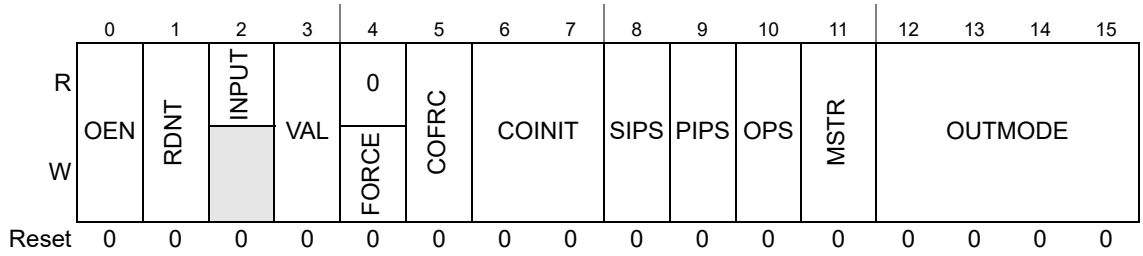


Figure 535. Control Register 2n (CTRL2n)

Table 593. Control Register 2n (CTRL2n) field descriptions

Field	Description
0 OEN	Output Enable This bit determines the direction of the external pin. 1 OFLAG output signal will be driven on the external pin. Other timer channels using this external pin as their input will see the driven value. The polarity of the signal will be determined by the OPS bit. 0 The external pin is configured as an input.
1 RDNT	Redundant Channel Enable This bit enables redundant channel checking between adjacent channels (0 and 1, 2 and 3, 4 and 5). When this bit is clear, the RCF bit in this channel cannot be set. When this bit is set, the RCF bit will be set by a miscompare between the OFLAG of this channel and the OFLAG of its redundant adjacent channel which will cause the output of this channel to go inactive (logic 0 prior to consideration of the OPS bit). 1 Enable redundant channel checking. 0 Disable redundant channel checking.
2 INPUT	External input signal This read only bit reflects the current state of the signal selected via SECSRC after application of the SIPS bit and filtering.
3 VAL	Forced OFLAG Value This bit determines the value of the OFLAG output signal when a software triggered FORCE command occurs.
4 FORCE	Force the OFLAG output This write only bit forces the current value of the VAL bit to be written to the OFLAG output. This bit always reads as a zero. The VAL and FORCE bits can be written simultaneously in a single write operation. Write to the FORCE bit only if OUTMODE is 0000 (software controlled). Setting this bit while the OUTMODE is a different value may yield unpredictable results

Table 593. Control Register 2n (CTRL2n) field descriptions (continued)

Field	Description
5 COFRC	Co-channel OFLAG Force This bit enables the compare from another channel within the module to force the state of this counter's OFLAG output signal. 1 Other channels may force the OFLAG of this channel. 0 Other channels cannot force the OFLAG of this channel.
6:7 COINIT	Co-channel Initialization These bits enable another channel within the module to force the re-initialization of this channel when the other channel has an active compare event 00 Other channels cannot force re-initialization of this channel. 01 Other channels may force a re-initialization of this channel's counter using the LOAD register. 10 Other channels may force a re-initialization of this channel's counter with the CMPLD2 register when this channel is counting down or the CMPLD1 register when this channel is counting up. 11 Reserved
8 SIPS	Secondary Source Input Polarity Select This bit inverts the polarity of the signal selected by the SECSRC bits. 1 Inverted polarity 0 True polarity
9 PIPS	Primary Source Input Polarity Select This bit inverts the polarity of the signal selected by the PRISRC bits. This only applies if the signal selected by PRISRC is not the prescaled IP Bus clock. 1 Inverted polarity 0 True polarity
10 OPS	Output Polarity Select This bit inverts the OFLAG output signal polarity. 1 Inverted polarity 0 True polarity
11 MSTR	Master Mode This bit enables the compare function's output to be broadcasted to the other channels in the module. The compare signal then can be used to reinitialize the other counters and/or force their OFLAG signal outputs. 1 Enable broadcast of compare events from this channel. 0 Disable broadcast of compare events from this channel.

Table 593. Control Register 2n (CTRL2n) field descriptions (continued)

Field	Description
12:15 OUTMODE	Output Mode These bits determine the mode of operation for the OFLAG output signal. 0000 Software controlled 0001 Clear OFLAG output on successful compare (COMP1 or COMP2) 0010 Set OFLAG output on successful compare (COMP1 or COMP2) 0011 Toggle OFLAG output on successful compare (COMP1 or COMP2) 0100 Toggle OFLAG output using alternating compare registers 0101 Set on compare with COMP1, cleared on secondary source input edge 0110 Set on compare with COMP2, cleared on secondary source input edge 0111 Set on compare, cleared on counter roll-over 1000 Set on successful compare on COMP1, clear on successful compare on COMP2 1001 Asserted while counter is active, cleared when counter is stopped. 1010 Asserted when counting up, cleared when counting down. 1011 Reserved 1100 Reserved 1101 Reserved 1110 Reserved 1111 Enable gated clock output while counter is active

38.4.4.10 Control Register 3n (CTRL3n)

Offset: 0x0012 + n*0x20 (n=0 to 5) Access: R/W

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	STPEN	ROC		0	1	1	1	1	C2FCNT			C1FCNT			DBGEN	
W																
Reset	0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0

Figure 536. Control Register 3n (CTRL3n)

Table 594. Control Register 3n (CTRL3n) field descriptions

Field	Description
0 STPEN	Stop Actions Enable This bit allows the tri-stating of the timer output during stop mode. 1 Output enable is disabled during stop mode. 0 Output enable is unaffected by stop mode.
1:2 ROC	Reload on Capture These bits enable the capture function to cause the counter to be reloaded from the LOAD register. 00 Do not reload the counter on a capture event. 01 Reload the counter on a capture 1 event. 10 Reload the counter on a capture 2 event. 11 Reload the counter on both a capture 1 event and a capture 2 event.
8:10 C2FCNT	CAPT2 FIFO Word Count This field reflects the number of words in the CAPT2 FIFO

Table 594. Control Register 3n (CTRL3n) field descriptions (continued)

Field	Description
11:13 C1FCNT	CAPT1 FIFO Word Count This field reflects the number of words in the CAPT1 FIFO.
14:15 DBGEN	Debug Actions Enable These bits allow the counter channel to perform certain actions in response to the chip entering debug mode. 00 Continue with normal operation during debug mode. (default) 01 Halt channel counter during debug mode. 10 Force OFLAG to logic 0 (prior to consideration of the OPS bit) during debug mode. 11 Both halt counter and force OFLAG to 0 during debug mode.

38.4.4.11 Status Register n (STSn)

Offset: 0x0014 + n*0x20 (n=0 to 5) Access: R/W

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	WDF	RCF	ICF2	ICF1	IEHF	IELF	TOF	TCF2	TCF1	TCF
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 537. Status Register n (STSn)

Table 595. Status Register n (STSn) field descriptions

Field	Description
6 WDF	Watchdog Time-out Flag This bit is set when the watchdog times out by counting down to zero. The watchdog must be enabled for time-out to occur and channel 0 must be in quadrature decode count mode (CNTMODE = 100). This bit is cleared by writing a 1 to this bit position after either writing a non-zero value to WDTOL and/or WDTOH or exiting quadrature decode counting mode. This bit is only in channel 0. If the watchdog timer is not implemented, then this bit is read-only zero.
7 RCF	Redundant Channel Flag This bit is set when there is a miscompare between this channel's OFLAG value and the OFLAG value of the corresponding redundant channel. Corresponding channels are grouped together in the following pairs: 0 and 1, 2 and 3, 4 and 5. This bit can only be set if the RDNT bit is set. This bit is cleared by writing a 1 to this bit position. This bit is only in even channels (0, 2, 4).
8 ICF2	Input Capture 2 Flag This bit is set when an input capture event (as defined by CPT2MODE) occurs and the word count of the CAPT2 FIFO exceeds the value of the CFWM field. This bit is cleared by writing a one to this bit position if ICF2DE is clear (no DMA) or it is cleared automatically by the DMA access if ICF2DE is set (DMA)
9 ICF1	Input Capture 1 Flag This bit is set when an input capture event (as defined by CPT1MODE) occurs and the word count of the CAPT1 FIFO exceeds the value of the CFWM field. This bit is cleared by writing a one to this bit position if ICF1DE is clear (no DMA) or it is cleared automatically by the DMA access if ICF1DE is set (DMA).

Table 595. Status Register n (STSn) field descriptions (continued)

Field	Description
10 IEHF	Input Edge High Flag This bit is set when a positive input transition occurs (on an input selected by SECSRC) while the counter is enabled. This bit is cleared by writing a one to this bit position.
11 IELF	Input Edge Low Flag This bit is set when a negative input transition occurs (on an input selected by SECSRC) while the counter is enabled. This bit is cleared by writing a one to this bit position.
12 TOF	Timer Overflow Flag This bit is set when the counter rolls over its maximum value 0xFFFF or 0x0000 (depending on count direction). This bit is cleared by writing a one to this bit location.
13 TCF2	Timer Compare 2 Flag This bit is set when a successful compare occurs with COMP2. This bit is cleared by writing a one to this bit location.
14 TCF1	Timer Compare 1 Flag This bit is set when a successful compare occurs with COMP1. This bit is cleared by writing a one to this bit location.
15 TCF	Timer Compare Flag This bit is set when a successful compare occurs. This bit is cleared by writing a one to this bit location.

38.4.4.12 Interrupt and DMA Enable Register n (INTDMAn)

Offset: 0x0016 + n*0x20 (n=0 to 5) Access: R/W

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	ICF2DE	ICF1DE	CMPLD2DE	CMPLD1DE	0	0	WDFIE	RCFIE	ICF2IE	ICF1IE	IEHFIE	IELFIE	TOFIE	TCF2IE	TCF1IE	TCFIE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 538. Interrupt and DMA Enable Register n (INTDMAn)

Table 596. Interrupt and DMA Enable Register n (INTDMAn) field descriptions

Field	Description
0 ICF2DE	Input Capture 2 Flag DMA Enable Setting this bit enables DMA read requests for CAPT2 when the ICF2 bit is set. Do not set both this bit and the ICF2IE bit.
1 ICF1DE	Input Capture 1 Flag DMA Enable Setting this bit enables DMA read requests for CAPT1 when the ICF1 bit is set. Do not set both this bit and the ICF1IE bit.

Table 596. Interrupt and DMA Enable Register n (INTDMAn) field descriptions (continued)

Field	Description
2 CMPLD2DE	Comparator Load Register 2 Flag DMA Enable Setting this bit enables DMA write requests to the CMPLD2 register whenever data is transferred out of the CMPLD2 reg into either the CNTR, COMP1, or COMP2 registers.
3 CMPLD1DE	Comparator Load Register 1 Flag DMA Enable Setting this bit enables DMA write requests to the CMPLD1 register whenever data is transferred out of the CMPLD1 reg into either the CNTR, COMP1, or COMP2 registers.
6 WDFIE	Watchdog Flag Interrupt Enable Setting this bit enables interrupts when the WDF bit is set. This bit is only in channel 0. If the watchdog timer is not implemented, then this bit serves no purpose.
7 RCFIE	Redundant Channel Flag Interrupt Enable Setting this bit enables interrupts when the RCF bit is set. This bit is only in even channels (0, 2, and 4).
8 ICF2IE	Input Capture 2 Flag Interrupt Enable Setting this bit enables interrupts when the ICF2 bit is set. Do not set both this bit and the ICF2DE bit.
9 ICF1IE	Input Capture 1 Flag Interrupt Enable Setting this bit enables interrupts when the ICF1 bit is set. Do not set both this bit and the ICF1DE bit.
10 IEHFIE	Input Edge High Flag Interrupt Enable Setting this bit enables interrupts when the IEHF bit is set.
11 IELFIE	Input Edge Low Flag Interrupt Enable Setting this bit enables interrupts when the IELF bit is set.
12 TOFIE	Timer Overflow Flag Interrupt Enable Setting this bit enables interrupts when the TOF bit is set.
13 TCF2IE	Timer Compare 2 Flag Interrupt Enable Setting this bit enables interrupts when the TCF2 bit is set.
14 TCF1IE	Timer Compare 1 Flag Interrupt Enable Setting this bit enables interrupts when the TCF1 bit is set.
15 TCFIE	Timer Compare Flag Interrupt Enable Setting this bit enables interrupts when the TCF bit is set.

38.4.4.13 Comparator Load Register 1n (CMPLD1n)

This read/write register is the preload value for the COMP1 register. This register can also be used to load into the CNTR register. This register is not byte accessible. More information on the use of this register can be found in *Section 38.3.2.14: Usage of Compare Load Registers*.

Offset: 0x0018 + n*0x20 (n=0 to 5) Access: R/W

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	CMPLD1															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 539. Comparator Load 1n (CMPLD1n)

Table 597. CMPLD1n field descriptions

Field	Description
0:15 CMPLD1	Comparator Load Register 1

38.4.4.14 Comparator Load Register 2n (CMPLD2n)

This read/write register is the preload value for the COMP2 register. This register can also be used to load into the CNTR register. This register is not byte accessible. More information on the use of this register can be found in *Section 38.3.2.14: Usage of Compare Load Registers*.

Offset: 0x001A + n*0x20 (n=0 to 5) Access: R/W

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	CMPLD2															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 540. Comparator Load 2n (CMPLD2n)

Table 598. CMPLD2n field descriptions

Field	Description
0:15 CMPLD2	Comparator Load Register 2

38.4.4.15 Compare and Capture Control Register n (CCCTRLn)

Offset: 0x001C + n*0x20 (n=0 to 5) Access: R/W

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	CLC2			CLC1			CMPMODE		CPT2MODE		CPT1MODE		CFWM		ONESHOT	ARM
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 541. Compare and Capture Control Register n (CCCTRLn)

Table 599. Compare and Capture Control Register n (CCCTRLn) field descriptions

Field	Description
0:2 CLC2	Compare Load Control 2 These bits control when COMP2 is preloaded. It also controls the loading of CNTR. 000 Never preload 001 Reserved 010 Load COMP2 with CMPLD1 upon successful compare with the value in COMP1. 011 Load COMP2 with CMPLD1 upon successful compare with the value in COMP2. 100 Load COMP2 with CMPLD2 upon successful compare with the value in COMP1. 101 Load COMP2 with CMPLD2 upon successful compare with the value in COMP2. 110 Load CNTR with CMPLD2 upon successful compare with the value in COMP1. 111 Load CNTR with CMPLD2 upon successful compare with the value in COMP2.
3:5 CLC1	Compare Load Control 1 These bits control when COMP1 is preloaded. It also controls the loading of CNTR 000 Never preload 001 Reserved 010 Load COMP1 with CMPLD1 upon successful compare with the value in COMP1. 011 Load COMP1 with CMPLD1 upon successful compare with the value in COMP2. 100 Load COMP1 with CMPLD2 upon successful compare with the value in COMP1. 101 Load COMP1 with CMPLD2 upon successful compare with the value in COMP2. 110 Load CNTR with CMPLD1 upon successful compare with the value in COMP1. 111 Load CNTR with CMPLD1 upon successful compare with the value in COMP2.
6:7 CMPMODE	Compare Mode These bits control when the COMP1 and COMP2 registers are used in regards to the counting direction. 00 COMP1 register is used when the counter is counting up. COMP2 register is used when the counter is counting up. 01 COMP1 register is used when the counter is counting down. COMP2 register is used when the counter is counting up. 10 COMP1 register is used when the counter is counting up. COMP2 register is used when the counter is counting down. 11 COMP1 register is used when the counter is counting down. COMP2 register is used when the counter is counting down.
8:9 CPT2MODE	Capture 2 Mode Control These bits control the operation of the CAPT2 register as well as the operation of the ICF2 flag by defining which input edges cause a capture event. The input source is the secondary count source. 00 Disabled 01 Capture falling edges 10 Capture rising edges 11 Capture any edge
10:11 CPT1MODE	Capture 1 Mode Control These bits control the operation of the CAPT1 register as well as the operation of the ICF1 flag by defining which input edges cause a capture event. The input source is the secondary count source. 00 Disabled 01 Capture falling edges 10 Capture rising edges 11 Capture any edge

Table 599. Compare and Capture Control Register n (CCCTRLn) field descriptions (continued)

Field	Description
12:13 CFWM	Capture FIFO Watermark This field represents the Watermark level for the CAPT1 and CAPT2 FIFOs. The capture flags, ICF1 and ICF2, will not be set until the word count of the corresponding FIFO is greater than this Watermark level
14 ONESHOT	One Shot Capture Mode This bit selects between free running and one shot mode for the input capture circuitry. 1 One shot mode is selected - If both capture circuits are enabled, then capture circuit 1 is armed first after the ARM bit is set. Once a capture occurs, capture circuit 1 is disarmed and capture circuit 2 is armed. After capture circuit 2 performs a capture, it is disarmed and the ARM bit is cleared. No further captures will be performed until the ARM bit is set again. If only one of the capture circuits is enabled, then a single capture will occur on the enabled capture circuit and the ARM bit is then cleared. 0 Free running mode is selected - If both capture circuits are enabled, then capture circuit 1 is armed first after the ARM bit is set. Once a capture occurs, capture circuit 1 is disarmed and capture circuit 2 is armed. After capture circuit 2 performs a capture, it is disarmed and capture circuit 1 is re-armed. The process continues indefinitely. If only one of the capture circuits is enabled, then captures continue indefinitely on the enabled capture circuit.
15 ARM	Arm Capture Setting this bit high starts the input capture process. This bit can be cleared at any time to disable input capture operation. This bit is self cleared when in one shot mode and the enabled capture circuit(s) has had a capture event(s). 1 Input capture operation as specified by the CPT1MODE and CPT2MODE bits is enabled. 0 Input capture operation is disabled.

38.4.4.16 Input Filter Register n (FILTn)

The FILT register Programs the values for the filtering of the corresponding input without regard for the fact that any eTimer channel can use the input as a count source. The FILT register is also used to control the filtering of the corresponding auxilliary input.

Offset: 0x001E + n*0x20 (n=0 to 5) Access: R/W

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	FILT_CNT			FILT_PER							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 542. Input Filter Register n (FILTn)

Table 600. Input Filter Register n (FILTn) field descriptions

Field	Description
5:7 FILT_CNT	Input Filter Sample Count These bits represent the number of consecutive samples that must agree prior to the input filter accepting an input transition. A value of 0 represents 3 samples. A value of 7 represents 10 samples. The value of FILT_CNT affects the input latency as described in *Section 38.4.4.16.1: Input Filter Considerations*.
8:15 FILT_PER	Input Filter Sample Period These bits represent the sampling period (in IPBus clock cycles) of the eTimer input signal. Each input is sampled multiple times at the rate specified by FILT_PER. If FILT_PER is 0x00 (default), then the input filter is bypassed. The value of FILT_PER affects the input latency as described in *Section 38.4.4.16.1: Input Filter Considerations*.

38.4.4.16.1 Input Filter Considerations

The FILT_PER value should be set such that the sampling period is larger the period of the expected noise. This way a noise spike will only corrupt one sample. The FILT_CNT value should be chosen to reduce the probability of noisy samples causing an incorrect transition to be recognized. The probability of an incorrect transition is defined as the probability of an incorrect sample raised to the FILT_CNT + 3 power.

The values of FILT_PER and FILT_CNT must also be traded off against the desire for minimal latency in recognizing input transitions. Turning on the input filter (setting FILT_PER to a non-zero value) introduces a latency of: (((FILT_CNT + 3) x FILT_PER) + 2) IPBus clock periods.

38.4.5 Watchdog Timer Registers

Note: A watchdog timer may or may not be implemented on each eTimer module instance.

The base address of the Watchdog Timer registers is equal to the base address of the eTimer plus an offset of 0x0100.

38.4.5.1 Watchdog Time-out Low Word Register (WDTOL)

Offset: 0x100 Access: R/W

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	WDTOL															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 543. Watchdog Time-out Low Word Register (WDTOL)

Table 601. WDTOL field descriptions

Field	Description
0:15 WDTOL	Watchdog Time-out Low Word Registers WDTOL and WDTOH are combined to form the 32-bit time-out count for the Timer watchdog function. This time-out count is used to monitor for inactivity on the inputs when channel 0 is in the quadrature decode count mode. The watchdog function is enabled whenever WDTO contains a non-zero value (although actual counting only occurs if channel 0 is in quadrature decode counting mode). The watchdog time-out down counter is loaded whenever WDTOH is written. These registers are not byte accessible. See *Section 38.3.3.5: Watchdog Timer,* for more information on the use of the watchdog timer.

38.4.5.2 Watchdog Time-out High Word Register (WDTOH)

Offset: 0x102 Access: R/W

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	WDTOH															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 544. Watchdog Time-out High Word Register (WDTOH)

Table 602. WDTOH field descriptions

Field	Description
0:15 WDTOH	Watchdog Time-out High Word Registers WDTOL and WDTOH are combined to form the 32-bit time-out count for the Timer watchdog function. This time-out count is used to monitor for inactivity on the inputs when channel 0 is in the quadrature decode count mode. The watchdog function is enabled whenever WDTO contains a non-zero value (although actual counting only occurs if channel 0 is in quadrature decode counting mode). The watchdog time-out down counter is loaded whenever WDTOH is written. These registers are not byte accessible. See *Section 38.3.3.5: Watchdog Timer,* for more information on the use of the watchdog timer.

38.4.6 Configuration Registers

The base address of the configuration registers is equal to the base address of the eTimer plus an offset of 0x010C.

38.4.6.1 Channel Enable Register (ENBL)

Offset: 0x10C Access: R/W

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	ENBL					
W																
Reset	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1

Figure 545. Channel Enable Register (ENBL)

Table 605. Channel Enable Register (ENBL) field descriptions

Field	Description
10:15 ENBL	Timer Channel Enable These bits enable the prescaler (if it is being used) and counter in each channel. Multiple ENBL bits can be set at the same time to synchronize the start of separate channels. If an ENBL bit is set, then the corresponding channel will start counting as soon as the CNTMODE field has a value other than '000'. When an ENBL bit is cleared, the corresponding channel maintains its current value. 1 Timer channel is enabled. (default) 0 Timer channel is disabled.

38.4.6.2 DMA Request 0 Select Registers (DREQ0)

Offset: 0x110 Access: R/W

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DREQ0_EN	0	0	0	0	0	0	0	0	0	0	DREQ0				
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 546. DMA Request 0 Select Register (DREQ0)

Table 606. DMA Request 0 Select Register (DREQ0) field descriptions

Field	Description
0 DREQ0_EN	DMA Request Enable Use these bits to enable each of the four module-level DMA request outputs. Program the DREQ fields prior to setting the corresponding enable bit. Clearing this enable bit will remove the request but will not clear the flag that is causing the request. 1 DMA request enabled. 0 DMA request disabled
11:15 DREQ0	DMA Request Select Use these fields to select which DMA request source will be muxed onto one of the four module-level DMA request outputs. Make sure each of the DREQ registers is programmed with a different value else a single DMA source will cause multiple DMA requests. Enable a DMA request in the channel specific INTDMA register after the DREQ registers are programmed. 00000 Channel 0 CAPT1 DMA read request 00001 Channel 0 CAPT2 DMA read request 00010 Channel 0 CMPLD1 DMA write request 00011 Channel 0 CMPLD2 DMA write request 00100 Channel 1 CAPT1 DMA read request 00101 Channel 1 CAPT2 DMA read request 00110 Channel 1 CMPLD1 DMA write request 00111 Channel 1 CMPLD2 DMA write request 01000 Channel 2 CAPT1 DMA read request 01001 Channel 2 CAPT2 DMA read request 01010 Channel 2 CMPLD1 DMA write request 01011 Channel 2 CMPLD2 DMA write request 01100 Channel 3 CAPT1 DMA read request 01101 Channel 3 CAPT2 DMA read request 01110 Channel 3 CMPLD1 DMA write request 01111 Channel 3 CMPLD2 DMA write request 10000 Channel 4 CAPT1 DMA read request 10001 Channel 4 CAPT2 DMA read request 10010 Channel 4 CMPLD1 DMA write request 10011 Channel 4 CMPLD2 DMA write request 10100 Channel 5 CAPT1 DMA read request 10101 Channel 5 CAPT2 DMA read request 10110 Channel 5 CMPLD1 DMA write request 10111 Channel 5 CMPLD2 DMA write request 11000 Reserved 11001 Reserved 11010 Reserved 11011 Reserved 11100 Reserved 11101 Reserved 11110 Reserved 11111 Reserved

38.4.6.3 DMA Request 1 Select Registers (DREQ1)

Offset: 0x112 Access: R/W

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DREQ1_EN	0	0	0	0	0	0	0	0	0	0	DREQ1				
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 547. DMA Request 1 Select Register (DREQ1)

Table 607. DMA Request 1 Select Register (DREQ1) field descriptions

Field	Description
0 DREQ1_EN	DMA Request Enable Use these bits to enable each of the four module-level DMA request outputs. Program the DREQ fields prior to setting the corresponding enable bit. Clearing this enable bit will remove the request but will not clear the flag that is causing the request. 1 DMA request enabled. 0 DMA request disabled

Table 607. DMA Request 1 Select Register (DREQ1) field descriptions

Field	Description
11:15 DREQ1	DMA Request Select Use these fields to select which DMA request source will be muxed onto one of the four module-level DMA request outputs. Make sure each of the DREQ registers is programmed with a different value else a single DMA source will cause multiple DMA requests. Enable a DMA request in the channel specific INTDMA register after the DREQ registers are programmed. 00000 Channel 0 CAPT1 DMA read request 00001 Channel 0 CAPT2 DMA read request 00010 Channel 0 CMPLD1 DMA write request 00011 Channel 0 CMPLD2 DMA write request 00100 Channel 1 CAPT1 DMA read request 00101 Channel 1 CAPT2 DMA read request 00110 Channel 1 CMPLD1 DMA write request 00111 Channel 1 CMPLD2 DMA write request 01000 Channel 2 CAPT1 DMA read request 01001 Channel 2 CAPT2 DMA read request 01010 Channel 2 CMPLD1 DMA write request 01011 Channel 2 CMPLD2 DMA write request 01100 Channel 3 CAPT1 DMA read request 01101 Channel 3 CAPT2 DMA read request 01110 Channel 3 CMPLD1 DMA write request 01111 Channel 3 CMPLD2 DMA write request 10000 Channel 4 CAPT1 DMA read request 10001 Channel 4 CAPT2 DMA read request 10010 Channel 4 CMPLD1 DMA write request 10011 Channel 4 CMPLD2 DMA write request 10100 Channel 5 CAPT1 DMA read request 10101 Channel 5 CAPT2 DMA read request 10110 Channel 5 CMPLD1 DMA write request 10111 Channel 5 CMPLD2 DMA write request 11000 Reserved 11001 Reserved 11010 Reserved 11011 Reserved 11100 Reserved 11101 Reserved 11110 Reserved 11111 Reserved

38.5 Resets

The eTimer module can only be reset by a full system reset. This forces all registers to their reset state and clears the OFLAG signals if they are asserted. The counters will be turned off until the settings in the CTRL registers are changed.

38.6 Clocks

Each timer channel receives its own copy of the peripheral bus clock. This allows for better power management by turning off the clock to unused channels. The watchdog timer (if implemented) has its own version of this clock. Each bit of the ENBL register uses the clock for that corresponding channel. The DREQn registers use the clock for channel, 0 but this clock can be turned off after these registers have been programmed.

38.7 Interrupts

The following can generate interrupts within the eTimer.

- Watchdog timer (if implemented)
- Each channel can generate an interrupt from several sources

The interrupt service routine (ISR) must check the related interrupt enables and interrupt flags to determine the actual cause of the interrupt.

Table 608. Interrupt Summary

Core Interrupt	Interrupt Flag	Interrupt Enable	Name	Description
Channels 0-*n*	TCF	TCFIE	Compare interrupt	Compare of counter and related compare register
	TCF1	TCF1IE	Compare 1 interrupt	Compare of the counter and COMP1 register
	TCF2	TCF2IE	Compare 2 interrupt	Compare of the counter and COMP2 register
	TOF	TOFIE	Overflow interrupt	Generated on counter roll-over or roll-under
	IELF	IELFIE	Input Low Edge interrupt	Falling edge of the secondary input signal
	IEHF	IEHFIE	Input High Edge interrupt	Rising edge of the secondary input signal
	ICF1	ICF1IE	Input Capture 1 interrupt	Input capture event for CAPT1
	ICF2	ICF2IE	Input Capture 2 interrupt	Input capture event for CAPT2
Watchdog (1)	WDF	WDFIE	Watchdog time-out interrupt	Watchdog has timed out
Redundant Channel Checking	RCF	RCFIE	Redundant Channel Fault interrupt	Miscompare with redundant channel

1. If applicable. Not all instantiations of eTimer implement a watchdog timer.

38.8 DMA

Each channel can request a DMA read access for each of the capture registers and a DMA write request for each of the compare preload registers. The DREQ registers select amongst these 32 DMA request sources to generate the 4 top level DMA request outputs.

Table 609. DMA Summary

DMA Request	DMA Enable	Name	Description
Channels 0-*n*	ICF1DE	CAPT1 read request	CAPT1 contains a value
	ICF2DE	CAPT2 read request	CAPT2 contains a value
	CMPLD1DE	CMPLD1 write request	CMPLD1 needs an update
	CMPLD2DE	CMPLD2 write request	CMPLD2 needs an update

39 System Timer Module (STM)

39.1 Introduction

This section provides an overview, list of features, and modes of operation for the STM.

Note: For chip-specific implementation details of this module's instances, see the chip configuration information.

39.1.1 Overview

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

39.1.2 Features

The STM has the following features:

- One 32-bit up counter with 8-bit prescaler
- Four 32-bit compare channels
- Independent interrupt source for each channel
- Counter can be stopped in debug mode

39.1.3 Modes of operation

The STM supports two device modes of operation: Normal and Debug. When the STM is enabled in Normal mode, its counter runs continuously. In Debug mode, operation of the counter is controlled by the FRZ bit in the STM Control Register (STM_CR). If STM_CR[FRZ] is set, the counter is stopped in Debug mode, otherwise it continues to run.

39.2 External signal description

The STM does not have any external interface signals.

39.3 Memory map and register definition

The STM programming model includes a group of 32-bit registers: a module control register, a counter value register, and three registers for each channel.

The STM registers can only be accessed using 32-bit (word) accesses. Attempted references using a different size or to a reserved address generates a bus error termination.

39.3.1 Memory map

The STM memory map is shown in *Table 610*.

Table 610. STM memory map

Address offset	Register	Location
0x0000	STM Control Register (STM_CR)	*Section 39.3.2.1*
0x0004	STM Counter Value (STM_CNT)	*Section 39.3.2.2*
0x0008–0x000B	Reserved	
0x000C–0x000F	Reserved	
STM channel registers per channel (n=0 to 3)		
0x0010 + n*0x10	STM Channel n Control Register (STM_CCRn)	*Section 39.3.2.3*
0x0014 + n*0x10	STM Channel n Interrupt Register (STM_CIRn)	*Section 39.3.2.4*
0x0018 + n*0x10	STM Channel n Compare Register (STM_CMPn)	*Section 39.3.2.5*
0x001C + n*0x10	Reserved	
0x004C–0x3FFF	Reserved	

39.3.2 Register descriptions

39.3.2.1 STM Control Register (STM_CR)

The STM Control Register (STM_CR) includes the prescale value, freeze control and timer enable bits.

Offset: 0x000 Access: Read/Write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	CPS								0	0	0	0	0	0	FRZ	TEN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 548. STM Control Register (STM_CR)

Table 611. STM_CR field descriptions

Field	Description
16:23 CPS	Counter Prescaler. Selects the clock divide value for the prescaler (1 - 256). 0x00 Divide PBRIDGE clock by 1 0x01 Divide PBRIDGE clock by 2 ... 0xFF Divide PBRIDGE clock by 256
30 FRZ	Freeze. Allows the timer counter to be stopped when the device enters debug mode. **Note:** When the microcomputer enters debug mode, the STM is notified and uses the FRZ bit to determine counter mode. 0 STM counter continues to run in debug mode 1 STM counter is stopped in debug mode
31 TEN	Timer Counter Enabled. 0 Counter is disabled 1 Counter is enabled

39.3.2.2 STM Count Register (STM_CNT)

The STM Count Register (STM_CNT) holds the timer count value.

Offset: 0x004 Access: Read/Write

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

Figure 549. STM Count Register (STM_CNT)

Table 612. STM_CNT field descriptions

Field	Description
0:31 CNT	Timer count value used as the time base for all channels. When enabled, the counter increments at the rate of the PBRIDGE clock divided by the prescale value.

39.3.2.3 STM Channel Control Register (STM_CCR*n*)

The STM Channel Control Register (STM_CCR*n*) has the enable bit for channel n of the timer.

Offset: 0x10 + *n**0x10 (*n* = 0 to 3) Access: Read/Write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	CEN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 550. STM Channel Control Register (STM_CCR*n*)

Table 613. STM_CCR*n* field descriptions

Field	Description
31 CEN	Channel Enable. 0 The channel is disabled 1 The channel is enabled

39.3.2.4 STM Channel Interrupt Register (STM_CIR*n*)

The STM Channel Interrupt Register (STM_CIR*n*) has the interrupt flag for channel n of the timer.

Offset: 0x14 + *n**0x10 (*n* = 0 to 3) Access: Read/Write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	CIF
W																w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 551. STM Channel Interrupt Register (STM_CIR*n*)

Table 614. STM_CIR*n* field descriptions

Field	Description
31 CIF	Channel Interrupt Flag 0 No interrupt request 1 Interrupt request due to a match on the channel

39.3.2.5 STM Channel Compare Register (STM_CMP*n*)

The STM channel compare register (STM_CMPn) holds the compare value for channel n.

Offset: 0x18 + *n**0x10 (*n* = 0 to 3) Access: Read/Write

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

Figure 552. STM Channel Compare Register (STM_CMP*n*)

Table 615. STM_CMP*n* field descriptions

Field	Description
0:31 CMP	Compare value for channel n. If the STM_CCR*n*[CEN] bit is set and the STM_CMP*n* register matches the STM_CNT register, a channel interrupt request is generated and the STM_CIR*n*[CIF] bit is set.

39.4 Functional description

The System Timer Module (STM) is a 32-bit timer designed to support commonly required system and application software timing functions. The STM includes a 32-bit up counter and four 32-bit compare channels with a separate interrupt source for each channel.

The STM has one 32-bit up counter (STM_CNT) that is used as the time base for all channels. When enabled, the counter increments at the PBRIDGE clock frequency divided by a prescale value. The STM_CR[CPS] field sets the divider to any value in the range from 1 to 256. The counter is enabled with the STM_CR[TEN] bit. When enabled in normal mode the counter continuously increments. When enabled in debug mode the counter operation is controlled by the STM_CR[FRZ] bit. When enabled in normal mode the counter continuously increments. When enabled in debug mode, the counter operation is controlled by the STM_CR[FRZ] bit. When the STM_CR[FRZ] bit is set, the counter is stopped in debug mode, otherwise it continues to run in debug mode. The counter rolls over at 0xFFFF_FFFF to 0x0000_0000 with no restrictions at this boundary.

The STM has four identical compare channels. Each channel includes a channel control register (STM_CCR*n*), a channel interrupt register (STM_CIR*n*) and a channel compare register (STM_CMP*n*). The channel is enabled by setting the STM_CCR*n*[CEN] bit. When enabled, the channel will set the STM_CIR*n*[CIF] bit and generate an interrupt request when the channel compare register matches the timer counter. The interrupt request is cleared by writing a 1 to the STM_CIR*n*[CIF] bit. A write of 0 to the STM_CIR*n*[CIF] bit has no effect.

Note: The STM counter does not advance when the PBRIDGE clock is stopped.

40 Software Watchdog Timer (SWT)

40.1 Introduction

This section provides an overview, list of features, and modes of operation for the SWT.

Note: For chip-specific implementation details of this module's instances, see the chip configuration information.

40.1.1 Overview

The Software Watchdog Timer (SWT) is a peripheral module that can prevent system lockup in situations such as software getting trapped in a loop or if a bus transaction fails to terminate. When enabled, the SWT requires periodic execution of a watchdog servicing operation. The servicing operation resets the timer to a specified time-out period. If this servicing action does not occur before the timer expires the SWT generates an interrupt or hardware reset. The SWT can be configured to generate a reset or interrupt on an initial time-out. A reset is always generated on a second consecutive time-out.

40.1.2 Features

The SWT has the following features:

- 32-bit time-out register to set the time-out period
- Programmable selection of window mode or regular servicing
- Programmable selection of reset or interrupt on an initial time-out
- Programmable selection of the servicing mode
- Master access protection
- Hard and soft configuration lock bits
- Reset configuration inputs allow timer to be enabled out of reset

40.1.3 Modes of operation

The SWT supports three device modes of operation: normal, debug and stop. When the SWT is enabled in normal mode, its counter runs continuously. In debug mode, operation of the counter is controlled by the FRZ bit in the SWT_CR *Section 40.3.1.1: SWT Control Register (SWT_CR)*. If the FRZ bit is set, the counter is stopped in debug mode, otherwise it continues to run. In stop mode, operation of the counter is controlled by the STP bit of the SWT_CR register. See *Section 40.3.1.1: SWT Control Register (SWT_CR)*. If the STP bit is set, the counter is stopped in stop mode, otherwise it continues to run.

40.2 External signal description

The SWT module does not have any external interface signals.

40.3 Memory map and register definition

The SWT programming model has seven 32-bit registers. The programming model can only be accessed using 32-bit (word) accesses. References using a different size are invalid.

Other types of invalid accesses include: writes to read-only registers, incorrect values written to the service register when enabled, accesses to reserved addresses and accesses by masters without permission. If the RIA bit in the SWT_CR is set then the SWT generates a system reset on an invalid access, otherwise a bus error is generated. If either the HLK or SLK bits in the SWT_CR are set, then the SWT_CR, SWT_TO, SWT_WN, and SWT_SK registers are read-only.

The SWT memory map is shown in *Table 616*.

Table 616. SWT memory map

Offset	Register	Location
0x0000	SWT Control Register (SWT_CR)	*Section 40.3.1.1*
0x0004	SWT Interrupt Register (SWT_IR)	*Section 40.3.1.2*
0x0008	SWT Time-Out Register (SWT_TO)	*Section 40.3.1.3*
0x000C	SWT Window Register (SWT_WN)	*Section 40.3.1.4*
0x0010	SWT Service Register (SWT_SR)	*Section 40.3.1.5*
0x0014	SWT Counter Output Register (SWT_CO)	*Section 40.3.1.6*
0x0018	SWT Service Key Register (SWT_SK)	*Section 40.3.1.7*
0x001C–0x3FFF	Reserved	

40.3.1 Register descriptions

The following sections detail the individual registers within the SWT programming model.

Note: See the memory map for the base address of SWT registers.

40.3.1.1 SWT Control Register (SWT_CR)

The SWT_CR contains fields for configuring and controlling the SWT. This register is read-only if either the SWT_CR[HLK] or SWT_CR[SLK] bits are set.

Offset: 0x0000 Access: Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	MAP0	MAP1	MAP2	MAP3	MAP4	MAP5	MAP6	MAP7	0	0	0	0	0	0	0	0
W																
Reset(1)	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	SMD		RIA	WND	ITR	HLK	SLK	1	STP	FRZ	WEN
W																
Reset(1)	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*

1. The reset value for the SWT_CR is device specific. See the device configuration chapter.

Figure 553. SWT Control Register (SWT_CR)

Table 617. SWT_CR field descriptions

Field	Description
0:7 MAP[0:7]	Master Access Protection for Master n The platform bus master assignments are device specific. Not all MAPn fields are implemented. See the device configuration chapter for master ports that are implemented on the SMPU. 0 Access for the master is not enabled 1 Access for the master is enabled
21:22 SMD	Service Mode 00 Fixed Service Sequence, the watchdog is serviced by writing the fixed sequence 0xA602, 0xB480 to the SWT_SR. 01 Keyed Service Sequence, the watchdog is serviced by writing two pseudorandom key values to the SWT_SR. 10 Fixed Address Execution, the watchdog is serviced by executing code at the address loaded into the designated IAC register which can not be updated while the watchdog is enabled. Support for this mode is device and instance specific. 11 Incremental Address Execution, the watchdog is serviced by executing code at the address loaded into the designated IAC register which can be updated. Support for this mode is device and instance specific.
23 RIA	Reset on Invalid Access 0 Invalid access to the SWT generates a bus error 1 Invalid access to the SWT causes a system reset if WEN=1
24 WND	Window Mode 0 Regular mode, service sequence can be done at any time 1 Windowed mode, the service sequence is only valid when the down counter is less than the value in the SWT_WN register.
25 ITR	Interrupt Then Reset 0 Generate a reset on a time-out 1 Generate an interrupt on an initial time-out, reset on a second consecutive time-out
26 HLK	Hard Lock This bit is only cleared at reset. 0 SWT_CR, SWT_TO, SWT_WN and SWT_SK are read/write registers if SLK=0 1 SWT_CR, SWT_TO, SWT_WN and SWT_SK are read-only registers
27 SLK	Soft Lock This bit is cleared by writing the unlock sequence to the service register. 0 SWT_CR, SWT_TO SWT_WN and SWT_SK are read/write registers if HLK=0 1 SWT_CR, SWT_TO, SWT_WN and SWT_SK are read-only registers
29 STP	Stop Mode Control Allows the watchdog timer to be stopped when the device enters stop mode. 0 SWT counter continues to run in stop mode 1 SWT counter is stopped in stop mode
30 FRZ	Debug Mode Control Allows the watchdog timer to be stopped when the device enters debug mode. 0 SWT counter continues to run in debug mode 1 SWT counter is stopped in debug mode
31 WEN	Watchdog Enabled 0 SWT is disabled 1 SWT is enabled

40.3.1.2 SWT Interrupt Register (SWT_IR)

The SWT_IR contains the time-out interrupt flag.

Offset: 0x0004 Access: Read/Write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	TIF
W																w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 554. SWT Interrupt Register (SWT_IR)

Table 618. SWT_IR field descriptions

Field	Description
31 TIF	Time-out Interrupt Flag The flag and interrupt are cleared by writing a 1 to this bit. Writing a 0 has no effect. 0 No interrupt request. 1 Interrupt request due to an initial time-out.

40.3.1.3 SWT Time-Out Register (SWT_TO)

The SWT Time-Out (SWT_TO) register contains the 32-bit time-out period. The reset value for this register is device specific. This register is read-only if either the SWT_CR[HLK] or SWT_CR[SLK] bits are set.

Offset: 0x0008 Access: Read/Write

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

1. The reset value of the SWT_TO register is device specific. See the device configuration chapter.

Figure 555. SWT Time-Out Register (SWT_TO)

Table 619. SWT_TO field descriptions

Field	Description
0:31 WTO	Watchdog time-out period in clock cycles An internal 32-bit down counter is loaded with this value when the service sequence is written or when the SWT is enabled. The initial value is 0x0003_FDE0. The minimum value is 0x0000_0100.

40.3.1.4 SWT Window Register (SWT_WN)

The SWT Window (SWT_WN) register contains the 32-bit window start value. This register is cleared on reset. This register is read-only if either the SWT_CR[HLK] or SWT_CR[SLK] bits are set.

Offset: 0x000C Access: Read/Write

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

Figure 556. SWT Window Register (SWT_WN)

Table 620. SWT_WN field descriptions

Field	Description
0:31 WST	Window start value When window mode is enabled, the service sequence can only be written when the internal down counter is less than this value.

40.3.1.5 SWT Service Register (SWT_SR)

The SWT Service (SWT_SR) register is the target for service operation writes used to reset the watchdog timer.

Offset: 0x0010 Access: Write

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

Figure 557. SWT Service Register (SWT_SR)

Table 621. SWT_SR field descriptions

Field	Description
16:31 WSC	Watchdog Service Code This field is used to service the watchdog and to clear the soft lock bit (SWT_CR[SLK]). If the SWT_CR[SMD] = 01, two pseudorandom key values are written to service the watchdog, see *Section 40.3.1.1: SWT Control Register (SWT_CR)* for details. Otherwise, the sequence 0xA602 followed by 0xB480 is written to the WSC field. To clear the soft lock bit (SWT_CR[SLK]), the value 0xC520 followed by 0xD928 is written to the WSC field. When read, the WSC field always returns zero.

40.3.1.6 SWT Counter Output Register (SWT_CO)

The SWT Counter Output (SWT_CO) register is a read-only register that shows the value of the internal down counter when the SWT is disabled.

Offset: 0x0014 Access: Read-Only

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

1. SWT_CO will reflect the internal value of the counter when WEN=0, and will be read as 0x0000_0000 when WEN=1

Figure 558. SWT Counter Output Register (SWT_CO)

Table 622. SWT_CO field descriptions

Field	Description
0:31 CNT	Watchdog Count When the watchdog is disabled (SWT_CR[WEN]=0), this field shows the value of the internal down counter. When the watchdog is enabled (SWT_CR[WEN]=1), this field is cleared (the value is 0x0000_0000). Values in this field can lag behind the internal counter value for up to six system plus eight counter clock cycles. Therefore, the value read from this field immediately after disabling the watchdog may be higher than the actual value of the internal counter.

40.3.1.7 SWT Service Key Register (SWT_SK)

The SWT Service Key (SWT_SK) register holds the previous (or initial) service key value. This register is read-only if either the SWT_CR[HLK] or SWT_CR[SLK] bits are set.

Offset: 0x0018 Access: Read/Write

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

Figure 559. SWT Service Key Register (SWT_SK)

Table 623. SWT_SK field descriptions

Field	Description
16:31 SK	Service Key This field is the previous (or initial) service key value used in keyed service mode. If SWT_CR[SMD] = 01, the next key value to be written to the SWT_SR is (17*SK+3) mod 2^16.

40.4 Functional description

40.4.1 Introduction

The SWT is a 32-bit timer designed to enable the system to recover in situations such as software getting trapped in a loop or if a bus transaction fails to terminate. It includes:

- A control register (SWT_CR),
- An interrupt register (SWT_IR),
- A time-out register (SWT_TO),
- A window register (SWT_WN),
- A service register (SWT_SR),
- A counter output register (SWT_CO), and
- A service key register (SWT_SK).

Accesses to SWT registers occur with no peripheral bus wait states. (The peripheral bus bridge may add one or more system wait states.) However, due to synchronization logic in the SWT design, recognition of the service sequence or configuration changes may require up to three system plus seven counter clock cycles.

40.4.1.1 SWR_CR register

The SWT_CR register includes bits to enable the timer, set configuration options and lock configuration of the module. The watchdog is enabled by setting the SWT_CR[WEN] bit. The reset value of the SWT_CR[WEN] bit is device specific[(m)]. If the reset value of this bit is 1, the watchdog starts operation automatically after reset is released. Some devices can be configured to clear this bit automatically during the boot process.

40.4.1.2 SWT_TO register

The SWT_TO register holds the watchdog time-out period in clock cycles unless the value is less than 0x100 in which case the time-out period is set to 0x100.When the SWT is enabled, this time-out period is loaded into an internal 32-bit down counter each time a valid service operation is performed.The source used to drive the down counter and the reset value of the SWT_TO register is device specific, see the device configuration chapter.

40.4.1.3 SWT_CO register

The SWT_CO register shows the value of the down counter when the watchdog is disabled. When the watchdog is enabled this register is cleared. The value shown in this register can lag behind the value in the internal counter for up to six system plus eight counter clock cycles.

The SWT_CO register can be used during a software self test of the SWT. For example, the SWT can be enabled and not serviced for a fixed period of time less than the time-out value. Then the SWT can be disabled (SWT_CR[WEN] cleared) and the value of the SWT_CO register read to determine if the internal down counter is working properly.

m. See SWT section in device configuration chapter.

40.4.2 Configuration locking

The configuration of the SWT can be locked through use of either a soft lock or a hard lock. In either case, when locked, the SWT_CR, SWT_TO, SWT_WN and SWT_SK registers are read-only.

40.4.2.1 Hard lock

The hard lock is enabled by setting the SWT_CR[HLK] bit, which can only be cleared by a reset.

40.4.2.2 Soft lock

The soft lock is enabled by setting the SWT_CR[SLK] bit and is cleared by writing the unlock sequence to the service register.

40.4.3 Unlock sequence

The unlock sequence is a write of 0xC520 followed by a write of 0xD928 to the SWT_SR[WSC] field. There is no timing requirement between the two writes. The unlock sequence logic ignores service sequence writes and recognizes the 0xC520, 0xD928 sequence regardless of previous writes. The unlock sequence can be written at any time and does not require the SWT_CR[WEN] bit to be set.

40.4.4 Servicing operations

When enabled, the SWT requires periodic execution of a servicing operation which is determined by the SWT_CR[SMD] field. Properly servicing the watchdog loads the internal down counter with the time-out period. The servicing modes are:

- fixed service sequence
- keyed service sequence
- fixed execution address
- incremental execution address

40.4.4.1 Fixed service sequence mode

If the SWT_CR[SMD] field is 00b, the fixed service sequence mode is selected which requires writing 0xA602, 0xB480 to the SWT_SR[WSC] field to service the watchdog. There is no timing requirement between the two writes and the service sequence logic ignores unlock sequence writes.

40.4.4.2 Keyed service sequence mode

If the SWT_CR[SMD] field is 01b, then the keyed service sequence mode is selected which requires writing two pseudorandom keys to the SWT_SR[WSC] field to service the watchdog. The key values are determined by the pseudorandom key generator defined in *Equation 24*. This algorithm will generate a sequence of 2^{16} different key values before repeating. The state of the key generator is held in the SWT_SK register. For example, if SWT_SK[SK] is 0x0100, then the service sequence keys are 0x1103, 0x2136. In this mode, each time a valid key is written to the SWT_SR register, the SWT_SK register is updated. So, after servicing the watchdog by writing 0x1103 and then 0x2136 to the SWT_SR[WSC] field, SWT_SK[SK] is 0x2136 and the next key sequence is 0x3499, 0x7E2C.

Equation 24

$$SK_{n+1} = (17*SK_n+3) \bmod 2^{16}$$

40.4.4.3 Fixed execution address mode

If the SWT_CR[SMD] field is set to 10b, the fixed execution address mode is selected which requires executing code at the address loaded into the designated IAC register to service the watchdog. In this mode, the IAC register can not be updated while the watchdog is enabled.

40.4.4.4 Incremental execution address mode

If the SWT_CR[SMD] field is set to 11b, the incremental execution address mode is selected in which requires executing code at the address loaded into the designated IAC register to service the watchdog. In this mode, the IAC register can be updated while the watchdog is enabled.

40.4.4.5 Window mode

If window mode is enabled (SWT_CR[WND] bit is set), the service sequence must be performed in the last part of the time-out period defined by the window register. The window is open when the down counter is less than the value in the SWT_WN register. Outside of this window, service sequence writes are invalid accesses and generate a bus error or reset depending on the value of the SWT_CR[RIA] bit. For example, if the SWT_TO register is set to 5000 and SWT_WN register is set to ‘1000’ then the service sequence must be performed in the last 20% of the time-out period. There is a short lag in the time it takes for the window to open due to synchronization logic in the watchdog design. This delay could be up to three system plus four counter clock cycles.

40.4.5 Time-out

The SWT_CR[ITR] (“Interrupt Then Reset” bit) controls the action taken when a time-out occurs. If the SWT_CR[ITR] bit is not set, a reset is generated immediately on a time-out. If the SWT_CR[ITR] bit is set, an initial time-out causes the SWT to generate an interrupt and load the down counter with the time-out period. If the service sequence is not written before the second consecutive time-out, the SWT generates a system reset. The interrupt is indicated by the time-out interrupt flag (SWT_IR[TIF]). The interrupt request is cleared by writing a one to the SWT_IR[TIF] bit.

40.4.6 Initialization

All registers should be initialized before setting the SWT_CR[WEN] bit to enable the watchdog. Registers can be initialized in any sequence.

41 Periodic Interrupt Timer (PIT)

41.1 Introduction

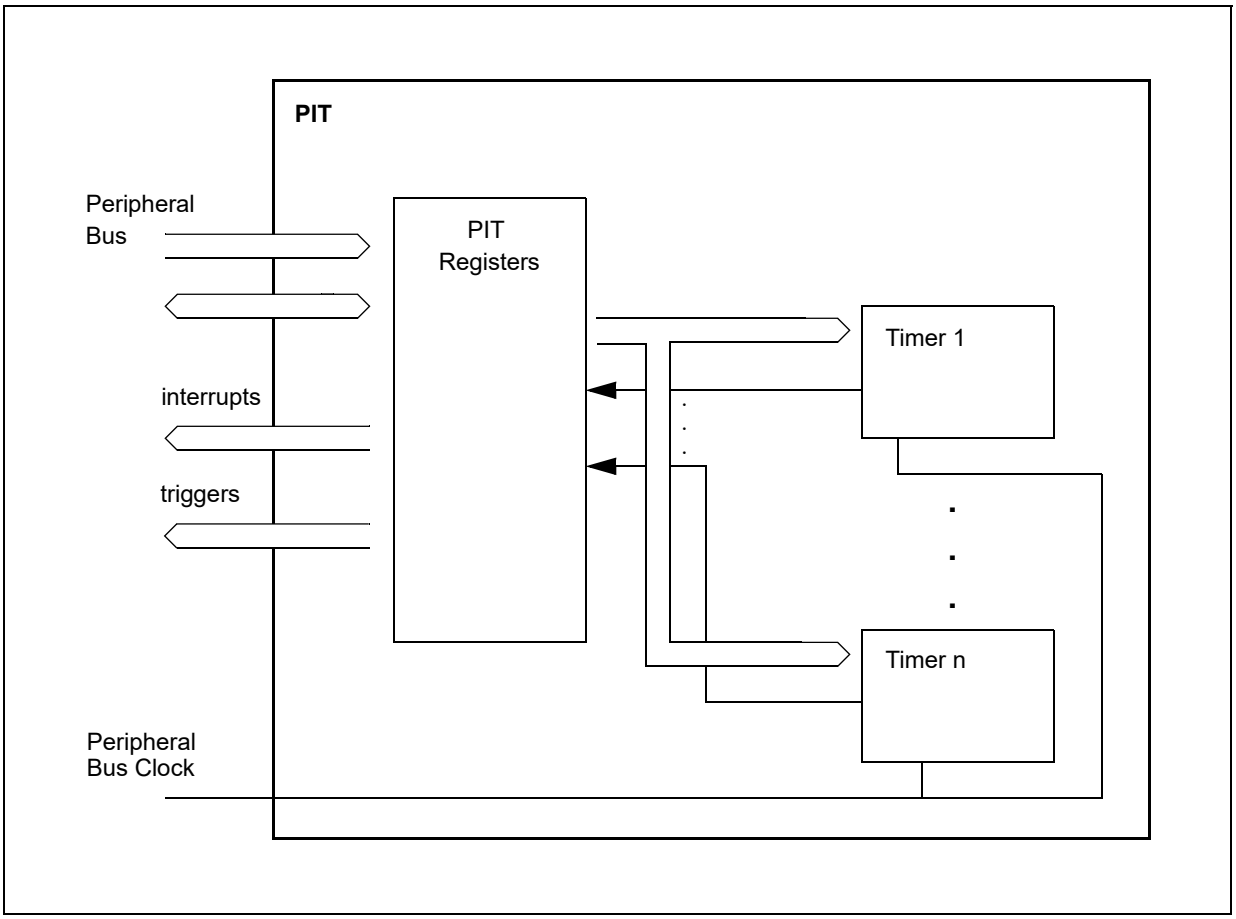
The PIT timer module is an array of timers that can be used to raise interrupts and trigger DMA channels.

Note: For the chip-specific implementation details of this module's instances see the Device Configuration chapter.

41.1.1 Block diagram

Figure 560 shows the PIT block diagram.

Figure 560. Block diagram of the PIT



Note: Refer to the Chip Configuration information for the number of PIT channels used in this MCU.

41.1.2 Features

The main features of this block are:

- Timers can generate DMA trigger pulses
- Timers can generate interrupts
- All interrupts are maskable
- Independent timeout periods for each timer

41.2 Signal description

This module has no external pins.

41.2.1 Memory map/register description

This section provides a detailed description of all registers accessible in this module.

Note: Reserved registers will read as 0, writes will have no effect.

Table 624. PIT memory map

Offset (hex)	Register name	Location
0x0000	PIT Control Register (MCR)	*Section 41.2.1.1*
0x0004–0x00FF	Reserved	
PIT registers per channel (n = 0 to 7)		
0x0100 + *n**0x10	PIT Timer Load Value Register n (LDVAL*n*)	*Section 41.2.1.2*
0x0104 + *n**0x10	PIT Current Timer Value Register n (CVAL*n*)	*Section 41.2.1.3*
0x0108 + *n**0x10	PIT Timer Control Register n (TCTRL*n*)	*Section 41.2.1.4*
0x010C + *n**0x10	PIT Timer Flag Register n (TFLG*n*)	*Section 41.2.1.5*
0x0180–0x3FFF	Reserved	

41.2.1.1 PIT Module Control Register (MCR)

The PIT Module Control Register (MCR) controls whether the timer clocks should be enabled and whether the timers should run in debug mode.

Offset: 0x0000 Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	MDIS	FRZ
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0

Figure 561. PIT Module Control Register (MCR)

Table 625. MCR field descriptions

Field	Description
30 MDIS	Module Disable This is used to disable the module clock. This bit must be enabled before any other setup is done. 0 Clock for PIT timers is enabled. 1 Clock for PIT timers is disabled.
31 FRZ	Freeze Allows the timers to be stopped when the device enters debug mode. 0 Timers continue to run in debug mode. 1 Timers are stopped in debug mode.

41.2.1.2 PIT Timer Load Value Register *n* (LDVAL*n*)

Offset: 0x0100 + n * 0x10 (n = 0 to 7) Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TSV															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 562. PIT Timer Load Value Register *n* (LDVAL*n*)

Table 626. LDVAL*n* field descriptions

Field	Description
16:31 TSV	Timer Start Value Bits These bits set the timer start value. The timer will count down until it reaches 0, then it will generate an interrupt and load this register value again. Writing a new value to this register will not restart the timer, instead the value will be loaded once the timer expires. To abort the current cycle and start a timer period with the new value, the timer must be disabled and enabled again.

41.2.1.3 PIT Current Timer Value Register *n* (CVAL*n*)

Offset: 0x0104 + *n**0x10 (n = 0 to 7) Access: User read-only

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TVL															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 563. PIT Current Timer Value Register *n* (CVAL*n*)

Table 627. CVAL*n* field descriptions

Field	Description
16:31 TVL	Current Timer Value If the timer is enabled, these bits represent the current timer value. If the timer is disabled, do not use this field as its value is unreliable. **Note:** The timer uses a down counter. The timer values are frozen in debug mode if the MCR[FRZ] bit is set.

41.2.1.4 PIT Timer Control Register *n* (TCTRL*n*)

Offset: 0x0108 + *n**0x10 (*n* = 0 to 7) Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	CHN	TIE	TEN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 564. Timer Control Register (TCTRL*n*)

Table 628. TCTRL*n* field descriptions

Field	Description
29 CHN	Chain Mode Bit When activated, timer n-1 needs to expire before timer n can decrement by 1. Timer 0 cannot be chained. 0 Timer is not chained. 1 Timer is chained to previous timer (example, for channel 2 if this bit is set timer2 is chained 1).
30 TIE	Timer Interrupt Enable Bit 0 Interrupt requests from Timer n are disabled. 1 Interrupt will be requested whenever TIF is set. When an interrupt is pending (TIF set), enabling the interrupt will immediately cause an interrupt event. To avoid this, the associated TIF flag must be cleared first.
31 TEN	Timer Enable Bit This bit enables or disables the timer. 0 Timer n is disabled. 1 Timer n is active.

41.2.1.5 PIT Timer Flag Register *n* (TFLG*n*)

Offset: 0x010C + *n**0x10 (*n* = 0 to 7) Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	TIF
W																w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 565. Timer Flag Register *n* (TFLG*n*)

Table 629. TFLG*n* field descriptions

Field	Description
31 TIF	Timer Interrupt Flag TIF is set to ‘1’ at the end of the timer period. This flag can be cleared only by writing it with 1. Writing 0 has no effect. If enabled (TIE = 1), TIF causes an interrupt request. 0 Timeout has not yet occurred. 1 Timeout has occurred.

41.3 Functional description

This section provides the functional description of the module.

41.3.1 General

This section gives detailed information on the internal operation of the module. Each timer can be used to generate trigger pulses as well as to generate interrupts. Each interrupt is available on a separate interrupt line.

41.3.1.1 Timers

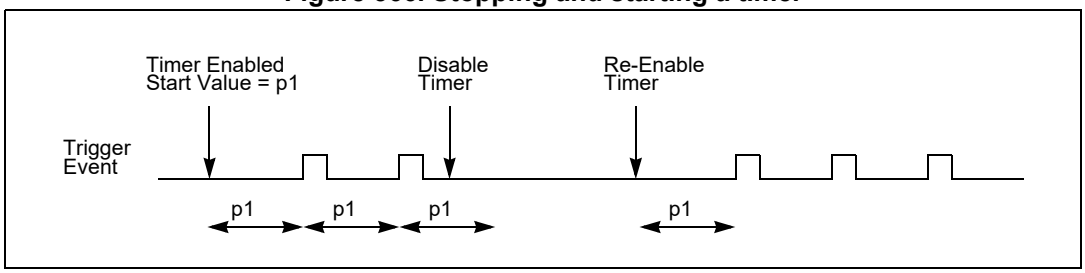
The timers generate triggers at periodic intervals, when enabled. They load their start values, as specified in their LDVAL registers, then count down until they reach 0. Then they load their respective start value again. Each time a timer reaches 0, it will generate a trigger pulse and set the interrupt flag.

All interrupts can be enabled or masked (by setting the TIE bits in the TCTRL registers). A new interrupt can be generated only after the previous one is cleared.

If desired, the current counter value of the timer can be read via the CVAL registers.

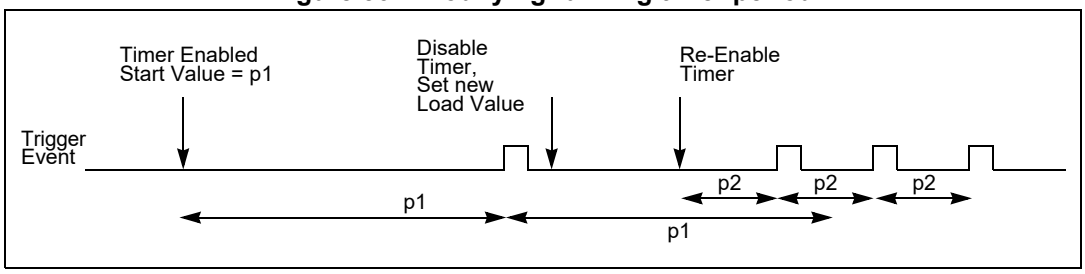
The counter period can be restarted, by first disabling, then enabling the timer with the TEN bit (see the following figure).

Figure 566. Stopping and starting a timer



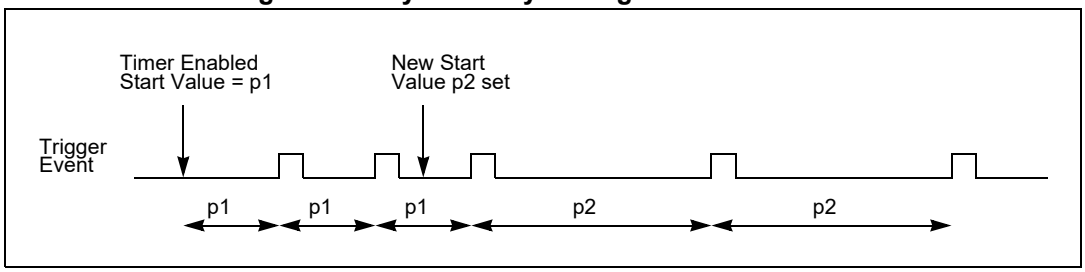
The counter period of a running timer can be modified, by first disabling the timer, setting a new load value and then enabling the timer again (see the following figure).

Figure 567. Modifying running timer period



It is also possible to change the counter period without restarting the timer by writing the LDVAL register with the new load value. This value will then be loaded after the next trigger event (see the following figure).

Figure 568. Dynamically setting a new load value



41.3.1.2 Debug mode

In debug mode, the timers will be frozen based on field MCR[FRZ]. This is intended to aid software development, allowing the developer to halt the processor, investigate the current state of the system (for example, the timer values) and then continue the operation.

41.3.2 Interrupts

All of the timers support interrupt generation. Refer to the MCU specification for related vector addresses and priorities.

Timer interrupts can be enabled by setting the TIE bits. The timer interrupt flags (TIF) are set to '1' when a timeout occurs on the associated timer, and are cleared to '0' by writing a '1' to that TIF bit.

41.3.3 Chained timers

When a timer has chain mode enabled, it will only count after the previous timer has expired. So if timer *n*-1 has counted down to 0, counter *n* will decrement the value by one. This allows to chain some of the timers together to form a longer timer. The first timer (timer 0) cannot be chained to any other timer.

41.4 Initialization and application information

In the example configuration:

- The PIT clock has a frequency of 50 MHz.
- Timer 1 creates an interrupt every 50 µs.
- Timer 3 creates a trigger event every 1 ms.

First the PIT module must be activated by writing a 0 to field MCR[MDIS].

The 50 MHz clock frequency equates to a clock period of 20 ns.

Timer 1 needs to trigger every 50 µs/20 ns = 2500 cycles and timer 3 every 1 ms/20 ns = 50000 cycles.

The value for the LDVAL register trigger is calculated as:

LDVAL trigger = (period / clock period) - 1

This means LDVAL1 should be written with 0x000009C4, and LDVAL3 should be written with 0x0000C350.

The interrupt for Timer 1 is enabled by setting bit TCTRL1[TIE]. The timer is started by writing '1' to bit TCTRL1[TEN].

Timer 3 shall be used only for triggering. Therefore Timer 3 is started by writing a ‘1’ to bit TCTRL3[TEN]; bit TIE stays at 0.

The following example code matches the described setup:

Example 3

```
// turn on PIT
PIT_MCR = 0x00;


// Timer 1
PIT_LDVAL1 = 0x000009C4; // setup timer 1 for 256000 cycles
PIT_TCTRL1 = TIE;        // enable Timer 1 interrupts
PIT_TCTRL1 |= TEN;         // start Timer 1

// Timer 3
PIT_LDVAL3 = 0x0000C350; // setup timer 3for 1500000 cycles
PIT_TCTRL3 |= TEN;         // start Timer 3
```

42 CAN Subsystem

42.1 Introduction

SPC574Sx has two CAN subsystems implemented:

- CAN Subsystem 0
- CAN Subsystem 1

Note: Refer to Section 7.9.1: CAN subsystem configuration in Chapter 7: Device configuration to see the subsystems implementation.

The Controller Area Network (CAN) subsystem consists of the modular CAN (M_CAN) modules along with an integrated intelligent CAN RAM controller. The CAN RAM controller consists of additional logic for arbitration between the requests for the RAM access by the various CANs and CPU, ECC encoder/decoder for the Message RAM data and active transmit message buffer protection from CPU write access. The subsystem follows the little endian format.

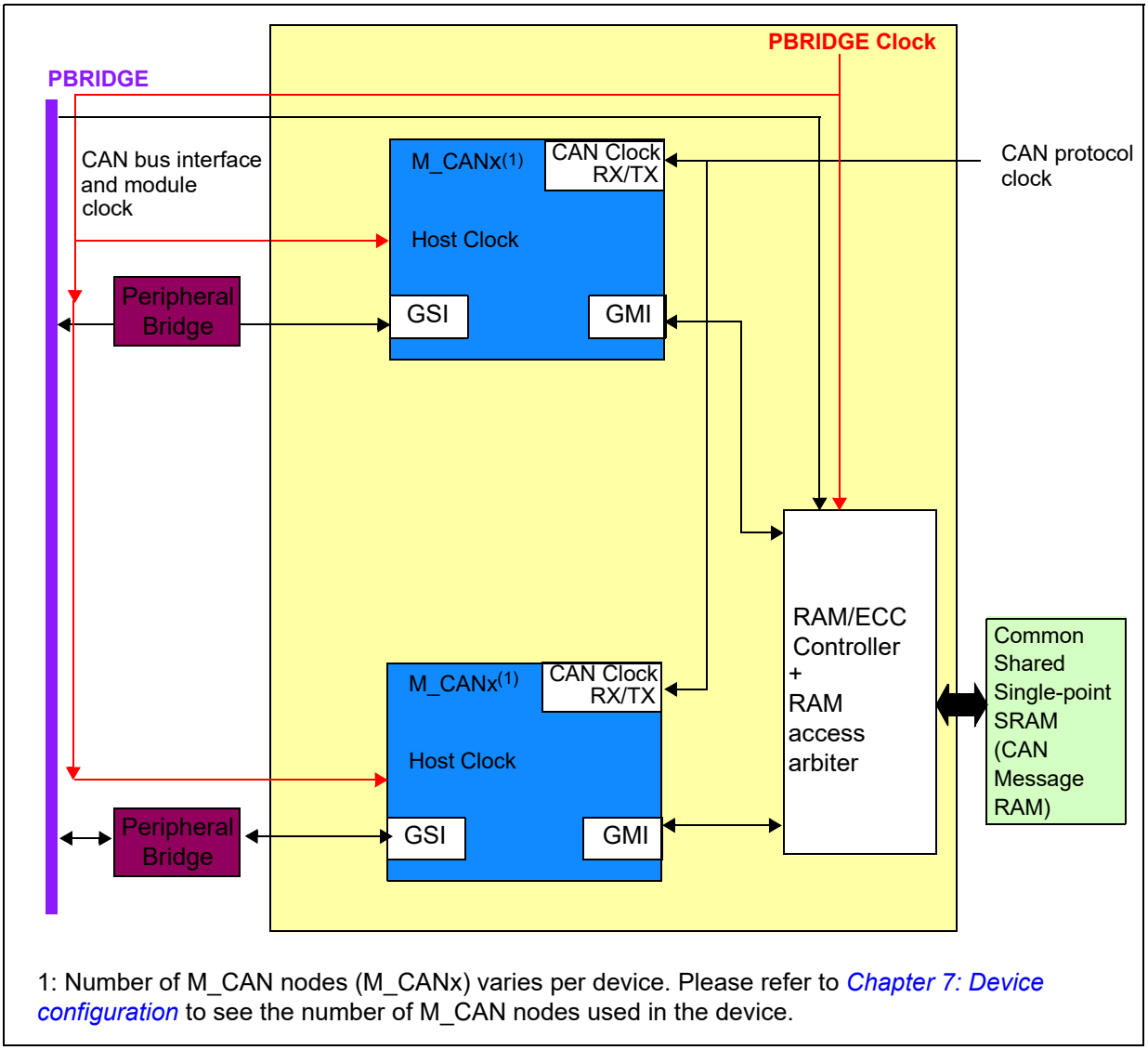
Note: Refer to Section 7.9.1: CAN subsystem configuration in Chapter 7: Device configuration to see the number of CAN nodes used in the device.

Table 630. IP user manual versions

IP Name	Version
M_CAN IP	Revision 3.2.1

The general CAN subsystem implementation is shown in *Section 42.2*. It interfaces with the Host processor bus using the peripheral bus. The RAM is not implemented inside the subsystem.

Figure 569. CAN subsystem generic block diagram



1: Number of M_CAN nodes (M_CANx) varies per device. Please refer to *Chapter 7: Device configuration* to see the number of M_CAN nodes used in the device.

42.2 Features

The CAN subsystem consists of the following major blocks:

- Modular CAN cores: The registers of the CAN module can be accessed using the Generic Slave Interface (GSI)
- CAN-RAM arbiter
- SRAM interface and memory organization
- ECC Controller

42.3 Modular CAN (M_CAN) cores

M_CAN functionality conforms to CAN specification V2.0B active for each M_CAN node.The M_CAN performs communication according to the CAN protocol specification 2.0 part A,B and to CAN FD 1.0.Flexible assignment of Message Objects to nodes.
The bit rate can be programmed to values up to 1 Mbit/s for standard CAN frames. High bit rates are possible in CAN FD mode. Additional transceiver hardware is required for connection to the physical layer.

Note: Implemented CAN nodes are fully compliant to ISO CAN FD specification ISO 11898-1: 2015.

The message storage is intended to be a single-ported Message RAM outside of the module. It is connected to the M_CAN via the Generic Master Interface. Depending on the chosen device, multiple M_CAN controllers can share the same Message RAM.

All functions concerning the handling of messages are implemented by the Rx Handler and the Tx Handler. The Rx Handler manages message acceptance filtering, the transfer of received messages from the CAN core to the Message RAM as well as providing receive message status information. The Tx Handler is responsible for the transfer of transmit messages from the Message RAM to the CAN core as well as providing transmit status information.

Acceptance filtering is implemented by a combination of up to 128 filter elements where each one can be configured as a range, as a bit mask, or as a dedicated ID filter.

The M_CAN’s clock domain concept allows the separation between the high precision CAN clock and the Host clock, which may be generated by an FMPLL.

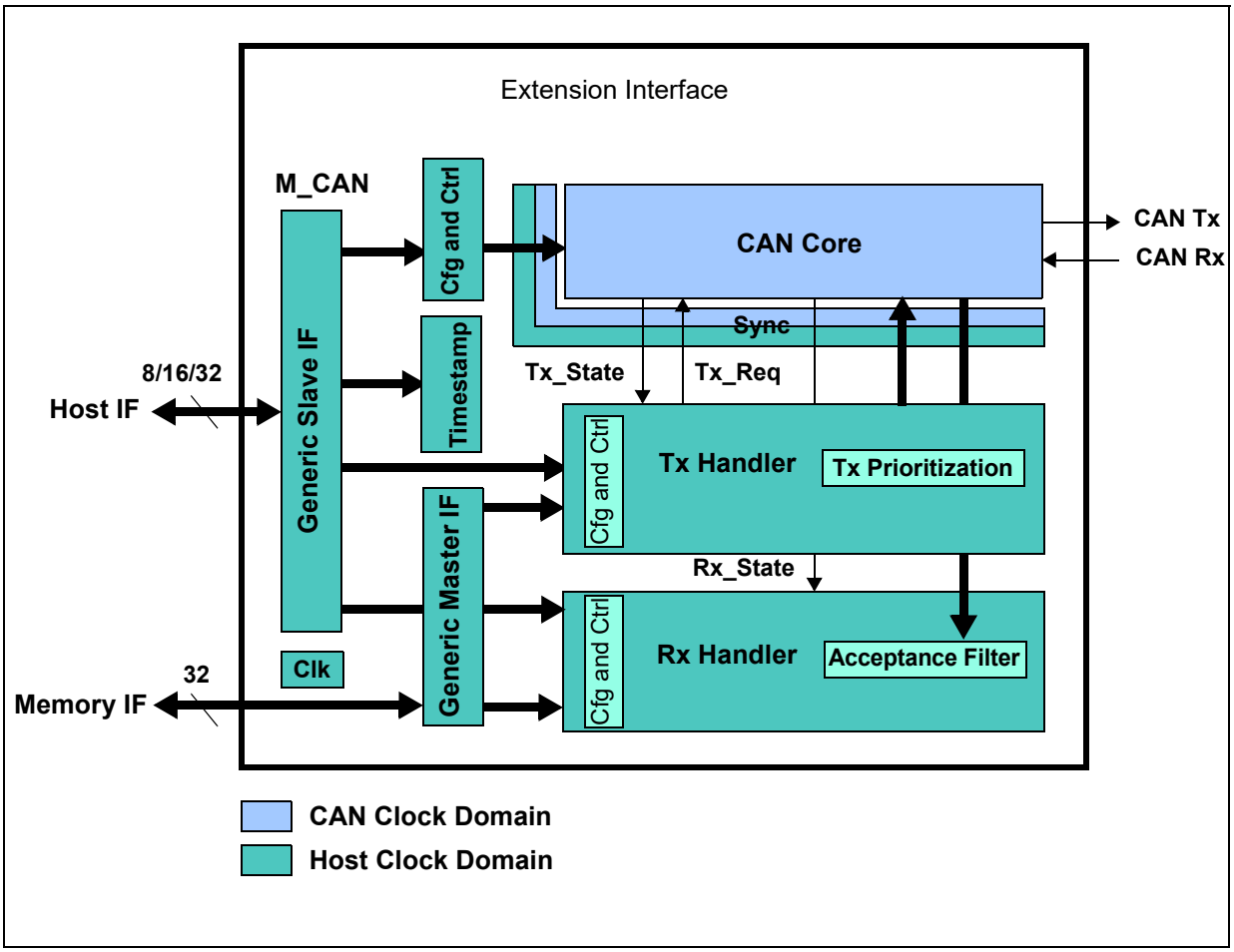
42.3.1 Features

The following are the features of Modular CAN cores.

- Conforms with CAN protocol version 2.0 part A, B and ISO 11898-1: 2015
- CAN Flexible data-rate (ISO CAN FD) protocol with 64 data bytes on M_CAN is supported
- Bit rates up to 1 Mbit/s in standard CAN mode
- Bit rates up to 8 Mbit/s in ISO CAN FD mode
- CAN error logging
- AUTOSAR optimized
- SAE J1939 optimized
- Improved acceptance filtering
- Two configurable Receive FIFOs
- Separate signaling on reception of High Priority Messages
- Up to 64 dedicated Receive Buffers
- Up to 32 dedicated Transmit Buffers
- Configurable Transmit FIFO
- Configurable Transmit Queue
- Configurable Transmit Event FIFO
- Direct Message RAM access for Host CPU
- Multiple M_CANs share the same Message RAM
- Programmable loop-back test mode
- Maskable module interrupts
- 8-/16-/32-bits Generic Slave Interface for connection customer-specific Host CPUs
- Two clock domains (CAN clock and Host clock)
- Power-down support
- Debug over M_CAN
- Clock calibration over M_CAN is supported

42.3.2 Block diagram

Figure 570. M_CAN core block diagram



- CAN core: CAN Protocol Controller and Rx/Tx Shift Register. Handles all ISO 11898-1 protocol functions. Supports 11-bit and 29-bit identifiers.
- Sync: Synchronizes signals from the Host clock domain to the CAN clock domain and vice versa.
- Clk: Synchronizes reset signal to the Host clock domain and to the CAN clock domain.
- Cfg and Ctrl: CAN core related configuration and control bits.
- Interrupt and Timestamp: Interrupt control and 16-bit CAN bit time counter for receive and transmit timestamp generation.
- Tx Handler: Controls the message transfer from the external Message RAM to the CAN core. A maximum of 32 Tx Buffers can be configured for transmission. Tx buffers can be used as dedicated Tx Buffers, as Tx FIFO, part of a Tx Queue, or as a combination of them. A Tx Event FIFO stores Tx timestamps together with the corresponding Message ID. Transmit cancellation is also supported.
- Rx Handler: Controls the transfer of received messages from the CAN core to the external Message RAM. The Rx Handler supports two Receive FIFOs, each of configurable size, and up to 64 dedicated Rx Buffers for storage of all messages that have passed acceptance filtering. A dedicated Rx Buffer, in contrast to a Receive FIFO, is used to store only messages with a specific identifier. An Rx timestamp is stored

together with each message. Up to 128 filters can be defined for 11-bit IDs and up to 64 filters for 29-bit IDs.

- Generic Slave Interface: Connects the M_CAN to a customer specific Host CPU. The Generic Slave Interface is capable to connect to an 8-/16-/32-bit bus to support a wide range of interconnection structures.
- Generic Master Interface: Connects the M_CAN access to an external 32-bit Message RAM. The maximum Message RAM size is 16 KB × 32 bits.
- Extension Interface: All flags from the Interrupt Register IR as well as selected internal status and control signals are routed to this interface. The interface is intended for connection of the M_CAN to a module-external interrupt unit or to other module-external components. The connection of these signals is optional.

42.3.3 Dual clock sources

To improve the EMC behavior, a spread spectrum clock can be used for the Host clock domain. Due to the high precision clocking requirements of the CAN core, a separate clock without any modulation has to be provided as CAN clock.

Within the M_CAN module there is a synchronization mechanism implemented to ensure save data transfer between the two clock domains.

Note: In order to achieve a stable function of the M_CAN, the Host clock must always be faster than or equal to the CAN clock. Also the modulation depth of the spread spectrum clock has to be regarded.

Caution: Stop this module when the device is in STOP mode. If CAN message reception is mandatory, check that the PLL is unlocked before transmitting any message. Use the external oscillator (XOSC) instead of the internal RC oscillator (IRCOSC) during the STOP mode.

42.3.4 Dual interrupt lines

The module provides two interrupt lines. Interrupts can be routed either to EINT0 or to EINT1. By default all interrupts are routed to interrupt line EINT0. By programming EINT0 and EINT1 bits of the Interrupt Line Enable (ILE) register, the interrupt lines can be enabled or disabled separately.

42.3.5 Memory map and register descriptions

42.3.5.1 Hardware reset description

After hardware reset, the registers of the M_CAN hold the reset values listed in *Table 631*. Additionally the *Bus_Off* state is reset and the M_CAN Tx output is set to *recessive* (HIGH). The value 0x0001 (bit 0 [INIT] of the CCCR register = ‘1’) in the CC Control Register enables software initialization. The M_CAN does not influence the CAN bus until the CPU resets bit 0 [INIT] of the CCCR register = ‘0’.

42.3.5.2 Register map

The M_CAN module allocates an address space of 256 bytes. All registers are organized as 32-bit registers. The M_CAN is accessible by the Host CPU via the Generic Slave Interface using a data width of 8-bit (byte access), 16-bit (half-word access), or 32-bit (word access).

Note: Write access by the Host CPU to registers/bits marked with "P=Protected Write" is possible only when the bit 1 [CCE] and bit 0 [INIT] of the CCCR register is set to '1'.

There is a delay from writing to a command register until the update of the related status register bits due to clock domain crossing.

Caution: Any write access to reserved or not implemented registers in the 16 KB slot assigned by the peripherals bridge to the M_CAN IP will not generate any bus access error.

Table 631. M_CAN memory map

Address Offset	Register Name	Access(1)	Reset value	Location
0x0000	Core Release Register (CREL)	R	0x3215_0320	*Section 42.3.5.2.1*
0x0004	Endian Register (ENDN)	R	0x8765_4321	*Section 42.3.5.2.2*
0x0008–0x000B	Reserved			
0x000C	Data Bit Timing and Prescaler Register (DBTP)	RP	0x0000_0A33	*Section 42.3.5.2.3*
0x0010	Test Register (TEST)	RP	0x0000_0000	*Section 42.3.5.2.4*
0x0014	RAM Watchdog Register (RWD)	RP	0x0000_0000	*Section 42.3.5.2.5*
0x0018	CC Control Register (CCCR)	RWPp	0x0000_0001	*Section 42.3.5.2.6*
0x001C	Nominal Bit Timing and Prescaler Register (NBTP)	RP	0x0600_0A03	*Section 42.3.5.2.7*
0x0020	Timestamp Counter Configuration Register (TSCC)	RP	0x0000_0000	*Section 42.3.5.2.8*
0x0024	Timestamp Counter Value Register (TSCV)	RC	0x0000_0000	*Section 42.3.5.2.9*
0x0028	Timeout Counter Configuration Register (TOCC)	RP	0xFFFF_0000	*Section 42.3.5.2.10*
0x002C	Timeout Counter Value Register (TOCV)	RC	0x0000_FFFF	*Section 42.3.5.2.11*
0x0030–0x003F	Reserved			
0x0040	Error Counter Register (ECR)	RX	0x0000_0000	*Section 42.3.5.2.12*
0x0044	Protocol Status Register (PSR)	RXS	0x0000_0707	*Section 42.3.5.2.13*
0x0048	Transmitter Delay Compensation Register (TDCR)	RP	0x0000_0000	*Section 42.3.5.2.14*
0x004C–0x004F	Reserved			
0x0050	Interrupt Register (IR)	R/W	0x0000_0000	*Section 42.3.5.2.15*
0x0054	Interrupt Enable Register (IE)	R/W	0x0000_0000	*Section 42.3.5.2.16*
0x0058	Interrupt Line Select Register (ILS)	R/W	0x0000_0000	*Section 42.3.5.2.17*
0x005C	Interrupt Line Enable Register (ILE)	R/W	0x0000_0000	*Section 42.3.5.2.18*
0x0060–0x007F	Reserved			
0x0080	Global Filter Configuration Register (GFC)	RP	0x0000_0000	*Section 42.3.5.2.19*
0x0084	Standard ID Filter Configuration Register (SIDFC)	RP	0x0000_0000	*Section 42.3.5.2.20*

Table 631. M_CAN memory map (continued)

Address Offset	Register Name	Access(1)	Reset value	Location
0x0088	Extended ID Filter Configuration Register (XIDFC)	RP	0x0000_0000	Section 42.3.5.2.21
0x008C–0x008F	Reserved			
0x0090	Extended ID AND Mask Register (XIDAM)	RP	0x1FFF_FFFF	Section 42.3.5.2.22
0x0094	High Priority Message Status Register (HPMS)	R	0x0000_0000	Section 42.3.5.2.23
0x0098	New Data 1 Register (NDAT1)	R/W	0x0000_0000	Section 42.3.5.2.24
0x009C	New Data 2 Register (NDAT2)	R/W	0x0000_0000	Section 42.3.5.2.25
0x00A0	Rx FIFO 0 Configuration Register (RXF0C)	RP	0x0000_0000	Section 42.3.5.2.26
0x00A4	Rx FIFO 0 Status Register (RXF0S)	R	0x0000_0000	Section 42.3.5.2.27
0x00A8	Rx FIFO 0 Acknowledge Register (RXF0A)	R/W	0x0000_0000	Section 42.3.5.2.28
0x00AC	Rx Buffer Configuration Register (RXBC)	RP	0x0000_0000	Section 42.3.5.2.29
0x00B0	Rx FIFO 1 Configuration Register (RXF1C)	RP	0x0000_0000	Section 42.3.5.2.30
0x00B4	Rx FIFO 1 Status Register (RXF1S)	R	0x0000_0000	Section 42.3.5.2.31
0x00B8	Rx FIFO 1 Acknowledge Register (RXF1A)	R/W	0x0000_0000	Section 42.3.5.2.32
0x00BC	Rx Buffer / FIFO Element Size Configuration Register (RXESC)	RP	0x0000_0000	Section 42.3.5.2.33
0x00C0	Tx Buffer Configuration Register (TXBC)	RP	0x0000_0000	Section 42.3.5.2.34
0x00C4	Tx FIFO/Queue Status Register (TXFQS)	R	0x0000_0000	Section 42.3.5.2.35
0x00C8	Tx Buffer Element Size Configuration Register (TXESC)	RP	0x0000_0000	Section 42.3.5.2.36
0x00CC	Tx Buffer Request Pending Register (TXBRP)	R	0x0000_0000	Section 42.3.5.2.37
0x00D0	Tx Buffer Add Request Register (TXBAR)	R/W	0x0000_0000	Section 42.3.5.2.38
0x00D4	Tx Buffer Cancellation Register (TXBCR)	R/W	0x0000_0000	Section 42.3.5.2.39
0x00D8	Tx Buffer Transmission Occurred Register (TXBTO)	R	0x0000_0000	Section 42.3.5.2.40
0x00DC	Tx Buffer Cancellation Finished Register (TXBCF)	R	0x0000_0000	Section 42.3.5.2.41
0x00E0	Tx Buffer Transmission Interrupt Enable (TXBTIE)	R/W	0x0000_0000	Section 42.3.5.2.42
0x00E4	Tx Buffer Cancellation Finished Interrupt Enable Register (TXBCIE)	R/W	0x0000_0000	Section 42.3.5.2.43
0x00E8–0x00EF	Reserved			
0x00F0	Tx Event FIFO Configuration Register (TXEFC)	RP	0x0000_0000	Section 42.3.5.2.44
0x00F4	Tx Event FIFO Status Register (TXEFS)	R	0x0000_0000	Section 42.3.5.2.45

Table 631. M_CAN memory map (continued)

Address Offset	Register Name	Access(1)	Reset value	Location
0x00F8	Tx Event FIFO Acknowledge Register (TXEFA)	R/W	0x0000_0000	*Section 42.3.5.2.46*
0x00FC–0x00FF	Reserved			

1. R = Read, S = Set on read, W = Write, P = Protected Write, C = Clear/preset on write, p = Protected set, X= Reset on read

42.3.5.2.1 Core Release Register (CREL)

Address: 0x0000 Access: Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	REL				STEP				SUBSTEP				YEAR			
W																
Reset(1)	-r	-r	-r	-r	-r	-r	-r	-r	-r	-r	-r	-r	-d	-d	-d	-d

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	MON								DAY							
W																
Reset(1)	-d	d-	-d	-d	-d	-d	-d	-d	-d	-d	-d	-d	-d	-d	-d	-d

1. -r = release, -d = time stamp. The coding of revisions depends on the module version used in the device. For the device reset value refer to *Table 631: M_CAN memory map*.

Figure 571. Core Release Register (CREL)

Table 632. CREL field descriptions

Field	Description
0:3 REL	Core Release One digit, BCD.
4:7 STEP	Step of Core Release One digit, BCD.
8:11 SUBSTEP	Substep of Core Release One digit, BCD.
12:15 YEAR	Time Stamp Year One digit, BCD.
16:23 MON	Time Stamp Month Two digits, BCD.
24:31 DAY	Time Stamp Day Two digits, BCD.

42.3.5.2.2 Endian Register (ENDN)

Address: 0x0004 Access: Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	ETV															
W																
Reset	1	0	0	0	0	1	1	1	0	1	1	0	0	1	0	1

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	ETV															
W																
Reset	0	1	0	0	0	0	1	1	0	0	1	0	0	0	0	1

Figure 572. Endian Register (ENDN)

Table 633. ENDN field descriptions

Field	Description
0:31 ETV	Endianness Test Value The endianness test value is 0x87654321.

42.3.5.2.3 Data Bit Timing and Prescaler Register (DBTP)

This register is only writable if bits **CCCR.CCE** and **CCCR.INIT** are set. The CAN bit time may be programed in the range of 4 to 25 time quanta. The CAN time quantum may be programmed in the range of 1 to 32 M_CAN clock periods. tq = (FBRP + 1) M_CAN clock period.

FTSEG1 is the sum of Prop_Seg and Phase_Seg1. **FTSEG2** is Phase_Seg2.

Therefore the length of the bit time is (programmed values) **[FTSEG1 + FTSEG2 + 3]** tq or (functional values) [Sync_Seg + Prop_Seg + Phase_Seg1 + Phase_Seg2] tq.

The Information Processing Time (IPT) is zero, meaning that the data for the next bit is available at the first clock edge after the sample point.

Address: 0x000C Access: Read Protected

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	TDC(1)	0	0	DBRP(1)				
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	DTSEG1(1)					DTSEG2(1)				DSJW(1)			
W																
Reset	0	0	0	0	1	0	1	0	0	0	1	1	0	0	1	1

1. These are protected write (P) bits which means that write access by the bits is possible only when the bit 1 [CCE] and bit 0 [INIT] of CCCR register are set to ‘1’.

Figure 573. Data Bit Timing and Prescaler Register (DBTP)

Table 634. DBTP field descriptions

Field	Description
8 TDC	Transceiver Delay Compensation 0 Transceiver Delay Compensation disabled 1 Transceiver Delay Compensation enabled
11:15 DBRP	Data Baud Rate Prescaler (0x00–0x1F) The value by which the oscillator frequency is divided for generating the bit time quanta. The bit time is built up from a multiple of this quanta. Valid values for the Baud Rate Prescaler are 0 to 31. The actual interpretation by the hardware of this value is such that one more than the value programmed here is used.
19:23 DTSEG1	Data time segment before sample point (0x0–0x1F) Valid values are 0 to 31. The actual interpretation by the hardware of this value is such that one more than the programmed value is used.
24:27 DTSEG2	Data time segment after sample point (0x0–0xF) Valid values are 0 to 15. The actual interpretation by the hardware of this value is such that one more than the programmed value is used.
28:31 DSJW	Data (Re) Synchronization Jump Width (0x0–0xF) Valid values are 0 to 15. The actual interpretation by the hardware of this value is such that one more than the value programmed here is used.

Note: With a CAN clock of 8 MHz, the reset value of 0x00000A33 configures the M_CAN for a fast bit rate of 500 kbit/s.

The bit rate configured for the CAN FD data phase via DBTP must be higher or equal to the bit rate configured for the arbitration phase via NBTP.

To achive high bit rates in CAN FD mode for the data phase up to 8 Mbit/s the number of time quanta bits per CAN bit can be chosen down to 4 time quanta per bit time.

42.3.5.2.4 Test Register (TEST)

Write access to the Test Register has to be enabled by setting CCCR[TEST] to ‘1’. All Test Register functions are set to their reset values when CCCR[TEST] is reset. Loop Back mode and software control of M_CAN Tx pin are hardware test modes. Programming of **TX** ≠ ‘00’ may disturb the message transfer on the CAN bus.

Address: 0x0010 Access: Read-Write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	RX	TX(1)		LBCK(1)	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	—	0	0	0	0	0	0	0

1. These are protected write (P) bits which means that write access by the bits is possible only when the bit 1 [CCE] and bit 0 [INIT] of CCCR register are set to ‘1’.

Figure 574. Test Register (TEST)

Table 635. TEST field descriptions

Field	Description
24 RX	Receive Pin Monitors the actual value of CAN Rx pin. 0 The CAN bus is dominant (CAN Rx input = ‘0’) 1 The CAN bus is recessive (CAN Rx input = ‘1’)
25:26 TX	Control of Transmit Pin 00 Reset value, CAN Tx output is controlled by the CAN core, updated at the end of the CAN bit time. 01 Sample Point can be monitored at pin CAN Tx output 10 Dominant (‘0’) level at pin CAN Tx output 11 Recessive (‘1’) at pin CAN Tx output
27 LBCK	Loop Back mode 0 Reset value, Loop Back mode is disabled 1 Loop Back mode is enabled (see *Section 42.3.7.9: Test modes*)

42.3.5.2.5 RAM Watchdog Register (RWD)

The RAM Watchdog monitors the READY output of the Message RAM. A Message RAM access via the M_CAN’s Generic Master Interface starts the Message RAM Watchdog Counter with the value configured by the WDC bits. The counter is reloaded with WDC bits when the Message RAM signals successful completion by activating its READY output. In case there is no response from the Message RAM until the counter has counted down to zero, the counter stops and interrupt flag WDI bit of the IR is set. The RAM Watchdog Counter is clocked by the Host clock .

Address: 0x0014 Access: Read Protected

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	WDV								WDC(1)							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. These are protected write (P) bits which means that write access by the bits is possible only when the bit 1 [CCE] and bit 0 [INIT] of CCCR register are set to ‘1’.

Figure 575. RAM Watchdog Register (RWD)

Table 636. RWD field descriptions

Field	Description
16:23 WDV	Watchdog Value Actual Message RAM Watchdog Counter Value.
24:31 WDC	Watchdog Configuration Start value of the Message RAM Watchdog Counter. With the reset value of ‘00’ the counter is disabled.

42.3.5.2.6 CC Control Register (CCCR)

The following section shows the CAN Control register.For details about setting and resetting of single bits see *Section 42.3.7.1.1: Software initialization*.

Address: 0x0018 Access: Read Protected

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	NISO(1)	TXP(1)	EFBI(1)	PXHD(1)	0	0	BRSE(1)	FDOE(1)	TEST(2)	DAR(1)	MON(2)	CSR	CSA	ASM(2)	CCE(1)	INIT
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

1. These are protected write (P) bits which means that write access by the bits is possible only when the bit 1 [CCE] and bit 0 [INIT] of CCCR register are set to ‘1’.
2. These are protected set (p) bits.

Figure 576. CC Control register (CCCR)

Table 637. CCCR field descriptions

Field	Description
16 NISO	Non ISO Operation If this bit is set, the M_CAN uses the CAN FD frame format as specified by the Bosch CAN FD Specification V1.0. 0 CAN FD frame format according to ISO11898-1 1 CAN FD frame format according to Bosch CAN FD Specification V1.0
17 TXP	Transmit Pause If this bit is set, the M_CAN pauses for two CAN bit times before starting the next transmission after itself has successfully transmitted a frame. 0 Transmit pause disabled 1 Transmit pause enabled
18 EFBI	Edge Filtering during Bus Integration 0 Edge filtering disabled 1 Two consecutive dominant tq required to detect an edge for hard synchronization
19 PXHD	Protocol Exception Handling Disable 0 Protocol exception handling enabled 1 Protocol exception handling disabled **Note:** When protocol exception handling is disabled, the M_CAN will transmit an error frame when it detects a protocol exception condition.
22 BRSE	Bit Rate Switch Enable 0 Bit rate switching for transmissions disabled 1 Bit rate switching for transmissions enabled **Note:** When CAN FD operation is disabled FDOE = ‘0’, BRSE is not evaluated.
23 FDOE	FD Operation Enable 0 FD operation disabled 1 FD operation enabled
24 TEST	Test Mode Enable 0 Normal operation, register TEST holds reset values 1 Test Mode, write access to register TEST enabled
25 DAR	Disable Automatic Retransmission 0 Automatic retransmission of messages not transmitted successfully enabled 1 Automatic retransmission disabled
26 MON	Bus Monitoring Mode Bit MON can only be set by the Host when both CCE and INIT are set to ‘1’. The bit can be reset by the Host at any time. 0 Bus Monitoring Mode is disabled 1 Bus Monitoring Mode is enabled
27 CSR	Clock Stop Request 0 No clock stop is requested 1 Clock stop requested. When clock stop is requested, first INIT and then CSA will be set after all pending transfer requests have been completed and the CAN b us reached idle.
28 CSA	Clock Stop Acknowledge 0 No clock stop acknowledged 1 M_CAN may be set in power down by stopping Host clock and core clock

Table 637. CCCR field descriptions (continued)

Field	Description
29 ASM	Restricted Operation Mode Bit ASM is only set by the Host when both CCE and INIT are set to '1'. The bit is reset by the Host at any time. 0 Normal CAN operation 1 Restricted Operation Mode active
30 CCE	Configuration Change Enable 0 The CPU has no write access to the protected configuration registers 1 The CPU has write access to the protected configuration registers (while CCCR.INIT = '1')
31 INIT	Initialization 0 Normal Operation 1 Initialization is started **Note:** Due to the synchronization mechanism between the two clock domains, there may be a delay until the value written to INIT can be read back. Therefore the programmer has to ensure that the previous value written to INIT has been accepted by reading INIT before setting INIT to a new value.

42.3.5.2.7 Nominal Bit Timing and Prescaler Register (NBTP)

This register is only writable if bits CCCR[CCE] and CCCR[INIT] are set. The CAN bit time may be programed in the range of [4…385] time quanta. The CAN time quantum may be programmed in the range of [1…512] M_CAN clock periods.

Equation 25 tq = (NBRP + 1) M_CAN clock period

NTSEG1 is the sum of Prop_Seg and Phase_Seg1. NTSEG2 is Phase_Seg2.

Therefore the length of the bit time is (prog rammed values) [NTSEG1 + NTSEG2 + 3] tq or (functional values) [Sync_Seg + Prop_Seg + Phase_Seg1 + Phase_Seg2] tq.

The Information Processing Time (IPT) is zero, meaning the data for the next bit is available at the first clock edge after the sample point.

Address: 0x001C Access: Read Protected

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	NSJW(1)								NBRP(1)							
Reset	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	NTSEG1(1)								0	NTSEG2(1)						
Reset	0	0	0	0	1	0	1	0	0	0	0	0	0	0	1	1

1. These are protected write (P) bits which means that write access by the bits is possible only when the bit 1 [CCE] and bit 0 [INIT] of CCCR register are set to '1'.

Figure 577. Nominal Bit Timing and Prescaler Register (NBTP)

Table 638. NBTP field descriptions

Field	Description
0:6 NSJW	(Re) Synchronization Jump Width (0x00–0x7F) Valid values are 0 to 127. The actual interpretation by the hardware of this value is such that one more than the value prog rammed here is used.
7:15 NBRP	Baud Rate Prescaler (0x000–0x1FF) The value by which the oscillator frequency is divided for generating the bit time quanta. The bit time is built up from a multiple of this quanta. Valid values for the Baud Rate Prescaler are 0 to 511. The actual interpretation by the hardware of this value is such that one more than the value prog rammed here is used.
16:23 NTSEG1	Time segment before sample point (0x01–0xFF) Valid values are 1 to 255. The actual interpretation by the hardware of this value is such that one more than the prog rammed value is used.
25:31 NTSEG2	The time segment after the sample point (0x00–0x7F) Valid values are 0 to 127. The actual interpretation by the hardware of this value is such that one more than the prog rammed value is used.

Note: With a CAN clock of 8 MHz, the reset value of 0x06000A03 configures the M_CAN for a bit rate of 500 kbit/s.

42.3.5.2.8 Timestamp Counter Configuration Register (TSCC)

Address: 0x0020 Access: RP

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	TCP(1)			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	TSS(1)	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. These are protected write (P) bits which means that write access by the bits is possible only when the bit 1 [CCE] and bit 0 [INIT] of CCCR register are set to ‘1’.

Figure 578. Timestamp Counter Configuration Register (TSCC)

Table 639. TSCC field descriptions

Field	Description
12:15 TCP	Timestamp Counter Prescaler (0x0–0xF) Configures the timestamp and timeout counters time unit in multiples of CAN bit times [1 to16]. The actual interpretation by the hardware of this value is such that one more than the value programmed here is used. **Note:** With CAN FD an external counter is required for timestamp generation (TSS = '10').
30:31 TSS	Timestamp Select 00 Timestamp counter value always 0x0000 01 Timestamp counter value incremented according to Timestamp Counter Prescaler (TCP) 10 External timestamp counter value used 11 Same as '00'

42.3.5.2.9 Timestamp Counter Value Register (TSCV)

Address: 0x0024 Access: R/W

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	TSC															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 579. Timestamp Counter Value Register (TSCV)

Table 640. TSCV field descriptions

Field	Description
16:31 TSC	Timestamp Counter The internal/external Timestamp Counter value is captured on start of frame (both Rx and Tx). When TSCC.TSS = '01', the Timestamp Counter is incremented in multiples of CAN bit times [1…16] depending on the configuration of TSCC.TCP. A wrap around sets interrupt flag IR.TSW. Write access resets the counter to zero. When TSCC.TSS = '10', TSC reflects the external Timestamp Counter value. A write access has no impact.

Note: A "wrap around" is a change of the Timestamp Counter value from non-zero to zero not caused by write access to TSCV.

42.3.5.2.10 Timeout Counter Configuration Register (TOCC)

For a description of the Timeout Counter see *Section 42.3.9*.

Address: 0x0028 Access: RP

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	TOP(1)															
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	TOS(1)		ETOC(1)
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. These are protected write (P) bits which means that write access by the bits is possible only when the bit 1 [CCE] and bit 0 [INIT] of CCCR register are set to '1'.

Figure 580. Timeout Counter Configuration Register (TOCC)

Table 641. TOCC field descriptions

Field	Description
0:15 TOP	Timeout Period Start value of the Timeout Counter (down-counter). Configures the Timeout Period.
29:30 TOS	Timeout Select When operating in Continuous mode, a write to **TOCV** presets the counter to the value configured by **TOCC[TOP]** and continues down-counting. When the Timeout Counter is controlled by one of the FIFOs, an empty FIFO presets the counter to the value configured by **TOCC[TOP]**. Down-counting s started when the first FIFO element is stored. 00 Continuous operation 01 Timeout controlled by Tx Event FIFO 10 Timeout controlled by Rx FIFO 0 11 Timeout controlled by Rx FIFO 1
31 ETOC	Enable Timeout Counter 0 Timeout Counter disabled 1 Timeout Counter enabled **Note:** For use of timeout function with CAN FD see *Section 42.3.9*.

42.3.5.2.11 Timeout Counter Value Register (TOCV)

Address: 0x002C Access: R/W

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TOC															
W																
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Figure 581. Timeout Counter Value Register (TOCV)

Table 642. TOCV field descriptions

Field	Description
16:31 TOC	The Timeout Counter is decremented in multiples of CAN bit times (1 to 16) depending on the configuration of TSCC.TCP. When decremented to zero, interrupt flag IR.TOO is set and the Timeout Counter is stopped. Start and reset/restart conditions are configured via TOCC.TOS.

42.3.5.2.12 Error Counter Register (ECR)

Address: 0x0040 Access: Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	CEL(1)							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	RP	REC							TEC							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. Access type is RX: reset on read.

Figure 582. Error Counter Register (ECR)

Table 643. ECR field descriptions

Field	Description
8:15 CEL	CAN Error Logging The counter is incremented each time when a CAN protocol error causes the Transmit Error Counter or the Receive Error Counter to be incremented. It is reset by read access to CEL. The counter stops at 0xFF; the next increment of TEC or REC sets interrupt flag ELO bit of the IR register.
16 RP	Receive Error Passive 0 The Receive Error Counter is below the error passive level of 128 1 The Receive Error Counter has reached the error passive level of 128
17:23 REC	Receive Error Counter Actual state of the Receive Error Counter, values between 0 and 127.
24:31 TEC	Transmit Error Counter Actual state of the Transmit Error Counter, values between 0 and 255.

Note: When CCCR.ASM is set, the CAN protocol controller does not increment TEC and REC when a CAN protocol error is detected, but CEL is still incremented.

42.3.5.2.13 Protocol Status Register (PSR)

Address: 0x0044 Access: RXS

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	TDCV						
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	PXE(1)	RFDF(1)	RBRS(1)	RESI(1)	DLEC(2)			BO	EW	EP	ACT		LEC(2)		
W																
Reset	0	0	0	0	0	1	1	1	0	0	0	0	0	1	1	1

1. Access type is RX: reset on read.
2. Access type is RS: set on read.

Figure 583. Protocol Status Register (PSR)

Table 644. PSR field descriptions

Field	Description
9:15 TDCV	Transmitter Delay Compensation Value (0x00-0x7F) Position of the secondary sample point, defined by the sum of the measured delay from CAN Tx to CAN Rx and TDCR.TDCO. The SSP position is, in the data phase, the number of mtq between the start of the transmitted bit and the secondary sample point. Valid values are 0 to 127 mtq.
17 PXE	Protocol Exception Event 0 No protocol exception event occurred since last read access 1 Protocol exception event occurred
18 RFDF	Received CAN FD Message This bit is set independent of acceptance filtering. 0 Since this bit was reset by the CPU, no CAN FD message has been received 1 Message in CAN FD format with FDF flag set has been received
19 RBRS	BRS flag of last received CAN FD Message This bit is set together with RFDF, independent of acceptance filtering. 0 Last received CAN FD message did not ha ve its BRS flag set. 1 Last received CAN FD message had its BRS flag set.
20 RESI	ESI CAN FD Message with ESI flag This bit is set together with RFDF, independent of acceptance filtering. 0 Last received CAN FD message did not have its ESI flag set. 1 Last received CAN FD message had its ESI flag set
21:23 DLEC	Data Phase Last Error Code Type of last error that occurred in the data phase of a CAN FD format frame with its BRS flag set. Coding is the same as for LEC. This field will be cleared to zero when a CAN FD format frame with its BRS flag set has been transferred (reception or transmission) without error.
24 BO	Bus_Off Status 0 The M_CAN is not Bus_Off 1 The M_CAN is in Bus_Off state
25 EW	Warning Status 0 Both error counters are below the Error_Warning limit of 96 1 At least one of error counter has reached the Error_Warning limit of 96
26 EP	Error Passive 0 The M_CAN is in the Error_Active state. It normally takes part in bus communication and sends an active error flag when an error has been detected 1 The M_CAN is in the Error_Passive state

Table 644. PSR field descriptions (continued)

Field	Description
27:28 ACT	Activity Monitors the module's CAN communication state. 00 Synchronizing – node is synchronizing on CAN communication 01 Idle-node is neither receiver nor transmitter 10 Receiver-node is operating as receiver 11 Transmitter-node is operating as transmitter
29:31 LEC	Last Error Code The LEC indicates the type of the last error to occur on the CAN bus. This field will be cleared to '0' when a message has been transferred (reception or transmission) without error. 000 No Error: No error occurred since LEC has been reset by successful reception or transmission. 001 Stuff Error: More than 5 equal bits in a sequence have occurred in a part of a received message where this is not allowed. 010 Form Error: A fixed format part of a received frame has the wrong format. 011 AckError: The message transmitted by the M_CAN was not acknowledged by another node. 100 Bit1Error: During the transmission of a message (with the exception of the arbitration field), the device wanted to send a recessive level (bit of logical value '1'), but the monitored bus value was dominant. 101 Bit0Error: During the transmission of a message (or acknowledge bit, or active error flag, or overload flag), the device wanted to send a dominant level (data or identifier bit logical value '0'), but the monitored bus value was recessive. During Bus_Off recovery this status is set each time a sequence of 11 recessive bits has been monitored. This enables the CPU to monitor the proceeding of the Bus_Off recovery sequence (indicating the bus is not stuck at dominant or continuously disturbed). 110 CRCError: The CRC check sum of a received message was incorrect. The CRC of an incoming message does not match with the CRC calculated from the received data. 111 NoChange: Any read access to the Protocol Status Register re-initializes the LEC to '7'. When the LEC shows the value '7', no CAN bus event was detected since the last CPU read access to the Protocol Status Register.

Note: When a frame in CAN FD format has reached the data phase with BRS flag set, the next CAN event (error or valid frame) will be shown in DLEC instead of LEC. An error in a fixed stuff bit of a CAN FD CRC sequence will be shown as a Form Error, not Stuff Error

The Bus_Off recovery sequence (see CAN Specification Rev. 2.0 or ISO11898-1) cannot be shortened by setting or resetting CCCR[INIT.] If the device goes Bus_Off, it will set CCCR[INIT] of its own accord, stopping all bus activities. Once CCCR[INIT] has been cleared by the CPU, the device will then wait for 129 occurrences of Bus Idle (129 × 11 consecutive recessive bits) before resuming normal operation. At the end of the Bus_Off recovery sequence, the Error Management Counters will be reset. During the waiting time after the resetting of CCCR[INIT], each time a sequence of 11 recessive bits has been monitored, a Bit 0 Error code is written to PSR[LEC], enabling the CPU to readily check up whether the CAN bus is stuck at dominant or continuously disturbed and to monitor the Bus_Off recovery sequence. ECR[REC] is used to count these sequences.

42.3.5.2.14 Transmitter Delay compensation Register (TDCR)

Address: 0x0048 Access: RP

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	TDCO(1)							0	TDCF(1)						
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. These are protected write (P) bits which means that write access by the bits is possible only when the bit 1 [CCE] and bit 0 [INIT] of CCCR register are set to ‘1’.

Figure 584. Transmitter Delay compensation Register (TDCR)

Table 645. TDCR field descriptions

Field	Description
17:23 TDCO	Transmitter Delay Compensation Offset (0x00-0x7F) Offset value defining the distance between the measured delay from CAN Tx to CAN Rx and the secondary sample point. Valid values are 0 to 127 mtq.
25:31 TDCF	Transmitter Delay Compensation Filter Window Length (0x00-0x7F) Defines the minimum value for the SSP position, dominant edges on CAN Rx that would result in an earlier SSP position are ignored for transmitter delay measurement. The feature is enabled when TDCF is configured to a value greater than TDCO. Valid values are 0 to 127 mtq.

42.3.5.2.15 Interrupt Register (IR)

The flags are set when one of the listed conditions is detected (edge-sensitive). The flags remain set until the Host clears them. A flag is cleared by writing a ‘1’ to the corresponding bit position. Writing a ‘0’ has no effect. A hard reset will clear the register. The configuration of IE controls whether an interrupt is generated. The configuration of ILS controls on which interrupt line an interrupt is signaled.

Address: 0x0050 Access: R/W

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	ARA	PED	PEA	WDI	BO	EW	EP	ELO	BEU	BEC	DRX	TOO	MRAF	TSW
W			w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TEFL	TEFF	TEFW	TEFN	TFE	TCF	TC	HPM	RF1L	RF1F	RF1W	RF1N	RF0L	RF0F	RF0W	RF0N
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 585. Interrupt Register (IR)

Table 646. IR field descriptions

Field	Description
2 ARA	Access to Reserved Address 0 No access to reserved address occurred 1 Access to reserved address occurred
3 PED	Protocol Error in Data Phase (Data Bit Time is used) 0 No protocol error in data phase 1 Protocol error in data phase detected (PSR.DLEC ≠ 0,7)
4 PEA	Protocol Error in Arbitration Phase (Nominal Bit Time is used) 0= No protocol error in arbitration phase 1= Protocol error in arbitration phase detected (PSR.LEC ≠ 0,7)
5 WDI	Watchdog Interrupt 0 No Message RAM Watchdog event occurred 1 Message RAM Watchdog event due to missing READY
6 BO	Bus_Off Status 0 Bus_Off status unchanged 1 Bus_Off status changed
7 EW	Warning Status 0 Error_Warning status unchanged 1 Error_Warning status changed
8 EP	Error Passive 0 Error_Passive status unchanged 1 Error_Passive status changed
9 ELO	Error Logging Overflow 0 CAN Error Logging Counter did not overflow 1 Overflow of CAN Error Logging Counter occurred

Table 646. IR field descriptions (continued)

Field	Description
10 BEU	Bit Error Uncorrected Message RAM bit error detected, uncorrected. Controlled by Message RAM bit error input signal generated by an optional external parity / ECC logic attached to the Message RAM. An uncorrected Message RAM bit error sets CCCR[INIT] to '1'. This is done to avoid transmission of corrupted data. 0 No bit error detected when reading from Message RAM 1 Bit error detected, uncorrected (for example parity logic)
11 BEC	Bit Error Corrected Message RAM bit error detected and corrected. Controlled by Message RAM bit error input signal generated by an optional external parity / ECC logic attached to the Message RAM. 0 No bit error detected when reading from Message RAM 1 Bit error detected and corrected (for example ECC)
12 DRX	Message stored to Dedicated Rx Buffer The flag is set whenever a received message has been stored into a dedicated Rx Buffer. 0 No Rx Buffer updated 1 At least one received message stored into a Rx Buffer
13 TOO	Timeout Occurred 0 No timeout 1 Timeout reached
14 MRAF	Message RAM Access Failure The flag is set, when the Rx Handler – Has not completed acceptance filtering or storage of an accepted message until the arbitration field of the following message has been received. In this case acceptance filtering or message storage is aborted and the Rx Handler starts processing of the following message. – Was not able to write a message to the Message RAM. In this case message storage is aborted. In both cases the FIFO put index is not updated resp. the New Data flag for a dedicated Rx Buffer is not set, a partly stored message is overwritten when the next message is stored to this location. The flag is also set when the Tx Handler is not able to read a message from the Message RAM in time. In this case message transmission is aborted. To leave Restricted Operation Mode, the Host CPU has to reset CCCR.ASM. 0 No Message RAM access failure occurred 1 Message RAM access failure occurred
15 TSW	Timestamp Wraparound 0 No timestamp counter wrap-around 1 Timestamp counter wrapped around
16 TEFL	Tx Event FIFO Event Lost 0 No Tx Event FIFO element lost 1 Tx Event FIFO element lost, also set after write attempt to Tx Event FIFO of size zero

Table 646. IR field descriptions (continued)

Field	Description
17 TEFF	Tx Event FIFO Full
18 TEFW	Tx Event FIFO Watermark Reached 0 Tx Event FIFO fill level below watermark 1 Tx Event FIFO fill level reached watermark
19 TEFN	Tx Event FIDO New Entry 0 Tx Event FIFO unchanged 1 Tx Handler wrote Tx Event FIFO element
20 TFE	Tx FIFO Empty 0 Tx FIFO non-empty 1 Tx FIFO empty
21 TCF	Transmission Cancellation Finished 0 No transmission cancellation finished 1 Transmission cancellation finished
22 TC	Transmission Completed 0 No transmission completed 1 Transmission completed
23 HPM	High Priority Message 0 No high priority message received 1 High priority message received
24 RF1L	Rx FIFO 1 Message Lost 0 No Rx FIFO 1 message lost 1 Rx FIFO 1 message lost, also set after write attempt to Rx FIFO 1 of size zero
25 RF1F	Rx FIFO 1 Full 0 Rx FIFO 1 not full 1 Rx FIFO 1 full
26 RF1W	Rx FIFO 1 Watermark Reached 0 Rx FIFO 1 fill level below watermark 1 Rx FIFO 1 fill level reached watermark
27 RF1N	Rx FIFO 1 New Message 0 No new message written to Rx FIFO 1 1 New message written to Rx FIFO 1
28 RF0L	Rx FIFO 0 Message Lost 0 No Rx FIFO 0 message lost 1 Rx FIFO 0 message lost, also set after write attempt to Rx FIFO 0 of size zero
29 RF0F	Rx FIFO 0 Full 0 Rx FIFO 0 not full 1 Rx FIFO 0 full

Table 646. IR field descriptions (continued)

Field	Description
30 RF0W	Rx FIFO 0 Watermark Reached 0 Rx FIFO 0 fill level below watermark 1 Rx FIFO 0 fill level reached watermark
31 RF0N	Rx FIFO 0 New Message 0 No new message written to Rx FIFO 0 1 New message written to Rx FIFO 0

42.3.5.2.16 Interrupt Enable Register (IE)

The settings in the Interrupt Enable register determine which status changes in the Interrupt register will be signaled on an interrupt line.

Address: 0x0054 Access: R/W

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	ARAE	PEDE	PEAE	WDIE	BOE	EWE	EPE	ELOE	BEUE	BECE	DRXE	TOOE	MRAFE	TSWE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TEFLE	TEFFE	TEFWE	TEFNE	TFEE	TCFE	TCE	HPME	RF1LE	RF1FE	RF1WE	RF1NE	RF0LE	RF0FE	RF0WE	RF0NE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 586. Interrupt Enable Register (IE)

Table 647. IE field descriptions

Field	Description
0:1	Reserved
2 ARAE	Access to Reserved Address Enable 0 Interrupt disabled 1 Interrupt enabled
3 PEDE	Protocol Error in Data Phase Enable 0 Interrupt disabled 1 Interrupt enabled
4 PEAE	Protocol Error in Arbitration Phase Enable 0 Interrupt disabled 1 Interrupt enabled
5 WDIE	Watchdog Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled

Table 647. IE field descriptions (continued)

Field	Description
6 BOE	Bus_Off Status Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
7 EWE	Warning Status Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
8 EPE	Error Passive Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
9 ELOE	Error Logging Overflow Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
10 BEUE	Bit Error Uncorrected Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
11 BECE	Bit Error Corrected Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
12 DRXE	Message stored to Dedicated Rx Buffer Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
13 TOOE	Timeout Occurred Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
14 MRAFE	Message RAM Access Failure Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
15 TSWE	Timestamp Wraparound Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
16 TEFLE	Tx Event FIFO Event Lost Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
17 TEFFE	Tx Event FIFO Full Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
18 TEFWE	Tx Event FIFO Watermark Reached Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
19 TEFNE	Tx Event FIFO New Entry Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled

Table 647. IE field descriptions (continued)

Field	Description
20 TFEE	Tx FIFO Empty Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
21 TCFE	Transmission Cancellation Finished Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
22 TCE	Transmission Completed Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
23 HPME	High Priority Message Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
24 RF1LE	Rx FIFO 1 Message Lost Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
25 RF1FE	Rx FIFO 1 Full Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
26 RF1WE	Rx FIFO 1 Watermark Reached Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
27 RF1NE	Rx FIFO 1 New Message Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
28 RF0LE	Rx FIFO 0 Message Lost Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
29 RF0FE	Rx FIFO 0 Full Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
30 RF0WE	Rx FIFO 0 Watermark Reached Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
31 RF0NE	Rx FIFO 0 New Message Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled

42.3.5.2.17 Interrupt Line Select Register (ILS)

The Interrupt Line Select register assigns an interrupt generated by a specific interrupt flag from the Interrupt Register to one of the two module interrupt lines.

Address: 0x0058 Access: R/W

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	ARAL	PEDL	PEAL	WDIL	BOL	EWL	EPL	ELOL	BEUL	BECL	DRXL	TOOL	MRAFL	TSWL
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TEFLL	TEFFL	TEFWL	TEFNL	TFEL	TCFL	TCL	HPML	RF1LL	RF1FL	RF1WL	RF1NL	RF0LL	RF0FL	RF0WL	RF0NL
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 587. Interrupt Line Select Register (ILS)

Table 648. ILS field descriptions

Field	Description
0:1	Reserved
2 ARAL	Access to Reserved Address Line 0 Interrupt assigned to interrupt line M_CAN INT0 1 Interrupt assigned to interrupt line M_CAN INT1
3 PEDL	Protocol Error in Data Phase Line 0 Interrupt assigned to interrupt line M_CAN INT0 1 Interrupt assigned to interrupt line M_CAN INT1
4 PEAL	Protocol Error in Arbitration Phase Line 0 Interrupt assigned to interrupt line M_CAN INT0 1 Interrupt assigned to interrupt line M_CAN INT1
5 WDIL	Watchdog Interrupt Line 0 Interrupt assigned to interrupt line M_CAN INT0 1 Interrupt assigned to interrupt line M_CAN INT1
6 BOL	Bus_Off Status Interrupt Line 0 Interrupt assigned to interrupt line M_CAN INT0 1 Interrupt assigned to interrupt line M_CAN INT1
7 EWL	Warning Status Interrupt Line 0 Interrupt assigned to interrupt line M_CAN INT0 1 Interrupt assigned to interrupt line M_CAN INT1
8 EPL	Error Passive Interrupt Line 0 Interrupt assigned to interrupt line M_CAN INT0 1 Interrupt assigned to interrupt line M_CAN INT1
9 ELOL	Error Logging Overflow Interrupt Line 0 Interrupt assigned to interrupt line M_CAN INT0 1 Interrupt assigned to interrupt line M_CAN INT1
10 BEUL	Bit Error Uncorrected Interrupt Line 0 Interrupt assigned to interrupt line M_CAN INT0 1 Interrupt assigned to interrupt line M_CAN INT1

Table 648. ILS field descriptions (continued)

Field	Description
11 BECL	Bit Error Corrected Interrupt Line 0 Interrupt assigned to interrupt line M_CAN INT0 1 Interrupt assigned to interrupt line M_CAN INT1
12 DRXL	Message stored to Dedicated Rx Buffer Interrupt Line 0 Interrupt assigned to interrupt line M_CAN INT0 1 Interrupt assigned to interrupt line M_CAN INT1
13 TOOL	Timeout Occurred Interrupt Line 0 Interrupt assigned to interrupt line M_CAN INT0 1 Interrupt assigned to interrupt line M_CAN INT1
14 MRAFL	Message RAM Access Failure Interrupt Line 0 Interrupt assigned to interrupt line M_CAN INT0 1 Interrupt assigned to interrupt line M_CAN INT1
15 TSWL	TSWL: Timestamp Wraparound Interrupt Line 0 Interrupt assigned to interrupt line M_CAN INT0 1 Interrupt assigned to interrupt line M_CAN INT1
16 TEFLL	Tx Event FIFO Event Lost Interrupt Line 0 Interrupt assigned to interrupt line M_CAN INT0 1 Interrupt assigned to interrupt line M_CAN INT1
17 TEFFL	Tx Event FIFO Full Interrupt Line 0 Interrupt assigned to interrupt line M_CAN INT0 1 Interrupt assigned to interrupt line M_CAN INT1
18 TEFWL	Tx Event FIFO Watermark Reached Interrupt Line 0 Interrupt assigned to interrupt line M_CAN INT0 1 Interrupt assigned to interrupt line M_CAN INT1
19 TEFNL	Tx Event FIFO New Entry Interrupt Line 0 Interrupt assigned to interrupt line M_CAN INT0 1 Interrupt assigned to interrupt line M_CAN INT1
20 TFEL	Tx FIFO Empty Interrupt Line 0 Interrupt assigned to interrupt line M_CAN INT0 1 Interrupt assigned to interrupt line M_CAN INT1
21 TCFL	Transmission Cancellation Finished Interrupt Line 0 Interrupt assigned to interrupt line M_CAN INT0 1 Interrupt assigned to interrupt line M_CAN INT1
22 TCL	Transmission Completed Interrupt Line 0 Interrupt assigned to interrupt line M_CAN INT0 1 Interrupt assigned to interrupt line M_CAN INT1
23 HPML	High Priority Message Interrupt Line 0 Interrupt assigned to interrupt line M_CAN INT0 1 Interrupt assigned to interrupt line M_CAN INT1
24 RF1LL	Rx FIFO 1 Message Lost Interrupt Line 0 Interrupt assigned to interrupt line M_CAN INT0 1 Interrupt assigned to interrupt line M_CAN INT1

Table 648. ILS field descriptions (continued)

Field	Description
25 RF1FL	Rx FIFO 1 Full Interrupt Line 0 Interrupt assigned to interrupt line M_CAN INT0 1 Interrupt assigned to interrupt line M_CAN INT1
26 RF1WL	Rx FIFO 1 Watermark Reached Interrupt Line 0 Interrupt assigned to interrupt line M_CAN INT0 1 Interrupt assigned to interrupt line M_CAN INT1
27 RF1NL	Rx FIFO 1 New Message Interrupt Line 0 Interrupt assigned to interrupt line M_CAN INT0 1 Interrupt assigned to interrupt line M_CAN INT1
28 RF0LL	Rx FIFO 0 Message Lost Interrupt Line 0 Interrupt assigned to interrupt line M_CAN INT0 1 Interrupt assigned to interrupt line M_CAN INT1
29 RF0FL	Rx FIFO 0 Full Interrupt Line 0 Interrupt assigned to interrupt line M_CAN INT0 1 Interrupt assigned to interrupt line M_CAN INT1
30 RF0WL	Rx FIFO 0 Watermark Reached Interrupt Line 0 Interrupt assigned to interrupt line M_CAN INT0 1 Interrupt assigned to interrupt line M_CAN INT1
31 RF0NL	Rx FIFO 0 New Message Interrupt Line 0 Interrupt assigned to interrupt line M_CAN INT0 1 Interrupt assigned to interrupt line M_CAN INT1

42.3.5.2.18 Interrupt Line Enable Register (ILE)

Each of the two interrupt lines to the CPU can be enabled / disabled separately by programming bits EINT0 and EINT1

Address: 0x005C Access: R/W

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	EINT1	EINT0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 588. Interrupt Line Enable Register (ILE)

Table 649. ILE field descriptions

Field	Description
30 EINT1	Enable Interrupt Line 1 0 Interrupt line disabled 1 Interrupt line enabled
31 EINT0	Enable Interrupt Line 0 0 Interrupt line disabled 1 bInterrupt line enabled

42.3.5.2.19 Global Filter Configuration Register (GFC)

Global settings for Message ID filtering. The Global Filter Configuration controls the filter path for standard and extended messages as described in *Figure 627* and *Figure 628*.

Address: 0x0080 Access: RP

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	ANFS(1)		ANFE(1)		RRFS(1)	RRFE(1)
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. These are protected write (P) bits which means that write access by the bits is possible only when the bit 1 [CCE] and bit 0 [INIT] of CCCR register are set to '1'.

Figure 589. Global Filter Configuration Register (GFC)

Table 650. GFC field descriptions

Field	Description
26:27 ANFS	Accept Non-matching Frames Standard Defines how received messages with 11-bit IDs that do not match any element of the filter list are treated. 00 Accept in Rx FIFO 0 01 Accept in Rx FIFO 1 10 Reject 11 Reject
28:29 ANFE	ANFE[1:0]: Accept Non-matching Frames Extended Defines how received messages with 29-bit IDs that do not match any element of the filter list are treated. 00 Accept in Rx FIFO 0 01 Accept in Rx FIFO 1 10 Reject 11 Reject

Table 650. GFC field descriptions (continued)

Field	Description
30 RRFS	Reject Remote Frames Standard 0 Filter remote frames with 11-bit standard IDs 1 Reject all remote frames with 11-bit standard IDs
31 RRFE	Reject Remote Frames Extended 0 Filter remote frames with 29-bit extended IDs 1 Reject all remote frames with 29-bit extended IDs

42.3.5.2.20 Standard ID Filter Configuration Register (SIDFC)

Settings for 11-bit standard Message ID filtering. The Standard ID Filter Configuration controls the filter path for standard messages as described in *Figure 621*.

Address: 0x0084 Access: RP

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	LSS(1)							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	FLSSA(1)														0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. These are protected write (P) bits which means that write access by the bits is possible only when the bit 1 [CCE] and bit 0 [INIT] of CCCR register are set to '1'.

Figure 590. Standard ID Filter Configuration register (SIDFC)

Table 651. SIDFC field descriptions

Field	Description
8:15 LSS	List Size Standard 0x00: No standard Message ID filter 0x01: Number of standard Message ID filter elements ... 0x80: Number of standard Message ID filter elements 0x81: Values greater than 128 are interpreted as 128 ... 0xFF: Values greater than 128 are interpreted as 128
16:29 FLSSA	Filter List Standard Start Address Start address of standard Message ID filter list (32-bit word address, see *Figure 617*).

42.3.5.2.21 Extended ID Filter Configuration Register (XIDFC)

Settings for 29-bit extended Message ID filtering. The Extended ID Filter Configuration controls the filter path for standard messages as described in *Figure 622*.

Address: 0x0088 Access: RP

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	LSE(1)						
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	FLESA(1)														0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. These are protected write (P) bits which means that write access by the bits is possible only when the bit 1 [CCE] and bit 0 [INIT] of CCCR register are set to ‘1’.

Figure 591. Extended ID Filter Configuration register (XIDFC)

Table 652. XIDFC field descriptions

Field	Description
9:15 LSE	List Size Extended 0x00: No extended Message ID filter 0x01: Number of extended Message ID filter elements ... 0x40: Number of extended Message ID filter elements 0x41: Values greater than 64 are interpreted as 64 ... 0x3F: Values greater than 64 are interpreted as 64
16:29 FLESA	Filter List Extended Start Address Start address of extended Message ID filter list (32-bit word address, see *Figure 617*).

42.3.5.2.22 Extended ID and Mask Register (XIDAM)

Address: 0x0090 Access: RP

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	EIDM(1)												
W																
Reset	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	EIDM(1)															
W																
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

1. These are protected write (P) bits which means that write access by the bits is possible only when the bit 1 [CCE] and bit 0 [INIT] of CCCR register are set to ‘1’.

Figure 592. Extended ID and Mask Register (XIDAM)

Table 653. XIDAM field descriptions

Field	Description
3:31 EIDM	Extended ID Mask For acceptance filtering of extended frames the Extended ID AND Mask is ANDed with the Message ID of a received frame. Intended for masking of 29-bit IDs in SAE J1939. With the reset value of all bits set to one the mask is not active.

42.3.5.2.23 High Priority Message Status Register (HPMS)

This register is updated every time a Message ID filter element configured to generate a priority event matches. This can be used to monitor the status of incoming high priority messages and to enable fast access to these messages.

Address: 0x0094 Access: R

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	FLST	FIDX							MSI		BIDX					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 593. High Priority Message Status Register (HPMS)

Table 654. HPMS field descriptions

Field	Description
16 FLST	Filter List Indicates the filter list of the matching filter element. 0 Standard Filter List 1 Extended Filter List
17:23 FIDX	Filter Index Index of matching filter element. Range is 0 to SIDFC[LSS] - 1 resp. XIDFC[LSE] - 1.
24:25 MSI	Message Storage Indicator 00 No FIFO selected 01 FIFO overrun 10 Message stored in FIFO 0 11 Message stored in FIFO 1
26:31 BIDX	Buffer Index Index of Rx FIFO element to which the message was stored. Only valid when MSI[1] = ‘1’.

42.3.5.2.24 New Data 1 Register (NDAT1)

Address: 0x0098 Access: R/W

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	ND31	ND30	ND29	ND28	ND27	ND26	ND25	ND24	ND23	ND22	ND21	ND20	ND19	ND18	ND17	ND16
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	ND15	ND14	ND13	ND12	ND11	ND10	ND9	ND8	ND7	ND6	ND5	ND4	ND3	ND2	ND1	ND0
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 594. New Data 1 Register (NDAT1)

Table 655. NDAT1 field descriptions

Field	Description
0:31 ND[31:0]	New Data[31:0] The register holds the New Data flags of Rx Buffers 0 to 31. The flags are set when the respective Rx Buffer has been updated from a received frame. The flags remain set until the Host clears them. A flag is cleared by writing a ‘1’ to the corresponding bit position. Writing a ’0’ has no effect. A hard reset will clear the register. 0 Rx Buffer not updated 1 Rx Buffer updated from new message

42.3.5.2.25 New Data 2 Register (NDAT2)

Address: 0x009C Access: R/W

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	ND63	ND62	ND61	ND60	ND59	ND58	ND57	ND56	ND55	ND54	ND53	ND52	ND51	ND50	ND49	ND48
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	ND47	ND46	ND45	ND44	ND43	ND42	ND41	ND40	ND39	ND38	ND37	ND36	ND35	ND34	ND33	ND32
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 595. New Data 2 Register (NDAT2)

Table 656. NDAT2 field descriptions

Field	Description
0:31 ND[63:32]	New Data[63:32] The register holds the New Data flags of Rx Buffers 32 to 63. The flags are set when the respective Rx Buffer has been updated from a received frame. The flags remain set until the Host clears them. A flag is cleared by writing a '1' to the corresponding bit position. Writing a '0' has no effect. A hard reset will clear the register. 0 Rx Buffer not updated 1 Rx Buffer updated from new message

42.3.5.2.26 Rx FIFO 0 Configuration Register (RXF0C)

Address: 0x00A0 Access: RP

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	F0OM	F0WM(1)							0	F0S(1)						
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	F0SA(1)														0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. These are protected write (P) bits which means that write access by the bits is possible only when the bit 1 [CCE] and bit 0 [INIT] of CCCR register are set to '1'.

Figure 596. Rx FIFO 0 Configuration Register (RXF0C)

Table 657. RXF0C field descriptions

Field	Description
0 F0OM	FIFO 0 Operation Mode FIFO 0 can be operated in blocking or in overwrite mode. 0 FIFO 0 blocking mode 1 FIFO 0 overwrite mode
1:7 F0WM	Rx FIFO 0 Watermark 0x00: Watermark interrupt disabled 0x01: Level for Rx FIFO 0 watermark interrupt (IR[RF0W]) 0x40: Level for Rx FIFO 0 watermark interrupt (IR[RF0W]) 0x41: Watermark interrupt disabled 0x3F: Watermark interrupt disabled

Table 657. RXF0C field descriptions (continued)

Field	Description
9:15 F0S	Rx FIFO 0 Size 0x00: No Rx FIFO 0 0x01: Number of Rx FIFO 0 elements ... 0x40: Number of Rx FIFO 0 elements 0x41: Values greater than 64 are interpreted as 64 ... 0x3F: Values greater than 64 are interpreted as 64 The Rx FIFO 0 elements are indexed from 0 to F0S-1.
16:29 F0SA	Rx FIFO 0 Start Address Start address of Rx FIFO 0 in Message RAM (32-bit word address, *Figure 617*).

42.3.5.2.27 Rx FIFO 0 Status Register (RXF0S)

Address: 0x00A4 Access: R

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	RF0L	F0F	0	0	F0PI					
W	0															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	F0GI						0	F0FL						
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 597. Rx FIFO 0 Status Register (RXF0S)

Table 658. RXF0S field descriptions

Field	Description
6 RF0L	Rx FIFO 0 Message Lost This bit is a copy of interrupt flag IR[RF0L]. When IR[RF0L] is reset, this bit is also reset. 0 No Rx FIFO 0 message lost 1 Rx FIFO 0 message lost, also set after write attempt to Rx FIFO 0 of size zero **Note:** Overwriting the oldest message when RXF0C.F0OM = '1' will not set this flag.
7 F0F	Rx FIFO 0 Full 0 Rx FIFO 0 not full 1 Rx FIFO 0 full
10:15 F0PI	Rx FIFO 0 Put Index Rx FIFO 0 write index pointer, range 0 to 63.

Table 658. RXF0S field descriptions (continued)

Field	Description
18:23 F0GI	Rx FIFO 0 Get Index Rx FIFO 0 read index pointer, range 0 to 63.
25:31 F0FL	Rx FIFO 0 Fill Level Number of elements stored in Rx FIFO 0, range 0 to 64.

42.3.5.2.28 Rx FIFO 0 Acknowledge Register (RXF0A)

Address: 0x00A8 Access: R/W

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	F0AI					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 598. Rx FIFO 0 Acknowledge Register (RXF0A)

Table 659. RXF0A field descriptions

Field	Description
26:31 F0AI	Rx FIFO 0 Acknowledge Index After the Host has read a message or a sequence of messages from Rx FIFO 0 it has to write the buffer index of the last element read from Rx FIFO 0 to F0AI. This will set the Rx FIFO 0 Get Index RXF0S[F0GI] to F0AI + 1 and update the FIFO 0 Fill Level RXF0S[F0FL].

42.3.5.2.29 Rx Buffer Configuration Register (RXBC)

Address: 0x00AC Access: RP

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	RBSA[1]														0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. These are protected write (P) bits which means that write access by the bits is possible only when the bit 1 [CCE] and bit 0 [INIT] of CCCR register are set to ‘1’.

Figure 599. Rx Buffer Configuration Register (RXBC)

Table 660. RXBC field descriptions

Field	Description
16:29 RBSA	Rx Buffer Start Address Configures the start address of the Rx Buffers section in the Message RAM (32-bit word address). Also used to reference debug messages A,B,C.

42.3.5.2.30 Rx FIFO 1 Configuration Register (RXF1C)

Address: 0x00B0 Access: RP

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	F1OM[1]	F1WM[1]							0	F1S[1]						
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	F1SA[1]														0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. These are protected write (P) bits which means that write access by the bits is possible only when the bit 1 [CCE] and bit 0 [INIT] of CCCR register are set to ‘1’.

Figure 600. Rx FIFO 1 Configuration Register (RXF1C)

Table 661. RXF1C field descriptions

Field	Description
0 F1OM	FIFO 1 Operation Mode FIFO 1 can be operated in blocking or in overwrite mode. 0 FIFO 1 blocking mode 1 FIFO 1 overwrite mode
1:7 F1WM	Rx FIFO 1 Watermark 0x00: Watermark interrupt disabled 0x01: Level for Rx FIFO 1 watermark interrupt IR[RF1W] ... 0x40: Level for Rx FIFO 1 watermark interrupt IR[RF1W] 0x41: Watermark interrupt disabled ... 0x7F: Watermark interrupt disabled
9:15 F1S	F1S[6:0]: Rx FIFO 1 Size 0x00: No Rx FIFO 1 0x01: Number of Rx FIFO 1 elements ... 0x40: Number of Rx FIFO 1 elements 0x41: Values greater than 64 are interpreted as 64 ... 0x7F: Values greater than 64 are interpreted as 64 The Rx FIFO 1 elements are indexed from 0 to F1S - 1
16:29 F1SA	Rx FIFO 1 Start Address Start address of Rx FIFO 1 in Message RAM (32-bit word address, see *Figure 617*).

42.3.5.2.31 Rx FIFO 1 Status Register (RXF1S)

Address: 0x00B4 Access: R

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DMS		0	0	0	0	RF1L	F1F	0	0	F1PI					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	F1GI						0	F1FL						
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 601. Rx FIFO 1 Status Register (RXF1S)

Table 662. RXF1S field descriptions

Field	Description
0:1 DMS	Debug Message Status 00 Idle state, wait for reception of debug messages, DMA request is cleared 01 Debug message A received 10 Debug messages A, B received 11 Debug messages A, B, C received, DMA request is set
6 RF1L	Rx FIFO 1 Message Lost This bit is a copy of interrupt flag IR[RF1L]. When IR[RF1L] is reset, this bit is also reset. 0 No Rx FIFO 1 message lost 1 Rx FIFO 1 message lost, also set after write attempt to Rx FIFO 1 of size zero **Note:** Overwriting the oldest message when RXF1C.F1OM = '1' will not set this flag.
7 F1F	Rx FIFO 1 Full 0 Rx FIFO 1 not full 1 Rx FIFO 1 full
10:15 F1PI	Rx FIFO 1 Put Index Rx FIFO 1 write index pointer, range 0 to 63.
18:23 F1GI	Rx FIFO 1 Get Index Rx FIFO 1 read index pointer, range 0 to 63.
25:31 F1FL	Rx FIFO 1 Fill Level Number of elements stored in Rx FIFO 1, range 0 to 64.

42.3.5.2.32 Rx FIFO 1 Acknowledge Register (RXF1A)

Address: 0x00B8 Access: R/W

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	F1AI					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 602. Rx FIFO 1 Acknowledge Register (RXF1A)

Table 663. RXF1A field descriptions

Field	Description
26:31 F1AI	Rx FIFO 1 Acknowledge Index After the Host has read a message or a sequence of messages from Rx FIFO 1 it has to write the buffer index of the last element read from Rx FIFO 1 to F1AI. This will set the Rx FIFO 1 Get Index RXF1S[F1GI] to F1AI + 1 and update the FIFO 1 Fill Level RXF1S[F1FL].

42.3.5.2.33 Rx Buffer / FIFO Element Size Configuration (RXESC)

Configures the number of data bytes belonging to an Rx Buffer / Rx FIFO element. Data field sizes >8 bytes are intended for CAN FD operation only.

Address: 0x00BC Access: RP

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	RBDS(1)				F1DS(1)				F0DS(1)		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. These are protected write (P) bits which means that write access by the bits is possible only when the bit 1 [CCE] and bit 0 [INIT] of CCCR register are set to '1'.

Figure 603. Rx Buffer / FIFO Element Size Configuration Register (RXESC)

Table 664. RXESC field descriptions

Field	Description
21:23 RBDS	Rx Buffer Data Field Size 000 8 byte data field 001 12 byte data field 010 16 byte data field 011 20 byte data field 100 24 byte data field 101 32 byte data field 110 48 byte data field 111 64 byte data field
25:27 F1DS	Rx FIFO 1 Data Field Size 000 8 byte data field 001 12 byte data field 010 16 byte data field 011 20 byte data field 100 24 byte data field 101 32 byte data field 110 48 byte data field 111 64 byte data field
29:31 F0DS	Rx FIFO 0 Data Field Size 000 8 byte data field 001 12 byte data field 010 16 byte data field 011 20 byte data field 100 24 byte data field 101 32 byte data field 110 48 byte data field 111 64 byte data field

Note: In case the data field size of an accepted CANframe exceeds the data field size configured for the matching Rx Buffer or Rx FIFO, only the number of bytes as configured by RXESC are stored to the Rx Buffer respectively, Rx FIFO element. The rest of the frame's data field is ignored.

42.3.5.2.34 Tx Buffer Configuration Register (TXBC)

Address: 0x00C0 Access: RP

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	TFQM(1)	TFQS(1)						0	0	NDTB(1)					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TBSA(1)														0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. These are protected write (P) bits which means that write access by the bits is possible only when the bit 1 [CCE] and bit 0 [INIT] of CCCR register are set to '1'.

Figure 604. Tx Buffer Configuration Register (TXBC)

Table 665. TXBC field descriptions

Field	Description
1 TFQM	Tx FIFO/Queue Mode 0 Tx FIFO operation 1 Tx Queue operation
2:7 TFQS	Tx FIFO/Queue Size 0x00: No Tx FIFO/Queue 0x01: Number of Tx Buffers used for Tx FIFO/Queue ... 0x20: Number of Tx Buffers used for Tx FIFO/Queue 0x21: Values greater than 32 are interpreted as 32 ... 0x3F: Values greater than 32 are interpreted as 32
10:15 NDTB	Number of Dedicated Transmit Buffers 0x00: No Dedicated Tx Buffers 0x01: Number of Dedicated Tx Buffers ... 0x20: Number of Dedicated Tx Buffers 0x21: Values greater than 32 are interpreted as 32 ... 0x3F: Values greater than 32 are interpreted as 32
16:29 TBSA	Tx Buffers Start Address Start address of Tx Buffers section in Message RAM (32-bit word address, see *Figure 617*).

Note: Be aware that the sum of TFQS and NDTB may be not greater than 32. There is no check for erroneous configurations. The Tx Buffers section in the Message RAM starts with the dedicated Tx Buffers.

42.3.5.2.35 Tx FIFO/Queue Status Register (TXFQS)

Address: 0x00C4 Access: R

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	TFQF	TFQPI				
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	TFGI					0	0	TFFL					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 605. Tx FIFO/Queue Status Register (TXFQS)

Table 666. TXFQS field descriptions

Field	Description
10 TFQF	Tx FIFO/Queue Full 0 Tx FIFO/Queue not full 1 Tx FIFO/Queue full
11:15 TFQPI	Tx FIFO/Queue Put Index Tx FIFO/Queue write index pointer, range 0 to 31.
19:23 TFGI	Tx FIFO Get Index Tx FIFO read index pointer, range 0 to 31. Read as zero when Tx Queue operation is configured (TXBC[TFQM] = ‘1’).
26:31 TFFL	Tx FIFO Free Level Number of consecutive free Tx FIFO elements starting from TFGI, range 0 to 32. Read as zero when Tx Queue operation is configured (TXBC[TFQM] = ‘1’).

Note: In case of mixed configurations where dedicated Tx Buffers are combined with a Tx FIFO or a Tx Queue, the Put and Get Indices indicate the number of the Tx Buffer starting with the first dedicated Tx Buffers. Example: For a configuration of 12 dedicated Tx Buffers and a Tx FIFO of 20 Buffers a Put Index of 15 points to the fourth buffer of the Tx FIFO.

42.3.5.2.36 Tx Buffer Element Size Configuration (TXESC)

Configures the number of data bytes belonging to a Tx Buffer element. Data field sizes > 8 bytes are intended for CAN FD operation only.

Address: 0x00C8 Access: RP

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	TBDS(1)		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. These are protected write (P) bits which means that write access by the bits is possible only when the bit 1 [CCE] and bit 0 [INIT] of CCCR register are set to ‘1’.

Figure 606. Tx Buffer Element Size Configuration Register (TXESC)

Table 667. TXESC field descriptions

Field	Description
29:31 TBDS	Tx Buffer Data Field Size 000 8 byte data field 001 12 byte data field 010 16 byte data field 011 20 byte data field 100 24 byte data field 101 32 byte data field 110 48 byte data field 111 64 byte data field **Note:** In case the data length code DLC of a Tx Buffer element is configured to a value higher than the Tx Buffer data field size TXESC.TBDS, the bytes not defined by the Tx Buffer are transmitted as ‘0xCC’ (padding bytes).

42.3.5.2.37 Tx Buffer Request Pending Register (TXBRP)

Address: 0x00CC Access: R

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	TRP31	TRP30	TRP29	TRP28	TRP27	TRP26	TRP25	TRP24	TRP23	TRP22	TRP21	TRP20	TRP19	TRP18	TRP17	TRP16
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TRP15	TRP14	TRP13	TRP12	TRP11	TRP10	TRP9	TRP8	TRP7	TRP6	TRP5	TRP4	TRP3	TRP2	TRP1	TRP0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 607. Tx Buffer Request Pending Register (TXBRP)

Table 668. TXBRP field descriptions

Field	Description
0:31 TRP[31:0]	Transmission Request Pending Each Tx Buffer has its own Transmission Request Pending bit. The bits are set via register TXBAR. The bits are reset after a requested transmission has completed or has been cancelled via register TXBCR. TXBRP bits are set only for those Tx Buffers configured via TXBC. After a TXBRP bit has been set, a Tx scan is started to check for the pending Tx request with the highest priority (Tx Buffer with lowest Message ID). A cancellation request resets the corresponding transmission request pending bit of register TXBRP. In case a transmission has already been started when a cancellation is requested, this is done at the end of the transmission, regardless whether the transmission was successful or not. The cancellation request bits are reset directly after the corresponding TXBRP bit has been reset. After a cancellation has been requested, a finished cancellation is signaled via TXBCF – After successful transmission together with the corresponding TXBTO bit – When the transmission has not yet been started at the point of cancellation – When the transmission has been aborted due to lost arbitration – When an error occurred during frame transmission In DAR mode all transmissions are automatically cancelled if they are not successful. The corresponding TXBCF bit is set for all unsuccessful transmissions. 0 No transmission request pending 1 Transmission request pending

Note: TXBRP bits which are set while a Tx scan is in progress are not considered during this particular Tx scan. In case a cancellation is requested for such a Tx Buffer, this Add Request is cancelled immediately, the corresponding TXBRP bit is reset.

42.3.5.2.38 Tx Buffer Add Request Register (TXBAR)

Address: 0x00D0 Access: R/W

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	AR31	AR30	AR29	AR28	AR27	AR26	AR25	AR24	AR23	AR22	AR21	AR20	AR19	AR18	AR17	AR16
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	AR15	AR14	AR13	AR12	AR11	AR10	AR9	AR8	AR7	AR6	AR5	AR4	AR3	AR2	AR1	AR0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 608. Tx Buffer Add Request Register (TXBAR)

Table 669. TXBAR field descriptions

Field	Description
0:31 AR[31:0]	Add Request Each Tx Buffer has its own Add Request bit. Writing a '1' will set the corresponding Add Request bit; writing a '0' has no impact. This enables the Host to set transmission requests for multiple Tx Buffers with one write to TXBAR. TXBAR bits are set only for those Tx Buffers configured via TXBC. When no Tx scan is running, the bits are reset immediately, else the bits remain set until the Tx scan process has completed. 0 No transmission request added 1 Transmission requested added

Note: If an add request is applied for a Tx Buffer with pending transmission request (corresponding TXBRP bit already set), this add request is ignored.

42.3.5.2.39 Tx Buffer Cancellation Request Register (TXBCR)

Address: 0x00D4 Access: R/W

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	CR31	CR30	CR29	CR28	CR27	CR26	CR25	CR24	CR23	CR22	CR21	CR20	CR19	CR18	CR17	CR16
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	CR15	CR14	CR13	CR12	CR11	CR10	CR9	CR8	CR7	CR6	CR5	CR4	CR3	CR2	CR1	CR0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 609. Tx Buffer Cancellation Request Register (TXBCR)

Table 670. TXBCR field descriptions

Field	Description
0:31 CR[31:0]	Cancellation Request Each Tx Buffer has its own Cancellation Request bit. Writing a '1' will set the corresponding Cancellation Request bit; writing a '0' has no impact. This enables the Host to set cancellation requests for multiple Tx Buffers with one write to TXBCR. TXBCR bits are set only for those Tx Buffers configured via TXBC. The bits remain set until the corresponding bit of TXBRP is reset. 0 No cancellation pending 1 Cancellation pending

42.3.5.2.40 Tx Buffer Transmission Occurred Register (TXBTO)

Address: 0x00D8 Access: Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	TO31	TO30	TO29	TO28	TO27	TO26	TO25	TO24	TO23	TO22	TO21	TO20	TO19	TO18	TO17	TO16
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TO15	TO14	TO13	TO12	TO11	TO10	TO9	TO8	TO7	TO6	TO5	TO4	TO3	TO2	TO1	TO0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 610. Tx Buffer Transmission Occurred Register (TXBTO)

Table 671. TXBTO field descriptions

Field	Description
0:31 TO[31:0]	Transmission Occurred Each Tx Buffer has its own Transmission Occurred bit. The bits are set when the corresponding TXBRP bit is cleared after a successful transmission. The bits are reset when a new transmission is requested by writing a ‘1’ to the corresponding bit of register TXBAR. 0 No transmission occurred 1 Transmission occurred

42.3.5.2.41 Tx Buffer Cancellation Finished Register (TXBCF)

Address: 0x00DC Access: Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	CF31	CF30	CF29	CF28	CF27	CF26	CF25	CF24	CF23	CF22	CF21	CF20	CF19	CF18	CF17	CF16
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	CF15	CF14	CF13	CF12	CF11	CF10	CF9	CF8	CF7	CF6	CF5	CF4	CF3	CF2	CF1	CF0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 611. Tx Buffer Cancellation Finished Register (TXBCF)

Table 672. TXBCF field descriptions

Field	Description
0:31 CF[31:0]	Cancellation Finished Each Tx Buffer has its own Cancellation Finished bit. The bits are set when the corresponding TXBRP bit is cleared after a cancellation was requested via TXBCR. In case the corresponding TXBRP bit was not set at the point of cancellation, CF is set immediately. The bits are reset when a new transmission is requested by writing a '1' to the corresponding bit of register TXBAR. 0 No transmit buffer cancellation 1 Transmit buffer cancellation finished

42.3.5.2.42 Tx Buffer Transmission Interrupt Enable Register (TXBTIE)

Address: 0x00E0 Access: Read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	TIE31	TIE30	TIE29	TIE28	TIE27	TIE26	TIE25	TIE24	TIE23	TIE22	TIE21	TIE20	TIE19	TIE18	TIE17	TIE16
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	TIE15	TIE14	TIE13	TIE12	TIE11	TIE10	TIE9	TIE8	TIE7	TIE6	TIE5	TIE4	TIE3	TIE2	TIE1	TIE0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 612. Tx Buffer Transmission Interrupt Enable Register (TXBTIE)

Table 673. TXBTIE field descriptions

Field	Description
0:31 TIE[31:0]	Transmission Interrupt Enable Each Tx Buffer has its own Transmission Interrupt Enable bit. 0 Transmission interrupt disabled 1 Transmission interrupt enable

42.3.5.2.43 Tx Buffer Cancellation Finished Interrupt Enable Register (TXBCIE)

Address: 0x00E4 Access: R/W

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	CFIE31	CFIE30	CFIE29	CFIE28	CFIE27	CFIE26	CFIE25	CFIE24	CFIE23	CFIE22	CFIE21	CFIE20	CFIE19	CFIE18	CFIE17	CFIE16
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	CFIE15	CFIE14	CFIE13	CFIE12	CFIE11	CFIE10	CFIE9	CFIE8	CFIE7	CFIE6	CFIE5	CFIE4	CFIE3	CFIE2	CFIE1	CFIE0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 613. Tx Buffer Cancellation Finished Interrupt Enable Register (TXBCIE)

Table 674. TXBCIE field description

Field	Description
0:31 CFIE[31:0]	Cancellation Finished Interrupt Enable Each Tx Buffer has its own Cancellation Finished Interrupt Enable bit. 0 Cancellation finished interrupt disabled 1 Cancellation finished interrupt enabled

42.3.5.2.44 Tx Event FIFO Configuration Register (TXEFC)

Address: 0x00F0 Access: RP

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0		EFWM(1)						0		EFS(1)					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	EFSA(1)														0	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. These are protected write (P) bits which means that write access by the bits is possible only when the bit 1 [CCE] and bit 0 [INIT] of CCCR register are set to ‘1’.

Figure 614. Tx Event FIFO Configuration Register (TXEFC)

Table 675. TXEFC field descriptions

Field	Description
2:7 EFWM	Event FIFO Watermark 0x00: Watermark interrupt disabled 0x01: Level for Tx Event FIFO watermark interrupt (IR[TEFW]) ... 0x20: Level for Tx Event FIFO watermark interrupt (IR[TEFW]) 0x21: Watermark interrupt disabled ... 0x3F: Watermark interrupt disabled
10:15 EFS	Event FIFO Size 0x00: Tx Event FIFO disabled 0x01: Number of Tx Event FIFO elements ... 0x20: Number of Tx Event FIFO elements 0x21: Values greater than 32 are interpreted as 32 ... 0x3F: Values greater than 32 are interpreted as 32 The Tx Event FIFO elements are indexed from 0 to EFS - 1
16:29 EFSA	Event FIFO Start Address. Start address of Tx Event FIFO in Message RAM (32-bit word address,*Figure 617*.)

42.3.5.2.45 Tx Event FIFO Status Register (TXEFS)

Address: 0x00F4 Access: R

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	TEFL	EFF	0	0	0	EFPI				
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	EFGI					0	0	EFFL					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 615. Tx Event FIFO Status Register (TXEFS)

Table 676. TXEFS field descriptions

Field	Description
6 TEFL	Tx Event FIFO Element Lost This bit is a copy of interrupt flag IR[TEFL]. When IR[TEFL] is reset, this bit is also reset. 0 No Tx Event FIFO element lost 1 Tx Event FIFO element lost, also set after write attempt to Tx Event FIFO of size zero.
7 EFF	Event FIFO Full 0 Tx Event FIFO not full 1 Tx Event FIFO full
11:15 EFPI	Event FIFO Put Index Tx Event FIFO write index pointer, range 0 to 31.
19:23 EFGI	Event FIFO Get Index Tx Event FIFO read index pointer, range 0 to 31.
26:31 EFFL	Event FIFO Fill Level Number of elements stored in Tx Event FIFO, range 0 to 32.

42.3.5.2.46 Tx Event FIFO Acknowledge Register (TXEFA)

Address: 0x00F8 Access: R/W

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	EFAI				
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 616. Tx Event FIFO Acknowledge Register (TXEFA)

Table 677. TXEFA field descriptions

Field	Description
27:31 EFAI	Event FIFO Acknowledge Index After the Host has read an element or a sequence of elements from the Tx Event FIFO it has to write the index of the last element read from Tx Event FIFO to EFAI. This will set the Tx Event FIFO Get Index TXEFS[EFGI] to EFAI + 1 and update the FIFO 0 Fill Level TXEFS[EFFL].

42.3.6 Message RAM

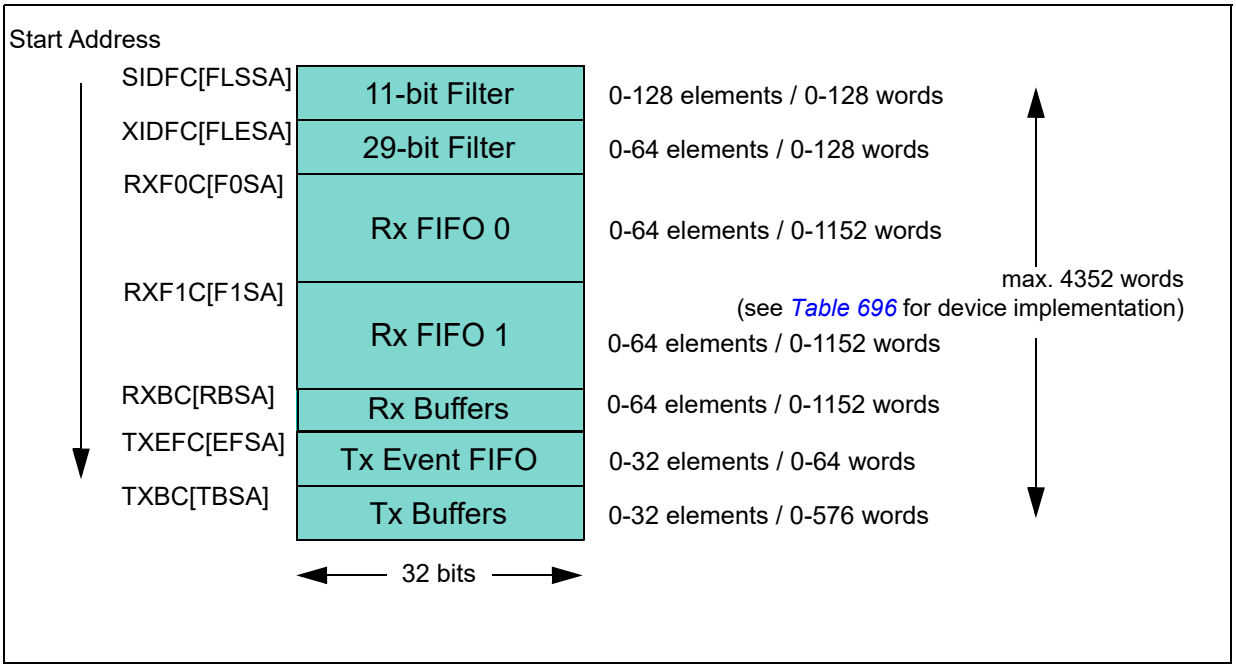
For storage of Rx/Tx messages and for storage of the filter configuration a single- or dual-ported Message RAM has to be connected to the M_CAN module.

Note: Because the Message RAM is equipped with ECC functionality, it is recommended to initialize the Message RAM after hardware reset by writing, for example, 0x00000000 to each Message RAM word to create valid parity/ECC checksums. This avoids that reading from uninitialized Message RAM sections will activate interrupt IR.BEC (Bit Error Corrected) or IR.BEU (Bit Error Uncorrected).

The Message RAM has a width of 32 bits. In case parity checking or ECC is used a respective number of bits has to be added to each word. The M_CAN module can be configured to allocate up to 4352 words in the Message RAM (for the chip-specific implementation details of each module's instance, see device configuration chapter). It is not necessary to configure each of the sections listed in *Figure 617*, nor is there any restriction with respect to the sequence of the sections.

When operated in CAN FD mode the required Message RAM size strongly depends on the element size configured for Rx FIFO0, Rx FIFO1, Rx Buffers, and Tx Buffers via RXESC.F0DS, RXESC.F1DS, RXESC.RBDS, and TXESC.TBDS.

Figure 617. Message RAM Configuration



When the M_CAN addresses the Message RAM it addresses 32-bit words, not single bytes. The configurable start addresses are 32-bit word addresses such as only bits 15 to 2 are evaluated, the two least significant bits are ignored.

Note: The M_CAN does not check for erroneous configuration of the Message RAM. Especially the configuration of the start addresses of the different sections and the number of elements of each section has to be done carefully to avoid falsification or loss of data.

42.3.6.1 Rx Buffer and FIFO element

Up to 64 Rx Buffers and two Rx FIFOs can be configured in the Message RAM. Each Rx FIFO section can be configured to store up to 64 received messages. The structure of a Rx Buffer / FIFO element is shown in the following figure. The element size can be configured for storage of CAN FD messages with up to 64 bytes data field via register RXESC.

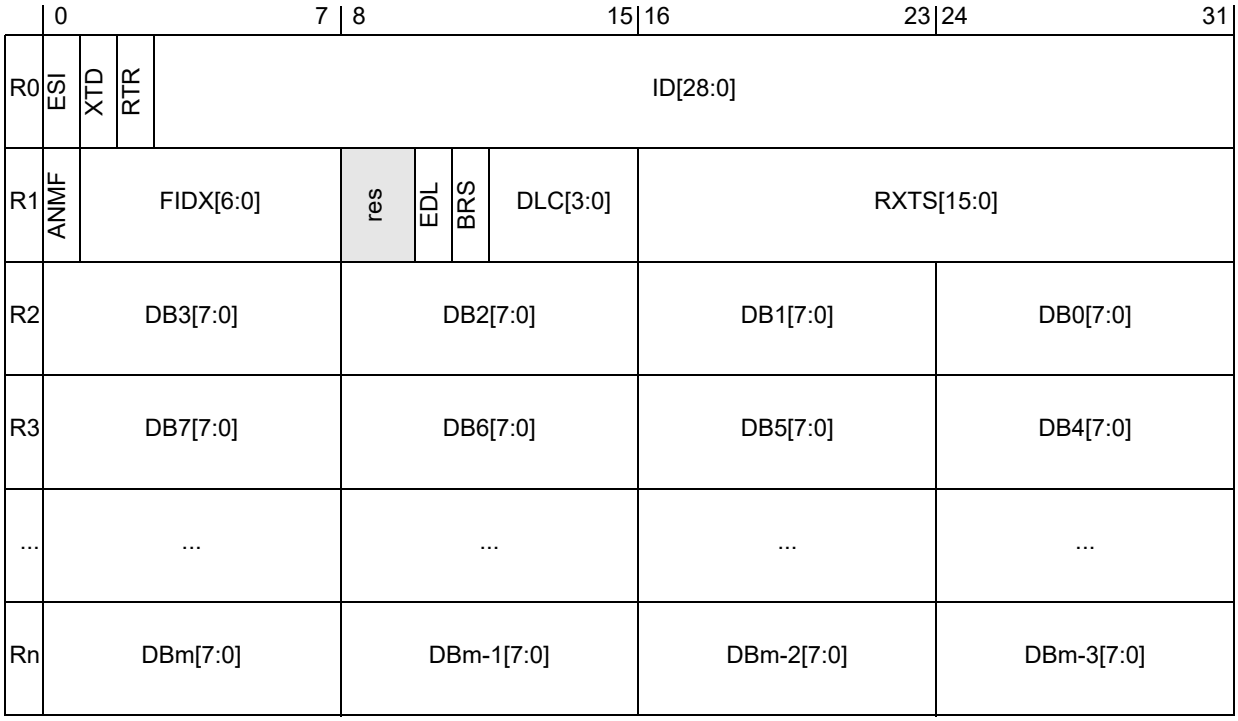


Figure 618. Rx Buffer and FIFO element

Table 678. Rx Buffer and FIFO element descriptions

Field	Description
R0 bit 0 ESI: Error State Indicator	0 Transmitting node is error active 1 Transmitting node is error passive
R0 bit 1 XTD: Extended Identifier	Signals to the Host whether the received frame has a standard or extended identifier. 0 11-bit standard identifier 1 29-bit extended identifier
R0 bit 2 RTR: Remote Transmission Request	Signals to the Host whether the received frame is a data frame or a remote frame. 0 Received frame is a data frame 1 Received frame is a remote frame **Note:** There are no remote frames in CAN FD format. In case a CAN FD frame was received (EDL = '1'), bit RTR reflects the state of the reserved bit r1.
R0 bits 3:31 ID[28:0]: Identifier	Standard or extended identifier depending on bit XTD. A standard identifier is stored into ID[28:18].
R1 bit 0 ANMF: Accepted Non-matching Frame	Acceptance of non-matching frames may be enabled via GFC[ANFS] and GFC[ANFE]. 0 Received frame matching filter index FIDX 1 Received frame did not match any Rx filter element

Table 678. Rx Buffer and FIFO element descriptions (continued)

Field	Description
R1 bits 1:7 FIDX[6:0]: Filter Index	0-127=Index of matching Rx acceptance filter element (invalid if ANMF = '1'). Range is 0 to SIDFC[LSS] - 1 resp. XIDFC[LSE] - 1.
R1 bit 10 EDL: Extended Data Length	0 Standard frame format 1 CAN FD frame format (new DLC-coding and CRC)
R1 bit 11 BRS: Bit Rate Switch	0 Frame received without bit rate switching 1 Frame received with bit rate switching
R1 bits 12:15 DLC[3:0]: Data Length Code	0-8 CAN + CAN FD: Received frame has 0-8 data bytes 9-15 CAN: Received frame has 8 data bytes 9-15 CAN FD: received frame has 12/16/20/24/32/48/64 data bytes
R1 bits 16:31 RXTS[15:0]: Rx Timestamp	Timestamp Counter value captured on start of frame reception. Resolution depending on configuration of the Timestamp Counter Prescaler TSCC[TCP].
R2 bits 0:7	DB3[7:0]: Data Byte 3
R2 bits 8:15	DB2[7:0]: Data Byte 2
R2 bits 16:23	DB1[7:0]: Data Byte 1
R2 bits 24:31	DB0[7:0]: Data Byte 0
R3 bits 0:7	DB7[7:0]: Data Byte 7
R3 bits 8:15	DB6[7:0]: Data Byte 6
R3 bits 16:23	DB5[7:0]: Data Byte 5
R3 bits 24:31	DB4[7:0]: Data Byte 4
...	...
Rn bits 0:7	DBm[7:0]: Data Byte m
Rn bits 8:15	DBm-1[7:0]: Data Byte m-1
Rn bits 16:23	DBm-2[7:0]: Data Byte m-2
Rn bits 24:31	DBm-3[7:0]: Data Byte m-3

Note: Depending on the configuration of the element size (RXESC), between two and sixteen 32-bit words (Rn = 3 to 17) are used for storage of a CAN message's data field.

42.3.6.2 Tx Buffer element

The Tx Buffers section can be configured to hold dedicated Tx Buffers as well as a Tx FIFO / Tx Queue. In case that the Tx Buffers section is shared by dedicated Tx buffers and a Tx FIFO / Tx Queue, the dedicated Tx Buffers start at the beginning of the Tx Buffers section followed by the buffers assigned to the Tx FIFO or Tx Queue. The Tx Handler distinguishes between dedicated Tx Buffers and Tx FIFO / Tx Queue by evaluating the Tx Buffer

configuration TXBC.TFQS and TXBC.NDTB. The element size can be configured for storage of CAN FD messages with up to 64 bytes data field via register TXESC.

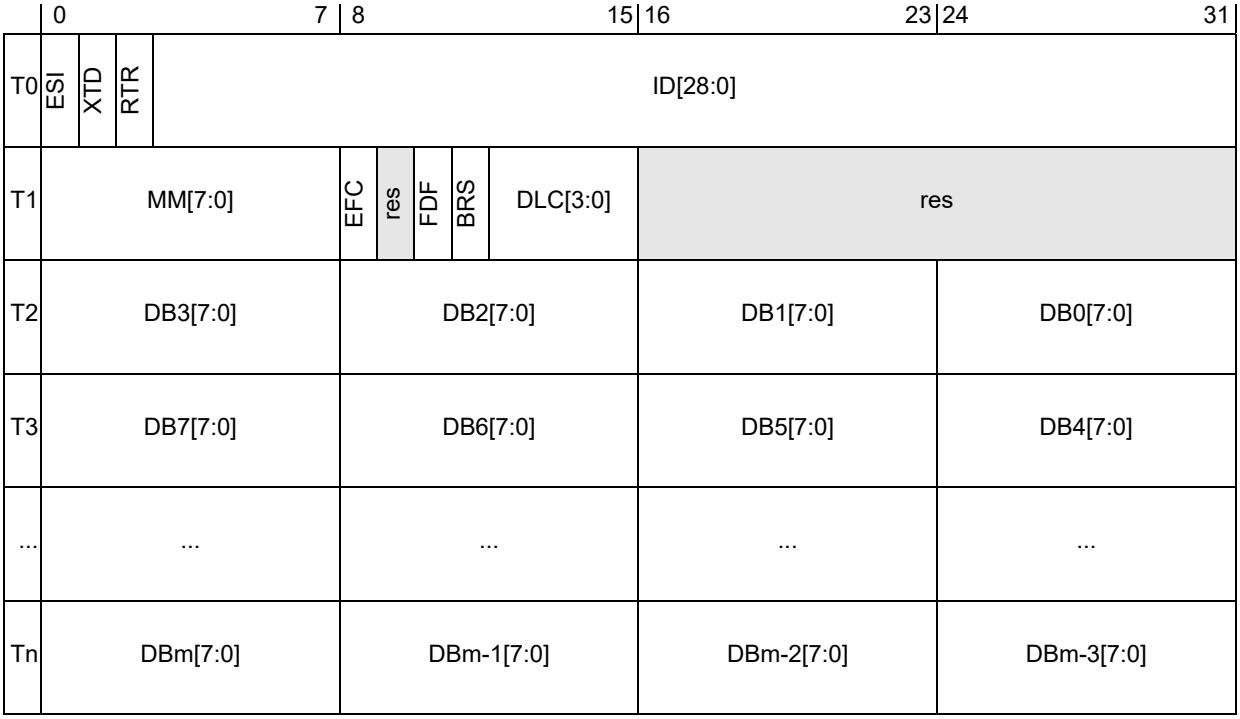


Figure 619. Tx Buffer element

Table 679. Tx Buffer element descriptions

Field	Description
T0 bit 0 ESI: Error State Indicator	0 ESI bit in CAN FD format depends only on error passive flag 1 ESI bit in CAN FD format transmitted recessive **Note:** The ESI bit of the transmit buffer is or'ed with the error passive flag to decide the value of the ESI bit in the transmitted FD frame. As required by the CAN FD protocol specification, an error active node may optionally transmit the ESI bit recessive, but an error passive node will always transmit the ESI bit recessive
T0 bit 1 XTD: Extended Identifier	0 11-bit standard identifier 1 29-bit extended identifier
T0 bit 2 RTR: Remote Transmission Request	0 Transmit data frame 1 Transmit remote frame **Note:** When RTR = 1, the M_CAN transmits a remote frame according to ISO11898-1, even if CCCR.FDOE enables the transmission in CAN FD format.
T0 bits 3:31 ID[28:0]: Identifier	Standard or extended identifier depending on bit XTD. A standard identifier has to be written to ID[28:18].
T1 bits 0:7 MM[7:0]: Message Marker	Written by CPU during Tx Buffer configuration. Copied into Tx Event FIFO element for identification of Tx message status.

Table 679. Tx Buffer element descriptions (continued)

Field	Description
T1 bit 8 EFC **Event FIFO Control**	0 Don't store Tx events 1 Store Tx events
T1 bit 10 FDF **FD Format**	0 Frame transmitted in Classic CAN format 1 Frame transmitted in CAN FD format
T1 bit 11 BRS **Bit Rat Switching**	0 CAN FD frames transmitted without bit rate switching 1 CAN FD frames transmitted with bit rate switching
T1 bits 12:15 DLC[3:0] **Data Length Code**	0-8 Transmit frame with 0-8 data bytes 9-15 Transmit frame with 8 data bytes 9-15 CAN FD: transmit frame has 12/16/20/24/32/48/64 data bytes
T2 bits 0:7	DB3[7:0]: Data Byte 3
T2 bits 8:15	DB2[7:0]: Data Byte 2
T2 bits 16:23	DB1[7:0]: Data Byte 1
T2 bits 24:31	DB0[7:0]: Data Byte 0
T3 bits 0:7	DB7[7:0]: Data Byte 7
T3 bits 8:15	DB6[7:0]: Data Byte 6
T3 bits 16:23	DB5[7:0]: Data Byte 5
T3 bits 24:31	DB4[7:0]: Data Byte 4
...	...
Tn bits 0:7	DBm[7:0]: Data Byte m
Tn bits 8:15	DBm-1[7:0]: Data Byte m-1
Tn bits 16:23	DBm-2[7:0]: Data Byte m-2
Tn bits 24:31	DBm-3[7:0]: Data Byte m-3

Note: Depending on the configuration of the element size (TXESC), between two and sixteen 32-bit words (Tn = 3 to 17) are used for storage of a CAN message's data field.

42.3.6.3 Tx Event FIFO element

Each element stores information about transmitted messages. By reading the Tx Event FIFO the Host CPU gets this information in the order the messages were transmitted. Status information about the Tx Event FIFO can be obtained from register TXEFS.

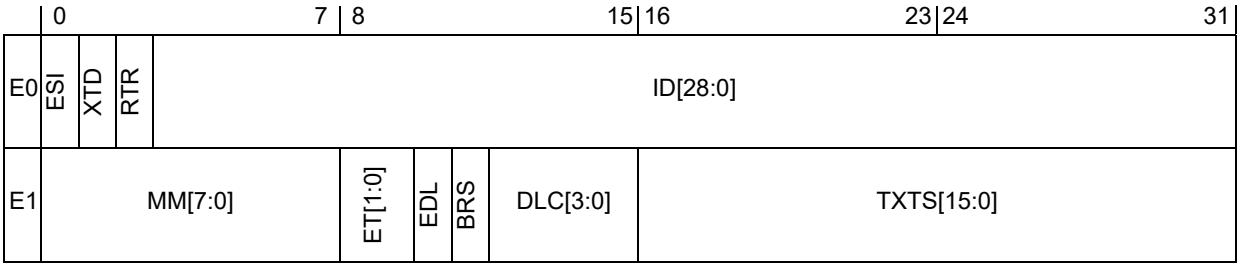


Figure 620. Tx Event FIFO element

Table 680. Tx Event FIFO element descriptions

Field	Description
E0 bit 0 ESI	Error State Indicator 0 Transmitting node is error active 1 Transmitting node is error passive
E0 bit 1 XTD	Extended Identifier 0 11-bit standard identifier 1 29-bit extended identifier
E0 bit 2 RTR	Remote Transmission Request 0 Data frame transmitted 1 Remote frame transmitted
E0 bits 3:31 ID[28:0]	Identifier Standard or extended identifier depending on bit XTD. A standard identifier has to be written to ID[28:18].
E1 bits 0:7 MM[7:0]	Message Marker Copied from Tx Buffer into Tx Event FIFO element for identification of Tx message status.
E1 bit 8:9 ET[1:0]	Event Type 00 Reserved 01 Tx event 10 Transmission in spite of cancellation (always set for transmissions in DAR mode) 11 Reserved
E1 bit 10 EDL	Extended Data Length 0 Standard frame format 1 CAN FD frame format (new DLC-coding and CRC)
E1 bit 11 BRS	Bit Rate Switch 0 Frame transmitted without bit rate switching 1 Frame transmitted with bit rate switching

Table 680. Tx Event FIFO element descriptions (continued)

Field	Description
E1 bits 12:15 DLC[3:0]	Data Length Code 0-8 CAN + CAN FD: frame with 0-8 data bytes transmitted 9-15 CAN: frame with 8 data bytes transmitted 9-15 CAN FD: frame with 12/16/20/24/32/48/64 data bytes transmitted
E1 bits 16:31 TXTS[15:0]	Tx Timestamp Timestamp Counter value captured on start of frame transmission. Resolution depending on configuration of the Timestamp Counter Prescaler TSCC[TCP].

42.3.6.4 Standard Message ID Filter element

Up to 128 filter elements can be configured for 11-bit standard IDs. When accessing a Standard Message ID Filter element, its address is the Filter List Standard Start Address SIDFC[FLSSA] plus the index of the filter element (0 to 127).

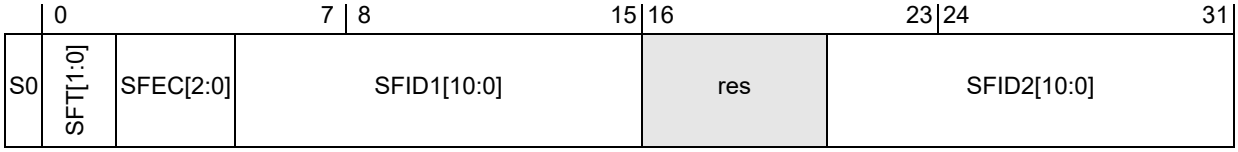


Figure 621. Standard Message ID Filter element

Table 681. Standard Message ID Filter element descriptions

Field	Description
Bits 0:1 SFT[1:0]	Standard Filter Type 00 Range filter from SFID1 to SFID2 (SFID2 ≥ SFID1) 01 Dual ID filter for SFID1 or SFID2 10 Classic filter: SFID1 = filter, SFID2 = mask 11 Reserved
Bit 2:4 SFEC[2:0]	Standard Filter Element Configuration All enabled filter elements are used for acceptance filtering of standard frames. Acceptance filtering stops at the first matching enabled filter element or when the end of the filter list is reached. If SFEC = ‘100’, ‘101’, or ‘110’ a match sets interrupt flag IR[HPM] and, if enabled, an interrupt is generated. In this case register HPMS is updated with the status of the priority match. 000 Disable filter element 001 Store in Rx FIFO 0 if filter matches 010 Store in Rx FIFO 1 if filter matches 011 Reject ID if filter matches 100 Set priority if filter matches 101 Set priority and store in FIFO 0 if filter matches 110 Set priority and store in FIFO 1 if filter matches 111 Store into Rx Buffer or as debug message, configuration of SFT[1:0] ignored

Table 681. Standard Message ID Filter element descriptions (continued)

Field	Description
Bits 5:15 SFID1[10:0]	Standard Filter ID 1 First ID of standard ID filter element. When filtering for Rx Buffers or for debug messages this field defines the ID of a standard message to be stored. The received identifiers must match exactly, no masking mechanism is used.
Bits 21:31 SFID2[10:0]	Standard Filter ID 2 This bit field has a different meaning depending on the configuration of SFEC: 1) SFEC = '001'...'110' Second ID of standard ID filter element 2) SFEC = '111' Filter for Rx Buffers or for debug messages
	SFID2[10:9] decides whether the received message is stored into an Rx Buffer or treated as message A, B, or C of the debug message sequence. 00 Store message into an Rx Buffer 01 Debug Message A 10 Debug Message B 11 Debug Message C
	SFID2[8:6] is used to control the filter event pins at the Extension Interface. A one at the respective bit position enables generation of a pulse at the related filter event pin with the duration of one M_CAN Host clock period in case the filter matches.
	SFID2[5:0] defines the offset to the Rx Buffer Start Address RXBC.RBSA for storage of a matching message.

42.3.6.5 Extended Message ID Filter element

Up to 64 filter elements can be configured for 29-bit extended IDs. When accessing an Extended Message ID Filter element, its address is the Filter List Extended Start Address XIDFC[FLESA] plus two times the index of the filter element (0 to 63).

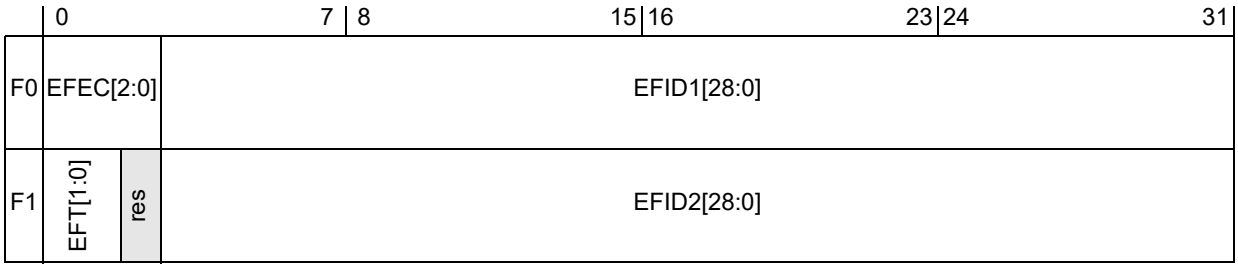


Figure 622. Extended Message ID Filter element

Table 682. Extended Message ID Filter element description

Field	Description
F0 bits 0:2 EFEC[2:0]	Extended Filter Element Configuration All enabled filter elements are used for acceptance filtering of extended frames. Acceptance filtering stops at the first matching enabled filter element or when the end of the filter list is reached. If EFEC = '100', '101', or '110' a match sets interrupt flag IR[HPM] and, if enabled, an interrupt is generated. In this case register HPMS is updated with the status of the priority match. 000 Disable filter element 001 Store in Rx FIFO 0 if filter matches 010 Store in Rx FIFO 1 if filter matches 011 Reject ID if filter matches 100 Set priority if filter matches 101 Set priority and store in FIFO 0 if filter matches 110 Set priority and store in FIFO 1 if filter matches 111 Store into Rx Buffer or as debug message, configuration of EFT[1:0] ignored
F0 bits 3:31 EFID1[28:0]	Extended Filter ID 1 First ID of extended ID filter element. When filtering for Rx Buffers or for debug messages this field defines the ID of an extended message to be stored. The received identifiers must match exactly, only XIDAM masking mechanism (see *Section 42.3.10.1.5*) is used.
F1 bits 0:1 EFT[1:0]	Extended Filter Type 00 Range filter from EFID1 to EFID1 (EFID2 ≥ EFID1) 01 Dual ID filter for EFID1 or EFID2 10 Classic filter: EFID1 = filter, EFID2 = mask 11 Range filter from EFID1 to EFID2 (EFID2 ≥ EFID1), XIDAM mask not applied
F1 bits 3:31 EFID2[28:0]	This bit field has a different meaning depending on the configuration of EFEC: 1) EFEC = '001'...'110' Second ID of extended ID filter element 2) EFEC = '111' Filter for Rx Buffers or for debug messages EFID2[10:9] decides whether the received message is stored into an Rx Buffer or treated as message A, B, or C of the debug message sequence. 00 Store message into an Rx Buffer 01 Debug Message A 10 Debug Message B 11 Debug Message C EFID2[8:6] is used to control the filter event pins at the Extension Interface. A one at the respective bit position enables generation of a pulse at the related filter event pin with the duration of one Host clock period in case the filter matches. EFID2[5:0] defines the offset to the Rx Buffer Start Address RXBC.RBSA for storage of a matching message.

42.3.7 M_CAN functional description

42.3.7.1 Operating modes

42.3.7.1.1 Software initialization

Software initialization is started by setting bit CCCR[INIT], either by software or by a hardware reset, when an uncorrected bit error was detected in the Message RAM, or by going Bus_Off. While CCCR[INIT] is set, message transfer from and to the CAN bus is stopped, the status of the CAN bus transmit output is recessive (HIGH). The counters of the Error Management Logic EML are unchanged. Setting CCCR[INIT] does not change any configuration register. Resetting CCCR[INIT] finishes the software initialization. Afterwards the Bit Stream Processor (BSP) synchronizes itself to the data transfer on the CAN bus by waiting for the occurrence of a sequence of 11 consecutive recessive bits (= Bus_Idle) before it can take part in bus activities and start the message transfer.

Access to the M_CAN configuration registers is only enabled when both bits CCCR[INIT] and CCCR[CCE] are set (protected write).

CCCR[CCE] can only be set/reset while CCCR[INIT] = ‘1’. CCCR[CCE] is automatically reset when CCCR[INIT] is reset.

The following registers are reset when CCCR[CCE] is set

- HPMS–High Priority Message Status
- RXF0S–Rx FIFO 0 Status
- RXF1S–Rx FIFO 1 Status
- TXFQS–Tx FIFO/Queue Status
- TXBRP–Tx Buffer Request Pending
- TXBTO–Tx Buffer Transmission Occurred
- TXBCF–Tx Buffer Cancellation Finished
- TXEFS–Tx Event FIFO Status

The Timeout Counter value TOCV[TOC] is preset to the value configured by TOCC[TOP] when CCCR[CCE] is set.

In addition the state machines of the Tx Handler and Rx Handler are held in idle state while CCCR[CCE] = ‘1’.

The following registers are only writable while CCCR[CCE] = ‘0’

- TXBAR–Tx Buffer Add Request
- TXBCR–Tx Buffer Cancellation Request

CCCR[TEST] and CCCR.MON can only be set by the Host while CCCR[INIT] = ‘1’ and CCCR[CCE] = ‘1’. Both bits may be reset at any time. CCCR.DAR can only be set/reset while CCCR[INIT] = ‘1’ and CCCR[CCE] = ‘1’.

Following sequence can be used as an example code for initializing the CAN module (M_CAN0 in example):

```
-- Setting the INIT bit high to start the initialization
     while (read_data != 0x003){
        write(M_CAN0,0xF,driver.regs.can0_cccr.addr,0x003);
           read_data = read(M_CAN0,0xF,driver.regs.can0_cccr.addr);
     };
```

```
-- Configuring the time stamp and timeout registers
     write(M_CAN0,0xF,driver.regs.can0_tscc.addr,0x000);
     write(M_CAN0,0xF,driver.regs.can0_tocc.addr,0x000);

-- Configuring the interrupt registers
     write(M_CAN0,0xF,driver.regs.can0_ie.addr,0xFFFFFFFF);
     write(M_CAN0,0xF,driver.regs.can0_ils.addr,0x00000002);
     write(M_CAN0,0xF,driver.regs.can0_ile.addr,0x00000003);

     write(M_CAN0,0xF,driver.regs.can0_gfc.addr,0x000);
     write(M_CAN0,0xF,driver.regs.can0_xidam.addr,0x00FFFFFF);

-- Writing the start addresses of all the M_CAN Buffers
     generate and drive M_CAN0 can0_rxf0c keeping {
                                          .f0sa == 0x100;
                                          .f0s  == 0x03;
                                          .f0wm == 0x01;
                                       };
     generate and drive M_CAN0 can0_rxf1c keeping {
                                             .f1sa == 0x200;
                                             .f1s  == 0x03;
                                             .f1wm == 0x01;
                                             };
     generate and drive M_CAN0 can0_sidfc keeping {
                                             .flssa == 0x000;
                                             .lss   == 0x80;
                                             };
     generate and drive M_CAN0 can0_xidfc keeping {
                                             .flesa == 0x080;
                                             .lse   == 0x40;
                                             };
     generate and drive M_CAN0 can0_txbc  keeping {
                                             .tbsa == 0x300;
                                             .ndtb == 0x020;
                                             .tfqs == 0x1D;
                                             };
     generate and drive M_CAN0 can0_txefc keeping {
                                          .efsa == 0x380;
                                          .efs == 0x020;
                                          .efwm == 0x00;
                                          };
-- Writing to the RAM for filter configuration
=================================================
-- Writing to TX Buffer
```

```
      write(RAM,0xF,0xC00,0x04040000);
      write(RAM,0xF,0xC04,0x10830000);
      write(RAM,0xF,0xC08,0x00010001);
      write(RAM,0xF,0xC10,0x50000002);
      write(RAM,0xF,0xC14,0x11830000);
      write(RAM,0xF,0xC18,0x00070102);

-- Writing to s-filter
      write(RAM,0xF,0x000,0x58020004);
      write(RAM,0xF,0x004,0x08000005);
      write(RAM,0xF,0x008,0x58060007);
      write(RAM,0xF,0x00C,0x08030008);
      write(RAM,0xF,0x010,0x880E07FB);
      write(RAM,0xF,0x014,0x980D07FE);
      write(RAM,0xF,0x018,0x68100100);
      write(RAM,0xF,0x01C,0x70120200);
      write(RAM,0xF,0x020,0x981F07F6);
      write(RAM,0xF,0x024,0x50140014);
      write(RAM,0xF,0x028,0x58160017);
      write(RAM,0xF,0x02C,0x08110019);
      write(RAM,0xF,0x030,0x4A22001A);
      write(RAM,0xF,0x034,0xA01A07FF);
      write(RAM,0xF,0x038,0x5BFF07E3);
      write(RAM,0xF,0x03C,0x080007FF);

-- Writing to x-filter
      write(RAM,0xF,0x200,0x201ABCD9);
      write(RAM,0xF,0x204,0x001ABCE5);
      write(RAM,0xF,0x208,0x403BCDEF);
      write(RAM,0xF,0x20C,0x003BCDEF);
      write(RAM,0xF,0x210,0x605CDEF2);
      write(RAM,0xF,0x214,0x405CDEF0);
      write(RAM,0xF,0x218,0x407DEEFF);
      write(RAM,0xF,0x21C,0x007DEF01);
      write(RAM,0xF,0x220,0x209ABCDE);
      write(RAM,0xF,0x224,0x009ABCDF);
      write(RAM,0xF,0x228,0xA0BBCDEF);
      write(RAM,0xF,0x22C,0x40BBCDEE);
      write(RAM,0xF,0x230,0x20DCDEF0);
      write(RAM,0xF,0x234,0x9FFFFFFF);
      write(RAM,0xF,0x238,0x40000000);
      write(RAM,0xF,0x23C,0x1FFFFFFF);
================================================
write(M_CAN0,0xF,driver.regs.can0_nbtp.addr,0x00002303);
```

```
print "Configuration done";
read_data = 0x01;
while (read_data != 0x00){
   write(M_CAN0,0xF,driver.regs.can0_cccr.addr, 0x000);
      wait [1];
      read_data = read(M_CAN0,0xF,driver.regs.can0_cccr.addr);
   };

print "Initialization done";
```

42.3.7.2 Normal operation

Once the M_CAN is initialized and CCCR[INIT] is reset to zero, the M_CAN synchronizes itself to the CAN bus and is ready for communication.

After passing the acceptance filtering, received messages including Message ID and DLC are stored into Rx FIFO 0 or Rx FIFO 1.

For messages to be transmitted dedicated Tx Buffers or a Tx FIFO, or both, or a Tx Queue can be initialized or updated. Automated transmission on reception of remote frames is not implemented.

42.3.7.3 CAN FD Operation

There are two variants in the CAN FD frame transmission, first the CAN FD frame without bit rate switching. The second variant is the CAN FD frame where control field, data field, and CRC field are transmitted with a higher bit rate than the beginning and the end of the frame.

The previously reserved bit in CAN frames with 11-bit identifiers and the first previously reserved bit in CAN frames with 29-bit identifiers will now be decoded as FDF bit. FDF = *recessive* signifies a CAN FD frame, FDF = *dominant* signifies a Classic CAN frame. In a CAN FD frame, the two bits following FDF, res and BRS, decide whether the bit rate inside of this CAN FD frame is switched. A CAN FD bit rate switch is signified by res = *dominant* and BRS = *recessive*. The coding of res = *recessive* is reserved for future expansion of the protocol. In case the M_CAN receives a frame with FDF = *recessive* and res = *recessive*, it will signal a Protocol Exception Event by setting bit PSR.PXE. When Protocol Exception Handling is enabled (CCCR.PXHD = ‘0’), this causes the operation state to change from Receiver (PSR.ACT = “10”) to Integrating (PSR.ACT = “00”) at the next sample point. In case Protocol Exception Handling is disabled (CCCR.PXHD = ‘1’), the M_CAN will treat a *recessive* res bit as an form error and will respond with an error frame.

CAN FD operation is enabled by programming CCCR.FDOE. In case CCCR.FDOE = ‘1’, transmission and reception of CAN FD frames is enabled. Transmission and reception of Classic CAN frames is always possible. Whether a CAN FD frame or a Classic CAN frame is transmitted can be configured via bit FDF in the respective Tx Buffer element. With CCCR.FDOE = ‘0’, received frames are interpreted as Classic CAN frames, which leads to the transmission of an error frame when receiving a CAN FD frame. When CAN FD operation is disabled, no CAN FD frames are transmitted even if bit FDF of a Tx Buffer element is set. CCCR.FDOE and CCCR.BRSE can only be changed while CCCR.INIT and CCCR.CCE are both set. CCCR.FDOE and CCCR.BRSE can only be changed while CCCR.INIT and CCCR.CCE are both set.

With CCCR.FDOE = ‘0’, the setting of bits FDF and BRS is ignored and frames are transmitted in Classic CAN format. With CCCR.FDOE = ‘1’ and CCCR.BRSE = ‘0’, only bit FDF of a Tx Buffer element is evaluated. With CCCR.FDOE = ‘1’ and CCCR.BRSE = ‘1’, transmission of CAN FD frames with bit rate switching is enabled. All Tx Buffer elements with bits FDF and BRS set are transmitted in CAN FD format with bit rate switching.

A mode change during CAN operation is only recommended under the following conditions:

- The failure rate in the CAN FD data phase is significant higher than in the CAN FD arbitrationphase. In this case disable the CAN FD bit rate switching option for transmissions
- During system startup all nodes are transmitting Classic CAN messages until it is verified thatthey are able to communicate in CAN FD format. If this is true, all nodes switch to CAN FD operation
- Wake-up messages in CAN Partial Networking have to be transmitted in Classic CAN format
- End-of-line programming in case not all nodes are CAN FD capable. Non CAN FD nodes are held in Silent mode until programming has completed. Then all nodes switch back to Classic CAN communication

In the CAN FD format, the coding of the DLC differs from the standard CAN format. The DLC codes 0 to 8 have the same coding as in standard CAN, the codes 9 to 15, which in standard CAN all code a data field of 8 bytes, are coded according to *Table 683*.

Table 683. Coding of DLC in CAN FD

DLC	9	10	11	12	13	14	15
Number of data bytes	12	16	20	24	32	48	64

In CAN FD frames, the bit timing will be switched inside the frame, after the BRS (Bit Rate Switch) bit, if this bit is *recessive*. Before the BRS bit, in the CAN FD arbitration phase, the nominal CAN bit timing is used as defined by the Nominal Bit Timing & Prescaler Register NBTP. In the following CAN FD data phase, the data phase bit timing is used as defined by the Data Bit Timing & Prescaler Register DBTP. The bit timing is switched back from the data phase timing at the CRC delimiter or when an error is detected, whichever occurs first.

The maximum configurable bit rate in the CAN FD data phase depends on the CAN clock frequency (CAN clock). Example: with a CAN clock frequency of 20MHz and the shortest configurable bit time of 4 tq, the bit rate in the data phase is 5 Mbit/s.

In both data frame formats, CAN FD and CAN FD with bit rate switching, the value of the bit ESI (Error Status Indicator) is determined by the transmitter’s error state at the start of the transmission. If the transmitter is error passive, ESI is transmitted *recessive*, else it is transmitted *dominant*.

42.3.7.4 Transceiver Delay Compensation

During the data phase of a CAN FD transmission only one node is transmitting, all others are receivers. The length of the bus line has no impact. When transmitting via CAN Tx output pin the M_CAN the transmitted data from its local CAN transceiver via CAN Rx input pin. The received data is delayed by the transmitter delay. In case this delay is greater than NTSEG1 (time segment before sample point), a bit error is detected. In order to enable a data phase bit time that is even shorter than the transmitter delay, the delay compensation is

introduced. Without transmitter delay compensation, the bit rate in the data phase of a CAN FD frame is limited by the transmitter delay.

42.3.7.4.1 Description

The CAN FD protocol unit has implemented a delay compensation mechanism to compensate the CAN transceiver's loop delay, thereby enabling transmission with higher bit rates during the CAN FD data phase independent of the delay of a specific CAN transceiver.

To check for bit errors during the data phase of transmitting nodes, the delayed transmit data is compared against the received data at the Secondary Sample Point SSP. If a bit error is detected, the transmitter will react on this bit error at the next following regular sample point. During arbitration The transmitter delay compensation enables configurations where the data bit time is shorter than the transmitter delay, it is described in detail in the new ISO11898-1. It is enabled by setting bit DBTP.TDC.phase the delay compensation is always disabled.

The received bit is compared against the transmitted bit at the SSP. The SSP position is defined as the sum of the measured delay from the M_CAN's Tx output through the transceiver to the CAN Rx input plus the transmitter delay compensation offset as configured by TDCR.TDCO. The transmitter delay compensation offset is used to adjust the position of the SSP inside the received bit (for example half of the bit time in the data phase). The position of the secondary sample point is rounded down to the next integer number of mtq.

PSR.TDCV shows the actual transmitter delay compensation value. PSR.TDCV is cleared when CCCR.INIT is set and is updated at each transmission of an FD frame while DBTP.TDC is set.

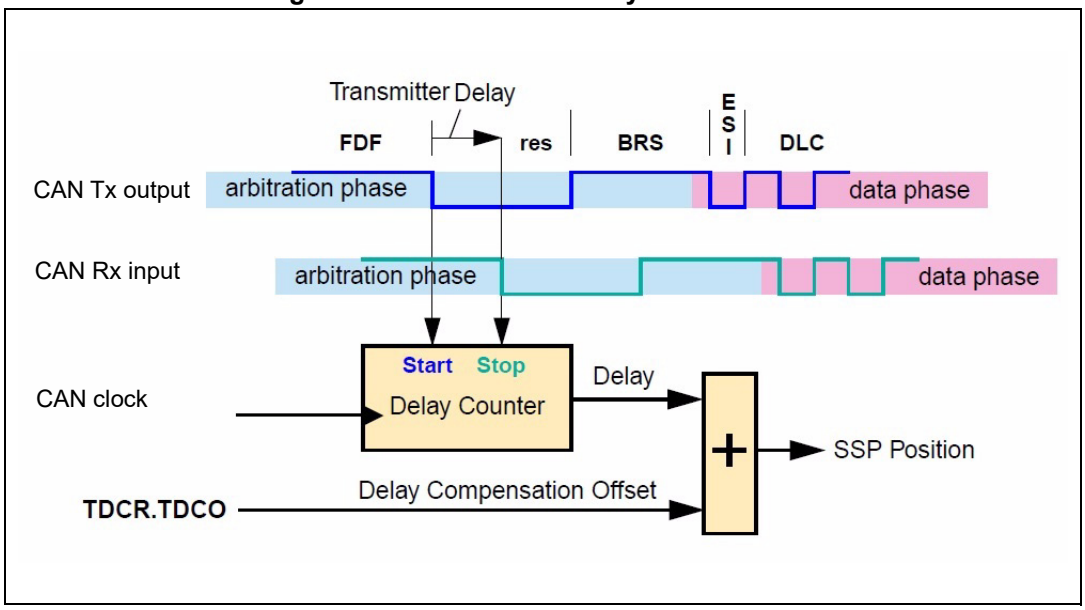
The following boundary conditions have to be considered for the transmitter delay compensation implemented in the M_CAN:

- The sum of the measured delay from CAN Tx output to CAN Rx input and the configured transmitter delay compensation offset TDCR.TDCO has to be less than 6 bit times in the data phase
- The sum of the measured delay from CAN Tx output to CAN Rx input and the configured transmitter delay compensation offset TDCR.TDCO has to be less or equal 127 mtq. In case this sum exceeds 127 mtq, the maximum value of 127 mtq is used for transmitter delay compensation
- The data phase ends at the sample point of the CRC delimiter, that stops checking of receive bits at the SSPs

Figure 623 below describes how the Transceiver Delay Compensation measurement.

If transmitter delay compensation is enabled by programming DBTP.TDC = '1', the measurement is started within each transmitted CAN FD frame at the falling edge of bit FDF to bit res. The measurement is stopped when this edge is seen at the receive input CAN Rx input of the transmitter. The resolution of this measurement is one mtq.

Figure 623. Transceiver Delay measurement



To avoid that a dominant glitch inside the received FDF bit ends the delay compensation measurement before the falling edge of the received res bit, resulting in a to early SSP position, the use of a transmitter delay compensation filter window can be enabled by programming TDCR.TDCF. This defines a minimum value for the SSP position. Dominant edges on CAN Rx input, that would result in an earlier SSP position are ignored for transmitter delay measurement. The measurement is stopped when the SSP position is at least TDCR.TDCF AND CAN Rx input is low.

42.3.7.5 Restricted Operation Mode

In Restricted Operation Mode the node is able to receive data and remote frames and to give acknowledge to valid frames, but it does not send data frames, remote frames, active error frames, or overload frames. In case of an error condition or overload condition, it does not send dominant bits, instead it waits for the occurrence of bus idle condition to resynchronize itself to the CAN communication. The error counters are not incremented. The Host can set the M_CAN into Restricted Operation mode by setting bit CCCR.ASM. The bit can only be set by the Host when both CCCR.CCE and CCCR.INIT are set to ‘1’. The bit can be reset by the Host at any time.

Restricted Operation Mode is automatically entered when the Tx Handler was not able to read data from the Message RAM in time. To leave Restricted Operation Mode, the Host CPU has to reset CCCR.ASM.

The Restricted Operation Mode can be used in applications that adapt themselves to different CAN bit rates. In this case the application tests different bit rates and leaves the Restricted Operation Mode after it has received a valid frame.

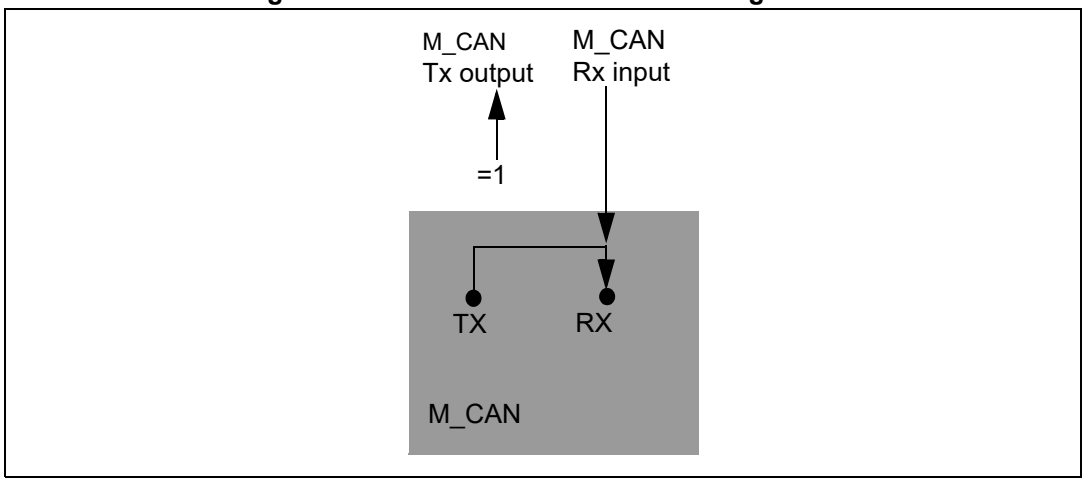
If the M_CAN is connected to a Clock Calibration on CAN Unit, CCCR.ASM is controlled by “calibration OK” input. In case “calibration OK” switches to ‘0’, bit CCCR.ASM is set. When “calibration OK” switches back to ‘1’, bit CCCR.ASM returns to the previously written value. The state of CCCR.ASM is the written value while “calibration OK” input is at ‘1’. The input is hardwired to ‘1’ when there is no Clock Calibration on CAN Unit connected.

42.3.7.6 Bus monitoring mode

The M_CAN is set in Bus Monitoring Mode by programming CCCR[MON] to one. In Bus Monitoring Mode (see ISO11898-1, 10.12 Bus monitoring), the M_CAN is able to receive valid data frames and valid remote frames, but cannot start a transmission. In this mode, it sends only recessive bits on the CAN bus. If the M_CAN is required to send a dominant bit (ACK bit, overload flag, active error flag), the bit is rerouted internally so that the M_CAN monitors this dominant bit, although the CAN bus may remain in recessive state. In Bus Monitoring Mode, register TXBRP is held in reset state. The Bus Monitoring Mode can be used to analyze the traffic on a CAN bus without affecting it by the transmission of dominant bits.

Figure 624 shows the connection of M_CAN Tx and Rx signals to the M_CAN in Bus Monitoring Mode.

Figure 624. Pin control in Bus Monitoring mode



42.3.7.7 Disabled automatic retransmission

According to the CAN Specification (see ISO11898-1, 6.3.3 Recovery Management), the M_CAN provides means for automatic retransmission of frames that have lost arbitration or that have been disturbed by errors during transmission. By default automatic retransmission is enabled. To support time-triggered communication as described in ISO 11898-1, chapter 9.2, the automatic retransmission may be disabled via CCCR[DAR].

42.3.7.7.1 Frame transmission in DAR mode

In DAR mode, all transmissions are automatically cancelled after they started on the CAN bus. A Tx Buffer's Tx Request Pending bit TXBRP[TRPx] is reset after successful

transmission, when a transmission has not yet been started at the point of cancellation, has been aborted due to lost arbitration, or when an error occurred during frame transmission.

- Successful transmission:
  - Corresponding Tx Buffer Transmission Occurred bit TXBTO[TOx] set
  - Corresponding Tx Buffer Cancellation Finished bit TXBCF[CFx] not set
- Successful transmission in spite of cancellation:
  - Corresponding Tx Buffer Transmission Occurred bit TXBTO[TOx] set
  - Corresponding Tx Buffer Cancellation Finished bit TXBCF[CFx] set
- Arbitration lost or frame transmission disturbed:
  - Corresponding Tx Buffer Transmission Occurred bit TXBTO[TOx] not set
  - Corresponding Tx Buffer Cancellation Finished bit TXBCF[CFx] set

In case of a successful frame transmission, and if storage of Tx events is enabled, a Tx Event FIFO element is written with Event Type ET = ‘10’ (transmission in spite of cancellation).

42.3.7.8 Power down (Sleep mode)

The M_CAN can be set into power down mode controlled by input signal clock stop request or via CC Control Register CCCR[CSR]. As long as the clock stop request signal is active, bit CCCR[CSR] is read as one. When all pending transmission requests have completed, the M_CAN waits until bus idle state is detected. Then the M_CAN sets then CCCR[INIT] to one to prevent any further CAN transfers. Now the M_CAN acknowledges that it is ready for power down by setting clock stop acknowledge output signal to one and CCCR[CSA] to one. In this state, before the clocks are switched off, further register accesses can be made. A write access to CCCR[INIT] will have no effect. Now the module clock inputs (CAN clock and Host clock) may be switched off. To leave power down mode, the application has to turn on the module clocks before resetting signal clock stop request signal resp. CC Control Register flag CCCR[CSR]. The M_CAN will acknowledge this by resetting clock stop acknowledge output signal and resetting CCCR[CSA]. Afterwards, the application can restart CAN communication by resetting bit CCCR[INIT].

42.3.7.9 Test modes

To enable write access to register TEST (see *Section 42.3.5.2.4: Test Register (TEST)*), bit CCCR[TEST] has to be set to one. This allows the configuration of the test modes and test functions.

Four output functions are available for the CAN Tx output pin by programming TEST.TX. Additionally to its default function – the serial data output – it can drive the CAN Sample Point signal to monitor the M_CAN’s bit timing and it can drive constant dominant or recessive values. The actual value at M_CAN Rx pin can be read from TEST[RX]. Both functions can be used to check the CAN bus physical layer.

Due to the synchronization mechanism between CAN clock and Host clock domain, there may be a delay of several Host clock periods between writing to TEST[TX] until the new configuration is visible at output Tx pin. This applies also when reading input Rx pin via TEST[RX].

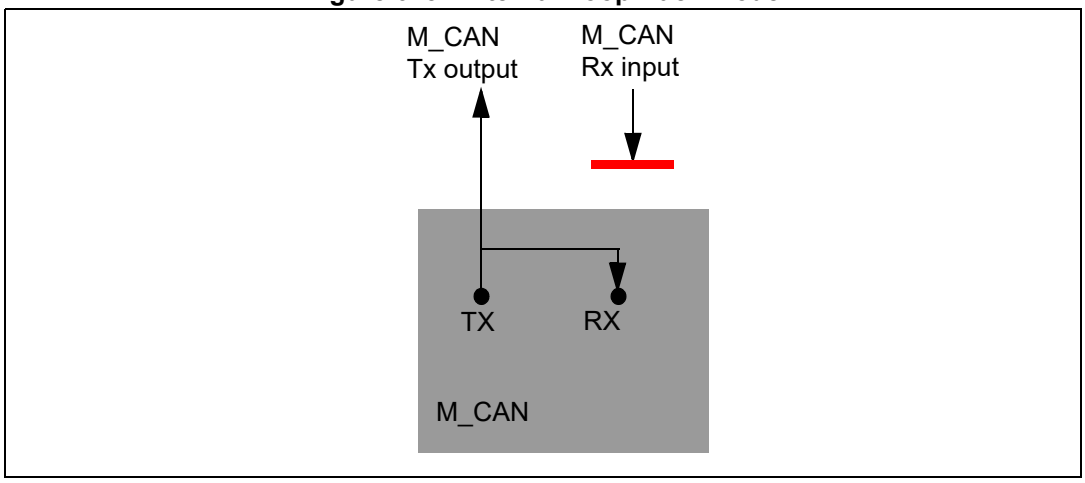
Note: Test modes should be used for production tests or self test only. The software control for Tx pin interferes with all CAN protocol functions. It is not recommended to use test modes for application.

42.3.7.9.1 External Loop Back mode

The M_CAN can be set in External Loop Back mode by programming TEST.LBCK to one. In Loop Back mode, the M_CAN treats its own transmitted messages as received messages and stores them (if they pass acceptance filtering) into Rx FIFOs. *Figure 625* shows the connection of Tx and Rx signals to the M_CAN in External Loop Back mode.

This mode is provided for hardware self-test. To be independent from external stimulation, the M_CAN ignores acknowledge errors (recessive bit sampled in the acknowledge slot of a data/remote frame) in Loop Back mode. In this mode the M_CAN performs an internal feedback from its Tx output to its Rx input. The actual value of the CAN Rx input pin is disregarded by the M_CAN. The transmitted messages can be monitored at the CAN Tx output pin.

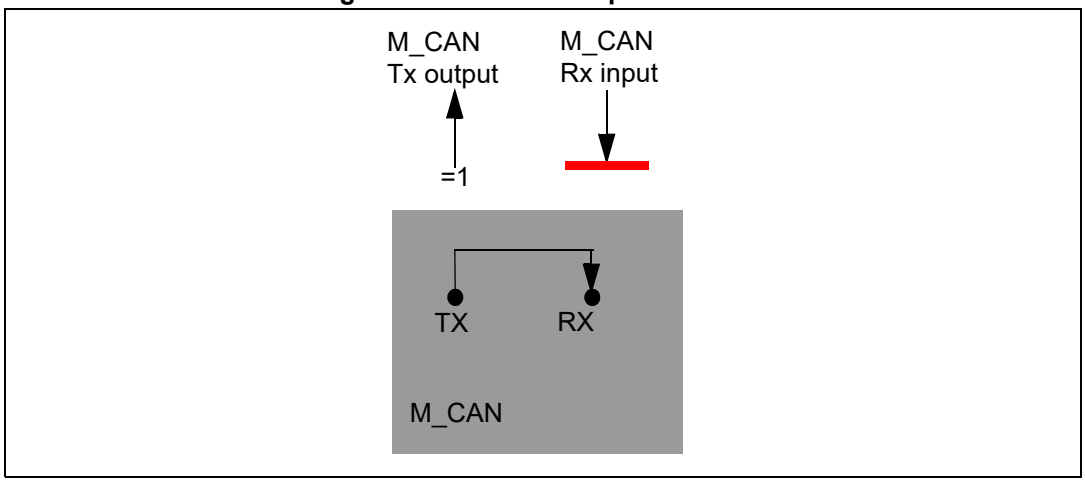
Figure 625. External Loop Back mode



42.3.7.9.2 Internal Loop Back mode

Internal Loop Back mode is entered by programming bits TEST[LBCK] and CCCR[MON] to one. This mode can be used for a “Hot Selftest”, meaning the M_CAN can be tested without affecting a running CAN system connected to the CAN Tx output and Rx input pins. In this mode M_CAN Tx pin is disconnected from the M_CAN and M_CAN Tx pin is held recessive. *Figure 626* shows the connection of M_CAN Tx pin and Rx to the M_CAN in case of Internal Loop Back mode.

Figure 626. Internal loop back mode



42.3.8 Timestamp generation

For timestamp generation the M_CAN supplies a 16-bit wrap-around counter. A prescaler TSCC[TCP] can be configured to clock the counter in multiples of CAN bit times (1…16). The counter is readable via TSCV[TSC]. A write access to register TSCV resets the counter to zero. When the timestamp counter wraps around interrupt flag IR[TSW] is set.

On start of frame reception / transmission the counter value is captured and stored into the timestamp section of an Rx Buffer /Rx FIFO (RXTS[15:0]) or Tx Event FIFO (TXTS[15:0]) element.

By programming bit TSCC.TSS, a 16-bit timestamp can be used.

42.3.9 Timeout counter

To signal timeout conditions for Rx FIFO 0, Rx FIFO 1, and the Tx Event FIFO the M_CAN supplies a 16-bit Timeout Counter. It operates as down-counter and uses the same prescaler controlled by TSCC.TCP as the Timestamp Counter. The Timeout Counter is configured via register TOCC. The actual counter value can be read from TOCV[TOC]. The Timeout Counter can only be started while CCCR[INIT] = ‘0’. It is stopped when CCCR[INIT] = ‘1’, for exampewhen the M_CAN enters Bus_Off state.

The operation mode is selected by TOCC[TOS]. When operating in Continuous mode, the counter starts when CCCR[INIT] is reset. A write to TOCV presets the counter to the value configured by TOCC.TOP and continues down-counting. When the Timeout Counter is controlled by one of the FIFOs, an empty FIFO presets the counter to the value configured by TOCC[TOP]. Down-counting is started when the first FIFO element is stored. Writing to TOCV has no effect.

When the counter reaches zero, interrupt flag IR[TOO] is set. In Continuous mode, the counter is immediately restarted at TOCC[TOP].

Note: The clock signal for the Timeout Counter is derived from the CAN core’s sample point signal. Therefore the point in time where the Timeout Counter is decremented may vary due to the synchronization / re-synchronization mechanism of the CAN core. If the baud rate switch feature in CAN FD is used, the timeout counter is clocked differently in arbitration and data field.

42.3.10 Rx handling

The Rx Handler controls the acceptance filtering, the transfer of received messages to the Rx Buffers or to one of the two Rx FIFOs, as well as the Rx FIFO's Put and Get Indices.

42.3.10.1 Acceptance filtering

The M_CAN offers the possibility to configure two sets of acceptance filters, one for standard identifiers and one for extended identifiers. These filters can be assigned to an Rx Buffer or to Rx FIFO 0,1. For acceptance filtering each list of filters is executed from element #0 until the first matching element. Acceptance filtering stops at the first matching element. The following filter elements are not evaluated for this message.

The main features are:

- Each filter element can be configured as
  - Range filter (from - to)
  - Filter for one or two dedicated IDs
  - Classic bit mask filter
- Each filter element is configurable for acceptance or rejection filtering
- Each filter element can be enabled / disabled individually
- Filters are checked sequentially, execution stops with the first matching filter element

Related configuration registers are:

- Global Filter Configuration (GFC)
- Standard ID Filter Configuration (SIDFC)
- Extended ID Filter Configuration (XIDFC)
- Extended ID AND Mask (XIDAM)

Depending on the configuration of the filter element (SFEC/EFEC) a match triggers one of the following actions:

- Store received frame in FIFO 0 or FIFO 1
- Store received frame in Rx Buffer
- Store received frame in Rx Buffer and generate pulse at filter event pin
- Reject received frame
- Set High Priority Message interrupt flag IR[HPM]
- Set High Priority Message interrupt flag IR[HPM] and store received frame in FIFO 0 or FIFO 1

Acceptance filtering is started after the complete identifier has been received. After acceptance filtering has completed, and if a matching Rx Buffer or Rx FIFO has been found, the Message Handler starts writing the received message data in portions of 32 bit to the matching Rx Buffer or Rx FIFO. If the CAN protocol controller has detected an error condition (such as,CRC error), this message is discarded with the following impact on the affected Rx Buffer or Rx FIFO:

Rx Buffer

New Data flag of matching Rx Buffer is not set, but Rx Buffer (partly) overwritten with received data. For error type see PSR.LEC respectively PSR.DLEC.

Rx FIFO

Put index of matching Rx FIFO is not updated, but related Rx FIFO element (partly) overwritten with received data. For error type see PSR.LEC respectively PSR.DLEC. In case the matching Rx FIFO is operated in overwrite mode, the boundary conditions described in *Section 42.3.10.2.2: Rx FIFO Overwrite Mode* have to be considered.

Note: When an accepted message is written to one of the two Rx FIFOs, or into a Rx Buffer, the unmodified received identifier is stored independent of the filter(s) used. The result of the acceptance filter process is strongly depending on the sequence of configured filter elements.

42.3.10.1.1 Range filter

The filter matches for all received frames with Message IDs in the range defined by SFID1/SFID2 resp. EFID1/EFID2. There are two possibilities when range filtering is used together with extended frames:

- EFT = '00': The Message ID of received frames is ANDed with the Extended ID AND Mask (XIDAM) before the range filter is applied
- EFT = '11': The Extended ID AND Mask (XIDAM) is not used for range filtering

42.3.10.1.2 Filter for specific IDs

A filter element can be configured to filter for one or two specific Message IDs. To filter for one specific Message ID, the filter element has to be configured with SFID1 = SFID2 resp. EFID1 =EFID2.

42.3.10.1.3 Classic Bit Mask Filter

Classic bit mask filtering is intended to filter groups of Message IDs by masking single bits of a received Message ID. With classic bit mask filtering SFID1/EFID2 is used as Message ID filter, while SFID2/EFID2 is used as filter mask.

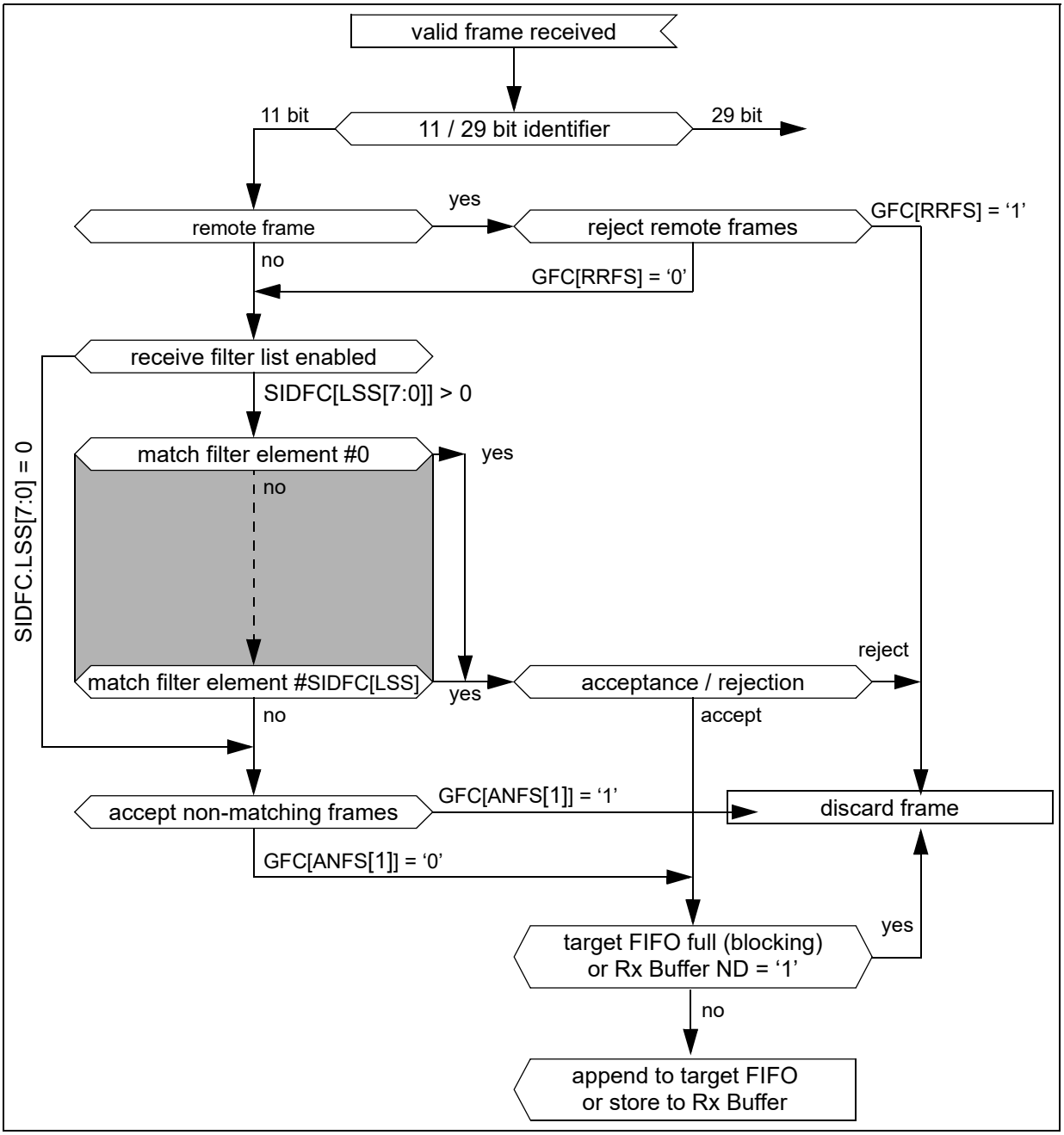
A zero bit at the filter mask will mask out the corresponding bit position of the configured ID filter, for example the value of the received Message ID at that bit position is not relevant for acceptance filtering. Only those bits of the received Message ID where the corresponding mask bits are one are relevant for acceptance filtering. In case all mask bits are one, a match occurs only when the received Message ID and the Message ID filter are identical. If all mask bits are zero, all Message IDs match.

42.3.10.1.4 Standard message ID filtering

Figure 621 shows the flow for standard message ID (11-bit Identifier) filtering.

Controlled by the Global Filter Configuration GFC and the Standard ID Filter Configuration SIDFC Message ID, Remote Transmission Request bit (RTR), and the Identifier Extension bit (IDE) of received frames are compared against the list of configured filter elements.

Figure 627. Standard Message ID Filter Path



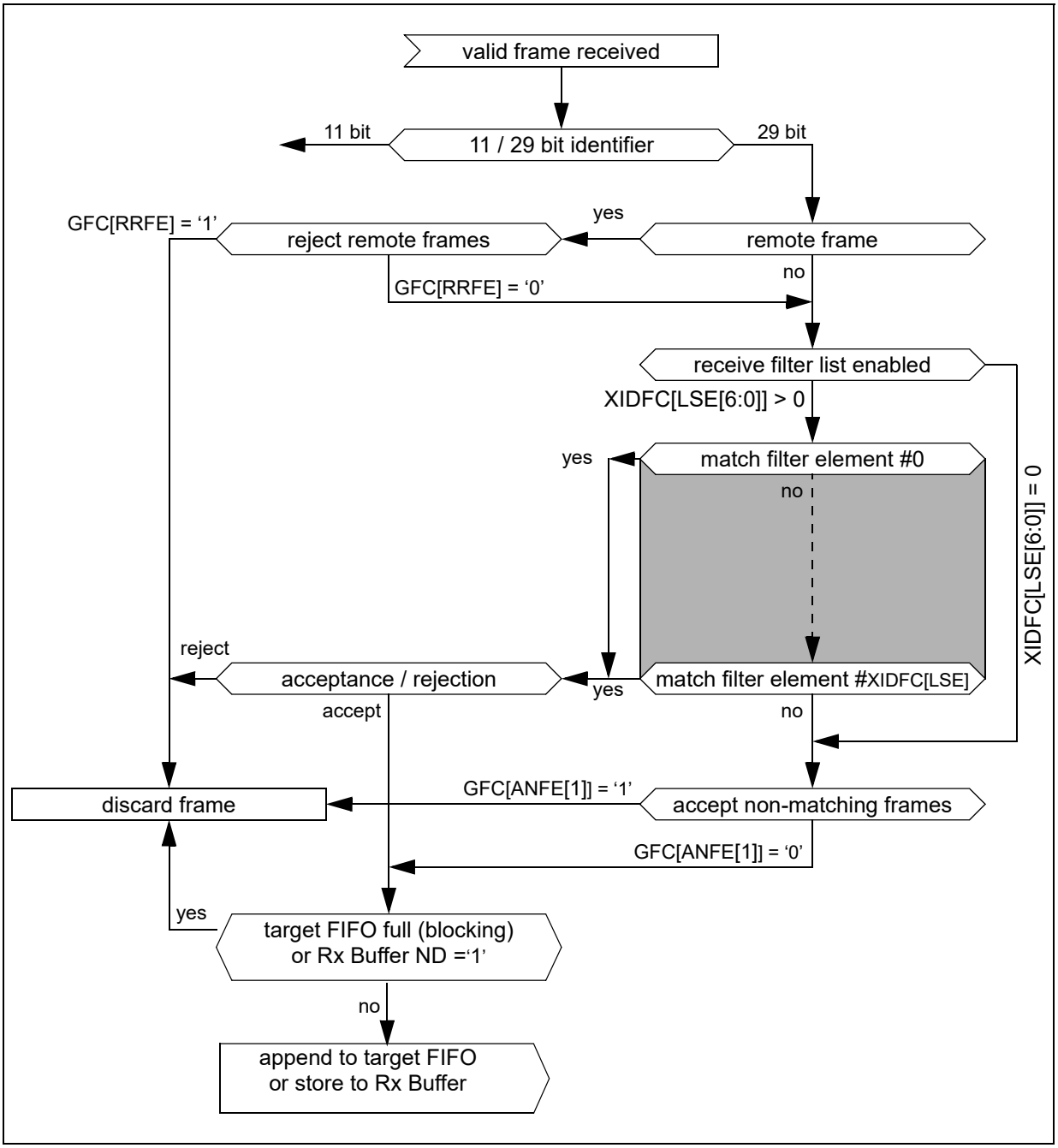
42.3.10.1.5 Extended message ID filtering

Figure 628 below shows the flow for extended Message ID (29-bit Identifier) filtering. The Extended Message ID Filter element is described in *Section 42.3.5.2.21: Extended ID Filter Configuration Register (XIDFC)*.

Remote Transmission Request bit (RTR), and the Identifier Extension bit (IDE) of received frames are compared against the list of configured filter elements which is controlled by the

Global Filter Configuration (GFC) and the Extended ID Filter Configuration (XIDFC) Message ID. The Extended ID AND Mask (XIDAM) is ANDed with the received identifier before the filter list is executed.

Figure 628. Extended Message ID Filter Path



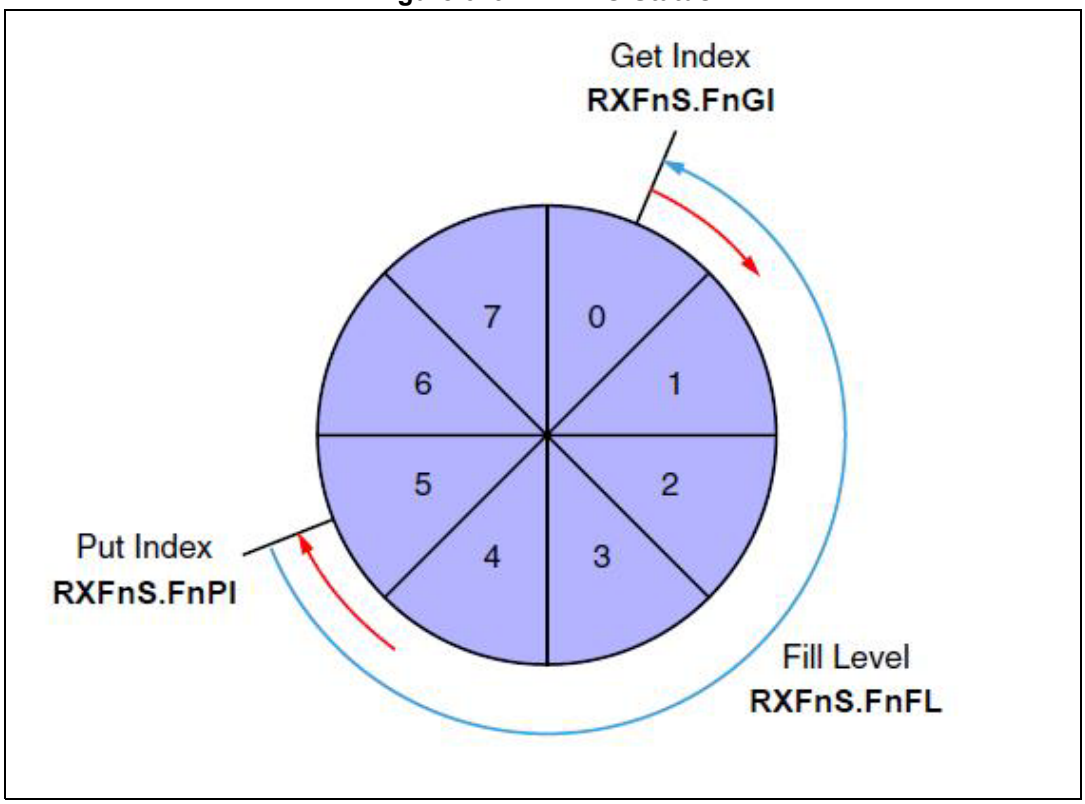
42.3.10.2 Rx FIFOs

Rx FIFO 0 and Rx FIFO 1 can be configured to hold up to 64 elements each. Configuration of the two Rx FIFOs is done via registers RXF0C and RXF1C.

Received messages that passed acceptance filtering are transferred to the Rx FIFO as configured by the matching filter element. For a description of the filter mechanisms available for Rx FIFO 0 and Rx FIFO 1 see *Section 42.3.10.1: Acceptance filtering*. The Rx FIFO element is described in *Section 42.3.6.1: Rx Buffer and FIFO element*.

To avoid an Rx FIFO overflow, the Rx FIFO watermark can be used. When the Rx FIFO fill level reaches the Rx FIFO watermark configured by RXFnC.FnWM, interrupt flag IR.RFnW is set. When the Rx FIFO Put Index reaches the Rx FIFO Get Index an Rx FIFO Full condition is signalled by RXFnS.FnF. In addition interrupt flag IR.RFnF is set.

Figure 629. Rx FIFO Status



When reading from an Rx FIFO, Rx FIFO Get Index RXFnS.FnGI × FIFO Element Size has to be added to the corresponding Rx FIFO start address RXFnC.FnSA.

Table 684. Rx Buffer / FIFO Element Size

RXESC.RBDS[2:0] RXESC.FnDS[2:0]	Data Field [bytes]	FIFO Element Size [RAM words]
000	8	4
001	12	5
010	16	6
011	20	7
100	24	8
101	32	10

Table 684. Rx Buffer / FIFO Element Size (continued)

RXESC.RBDS[2:0] RXESC.FnDS[2:0]	Data Field [bytes]	FIFO Element Size [RAM words]
110	48	14
111	64	18

42.3.10.2.1 Rx FIFO Blocking Mode

The Rx FIFO blocking mode is configured by RXFnC.FnOM = ‘0’. This is the default operation mode for the Rx FIFOs.

When an Rx FIFO full condition is reached (RXFnS.FnPI = RXFnS.FnGI), no further messages are written to the corresponding Rx FIFO until at least one message has been read out and the Rx FIFO Get Index has been incremented. An Rx FIFO full condition is signalled by RXFnS.FnF = ‘1’. In addition interrupt flag IR.RFnF is set.

In case a message is received while the corresponding Rx FIFO is full, this message is discarded and the message lost condition is signalled by RXFnS.RFnL = ‘1’. In addition interrupt flag IR.RFnL is set.

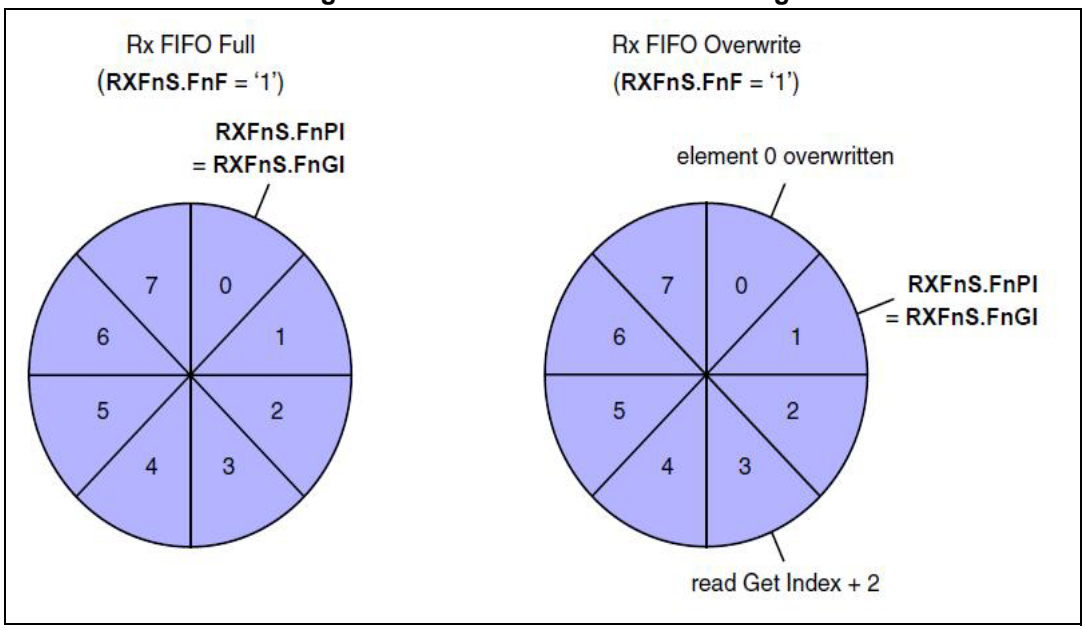
42.3.10.2.2 Rx FIFO Overwrite Mode

The Rx FIFO overwrite mode is configured by RXFnC.FnOM = ‘1’.

When an Rx FIFO full condition (RXFnS.FnPI = RXFnS.FnGI) is signalled by RXFnS.FnF = ‘1’, the next message accepted for the FIFO will overwrite the oldest FIFO message. Put and get index are both incremented by one.

When an Rx FIFO is operated in overwrite mode and an Rx FIFO full condition is signalled, reading of the Rx FIFO elements should start at least at get index + 1. The reason for that is, that it might happen, that a received message is written to the Message RAM (put index) while the CPU is reading from the Message RAM (get index). In this case inconsistent data may be read from the respective Rx FIFO element. Adding an offset to the get index when reading from the Rx FIFO avoids this problem. The offset depends on how fast the CPU accesses the Rx FIFO. *Figure 630* shows an offset of two with respect to the get index when reading the Rx FIFO. In this case the two messages stored in element 1 and 2 are lost.

Figure 630. Rx FIFO Overflow Handling



After reading from the Rx FIFO, the number of the last element read has to be written to the Rx FIFO Acknowledge Index RXFnA.FnA. This increments the get index to that element number. In case the put index has not been incremented to this Rx FIFO element, the Rx FIFO full condition is reset (RXFnS.FnF = '0').

42.3.11 Dedicated Rx Buffers

The M_CAN supports up to 64 dedicated Rx Buffers. The start address of the dedicated Rx Buffer section is configured via RXBC.RBSA.

For each Rx Buffer a Standard or Extended Message ID Filter Element with SFEC / EFEC = '111' and SFID2 / EFID2[10:9] = '00' has to be configured (see *Section 42.3.6.4: Standard Message ID Filter element* and *Section 42.3.6.5: Extended Message ID Filter element*).

After a received message has been accepted by a filter element, the message is stored into the Rx Buffer in the Message RAM referenced by the filter element. The format is the same as for an Rx FIFO element. In addition the flag IR.DRX (Message stored in Dedicated Rx Buffer) in the interrupt register is set.

Table 685. Example Filter Configuration for Rx buffers

Filter Element	SFID1[10:0] EFID1[28:0]	SFID2[10:9] EFID2[10:9]	SFID2[5:0] EFID2[5:0]
0	ID debug message 1	00	00 0000
1	ID debug message 2	00	00 0001
2	ID debug message 3	00	00 0010

After the last word of a matching received message has been written to the Message RAM, the respective New Data flag in register NDAT1,2 is set. As long as the New Data flag is set, the respective Rx Buffer is locked against updates from received matching frames. The New Data flags have to be reset by the Host by writing a '1' to the respective bit position.

While an Rx Buffer's New Data flag is set, a Message ID Filter Element referencing this specific Rx Buffer will not match, causing the acceptance filtering to continue. Following Message ID Filter Elements may cause the received message to be stored into another Rx Buffer, or into an Rx FIFO, or the message may be rejected, depending on filter configuration.

42.3.11.1 Rx Buffer Handling

- Reset interrupt flag IR.DRX
- Read New Data registers
- Read messages from Message RAM
- Reset New Data flags of processed messages

42.3.12 Debug on CAN Support

Debug messages are stored into Rx Buffers. For debug handling three consecutive Rx buffers (for example #61, #62, #63) have to be used for storage of debug messages A, B, and C. The format is the same as for an Rx Buffer or an Rx FIFO element (see *Section 42.3.6.1: Rx Buffer and FIFO element*).

Advantage: Fixed start address for the DMA transfers (relative to RXBC.RBSA), no additional configuration required.

For filtering of debug messages Standard / Extended Filter Elements with SFEC / EFEC = '111' have to be set up. Messages matching these filter elements are stored into the Rx Buffers addressed by SFID2 / EFID2[5:0].

After message C has been stored, the DMA request output is activated and the three messages can be read from the Message RAM under DMA control. The RAM words holding the debug messages will not be changed by the M_CAN while DMA request output is activated. The behavior is similar to that of an Rx Buffers with its New Data flag set.

After the DMA has completed the DMA unit sets the DMA acknowledge input . This resets the DMA request output. Now the M_CAN is prepared to receive the next set of debug messages.

42.3.12.1 Filtering for Debug Messages

Filtering for debug messages is done by configuring one Standard / Extended Message ID Filter Element for each of the three debug messages. To enable a filter element to filter for debug messages SFEC / EFEC has to be programmed to '111'. In this case fields SFID1 / SFID2 and EFID1 / EFID2 have a different meaning. While SFID2 /EFID2[10:9] controls the debug message handling state machine, SFID2 / EFID2[5:0] controls the location for storage of a received debug message.

When a debug message is stored, neither the respective New Data flag nor IR.DRX are set. The reception of debug messages can be monitored via RXF1S.DMS.

Table 686. Example Filter Configuration for debug messages

Filter Element	SFID1[10:0] EFID1[28:0]	SFID2[10:9] EFID2[10:9]	SFID2[5:0] EFID2[5:0]
0	ID debug message A	01	11 1101

Table 686. Example Filter Configuration for debug messages (continued)

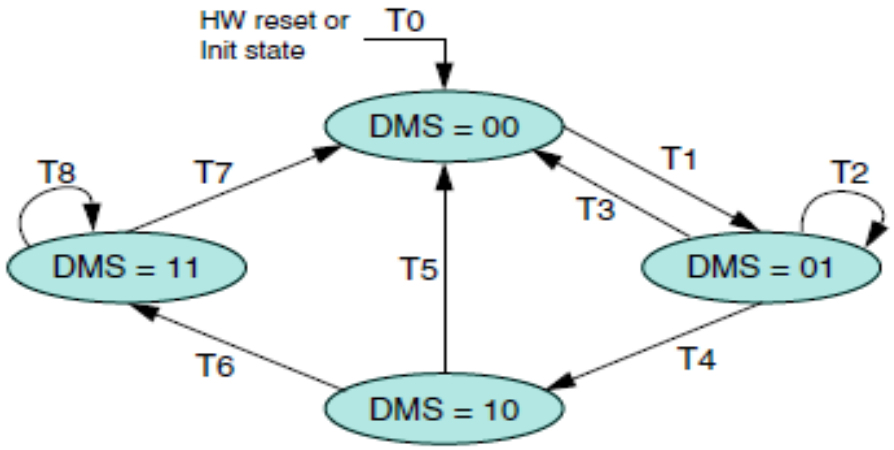
Filter Element	SFID1[10:0] EFID1[28:0]	SFID2[10:9] EFID2[10:9]	SFID2[5:0] EFID2[5:0]
1	ID debug message B	10	11 1110
2	ID debug message C	11	11 1111

42.3.12.2 Debug Message Handling

The debug message handling state machine assures that debug messages are stored to three consecutive Rx Buffers in correct order. In case of missing messages the process is restarted. The DMA request is activated only when all three debug messages A, B, C have been received in correct order.

The status of the debug message handling state machine is signaled via RXF1S.DMS.

Figure 631. Debug Message Handling State Machine



T0: Reset m_cam_dma_req output, enable reception of debug messages A, B, and C
T1: Reception of debug message A
T2: Reception of debug message A
T3: Reception of debug message C
T4: Reception of debug message B
T5: Reception of debug messages A, B
T6: Reception of debug message C
T7: DMA transfer completed
T8: Reception of debug message A,B,C (message rejected)

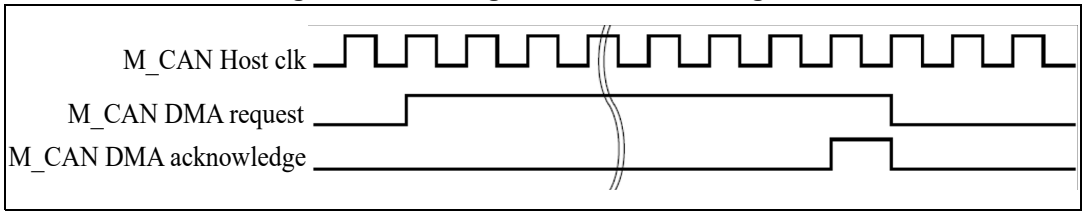
42.3.12.3 Interface to DMA Controller

When all three debug messages A, B, C have been received in the correct order, M_CAN DMA request signal is activated to trigger a DMA transfer. The RAM words holding debug messages A, B, C will not be changed by the M_CAN while M_CAN DMA request signal is active.

After the transfer of the received messages has completed the DMA unit activates the M_CAN DMA ACK signal. This resets M_CAN DMA request signal. The debug message

handling state machine enters idle state (DMS = ‘00’) and waits for reception of the next debug messages.

Figure 632. Timing of DMA interface signals



Note: If the DMA unit activates input M_CAN DMA acknowledge signal before the DMA transfer has completed, the Rx Buffer elements holding debug messages A, B, C are unlocked and may be overwritten by received debug messages.

42.3.13 Tx handling

The Tx Handler handles transmission requests for the dedicated Tx Buffers, the Tx FIFO, and the Tx Queue. It controls the transfer of transmit messages to the CAN core, the Put and Get Indices, and the Tx Event FIFO. Up to 32 Tx Buffers can be set up for message transmission. The Tx Buffer element is described in the *Note: on page 1201*.

The Tx Handler starts a Tx scan to check for the highest priority pending Tx request (Tx Buffer with lowest Message ID) when the Tx Buffer Request Pending register TXBRP is updated, or when a transmission has been started.

Note: AUTOSAR requires at least three Tx Queue Buffers and support of transmit cancellation.

42.3.13.1 Transmit Pause

The transmit pause feature is intended for use in CAN systems where the CAN message identifiers are (permanently) specified to specific values and cannot easily be changed. These message identifiers may have a higher CAN arbitration priority than other defined messages, while in a specific application their relative arbitration priority should be inverse. This may lead to a case where one ECU sends a burst of CAN messages that cause another ECU’s CAN messages to be delayed because that other messages have a lower CAN arbitration priority.

If for example CAN ECU-1 has the transmit pause feature enabled and is requested by its application software to transmit four messages, it will, after the first successful message transmission, wait for two CAN bit times of bus idle before it is allowed to start the next requested message. If there are other ECUs with pending messages, those messages are started in the idle time, they would not need to arbitrate with the next message of ECU-1. After having received a message, ECU-1 is allowed to start its next transmission as soon as the received message releases the CAN bus.

The transmit pause feature is controlled by bit CCCR.TXP. If the bit is set, the M_CAN will, each time it has successfully transmitted a message, pause for two CAN bit times before starting the next transmission. This enables other CAN nodes in the network to transmit messages even if their messages have lower prior identifiers. Default is transmit pause disabled (CCCR.TXP = ‘0’).

This feature looses up burst transmissions coming from a single node and it protects against “babbling idiot” scenarios where the application program erroneously requests too many transmissions.

42.3.13.2 Dedicated Tx Buffers

Dedicated Tx Buffers are intended for message transmission under complete control of the Host CPU. Each Dedicated Tx Buffer is configured with a specific Message ID. In case that multiple Tx Buffers are configured with the same Message ID, the Tx Buffer with the lowest buffer number is transmitted first.

If the data section has been updated, a transmission is requested by an Add Request via TXBAR[ARn]. The requested messages arbitrate internally with messages from an optional Tx FIFO or Tx Queue and externally with messages on the CAN bus, and are sent out according to their Message ID.

A Dedicated Tx Buffer allocates Element Size 32-bit words in the Message RAM (see *Table 687*). Therefore the start address of a dedicated Tx Buffer in the Message RAM is calculated by adding transmit buffer index (0…31) × Element Size to the Tx Buffer Start Address TXBC.TBSA.

Table 687. Tx Buffer / FIFO / Queue Element Size

TXESC.TBDS[2:0]	Data Field [bytes]	Element Size [RAM words]
000	8	4
001	12	5
010	16	6
011	20	7
100	24	8
101	32	10
110	48	14
111	64	18

42.3.13.3 Tx FIFO

Tx FIFO operation is configured by programming TXBC[TFQM] to ‘0’. Messages stored in the Tx FIFO are transmitted starting with the message referenced by the Get Index TXFQS[TFGI]. After each transmission the Get Index is incremented cyclically until the Tx FIFO is empty. The Tx FIFO enables transmission of messages with the same Message ID from different Tx Buffers in the order these messages have been written to the Tx FIFO. The M_CAN calculates the Tx FIFO Free Level TXFQS[TFFL] as difference between Get and Put Index. It indicates the number of available (free) Tx FIFO elements.

New transmit messages have to be written to the Tx FIFO starting with the Tx Buffer referenced by the Put Index TXFQS[TFQPI]. An Add Request increments the Put Index to the next free Tx FIFO element. When the Put Index reaches the Get Index, Tx FIFO Full (TXFQS[TFQF] = ‘1’) is signaled. In this case no further messages should be written to the Tx FIFO until the next message has been transmitted and the Get Index has been incremented.

When a single message is added to the Tx FIFO, the transmission is requested by writing a ‘1’ to the TXBAR bit related to the Tx Buffer referenced by the Tx FIFOs Put Index.

When multiple (n) messages are added to the Tx FIFO, they are written to n consecutive Tx Buffers starting with the Put Index. The transmissions are then requested via TXBAR. The

Put Index is then cyclically incremented by n. The number of requested Tx buffers should not exceed the number of free Tx Buffers as indicated by the Tx FIFO Free Level.When a transmission request for the Tx Buffer referenced by the Get Index is cancelled, the Get Index is incremented to the next Tx Buffer with pending transmission request and the Tx FIFO Free Level is recalculated. When transmission cancellation is applied to any other Tx Buffer, the Get Index and the FIFO Free Level remain unchanged.

A Tx FIFO element allocates Element Size 32-bit words in the Message RAM (see *Table 687*). Therefore the start address of the next available (free) Tx FIFO Buffer is calculated by adding Tx FIFO/Queue Put Index TXFQS.TFQPI (0…31) × Element Size to the Tx Buffer Start Address TXBC.TBSA.

42.3.13.4 Tx Queue

Tx Queue operation is configured by programming TXBC[TFQM] to '1'. Messages stored in the Tx Queue are transmitted starting with the message with the lowest Message ID (highest priority). In case that multiple Queue Buffers are configured with the same Message ID, the Queue Buffer with the lowest buffer number is transmitted first. New messages have to be written to the Tx Buffer referenced by the Put Index TXFQS[TFQPI]. An Add Request cyclically increments the Put Index to the next free Tx Buffer. In case that the Tx Queue is full (TXFQS[TFQF] = '1'), the Put Index is not valid and no further message should be written to the Tx Queue until at least one of the requested messages has been sent out or a pending transmission request has been cancelled.

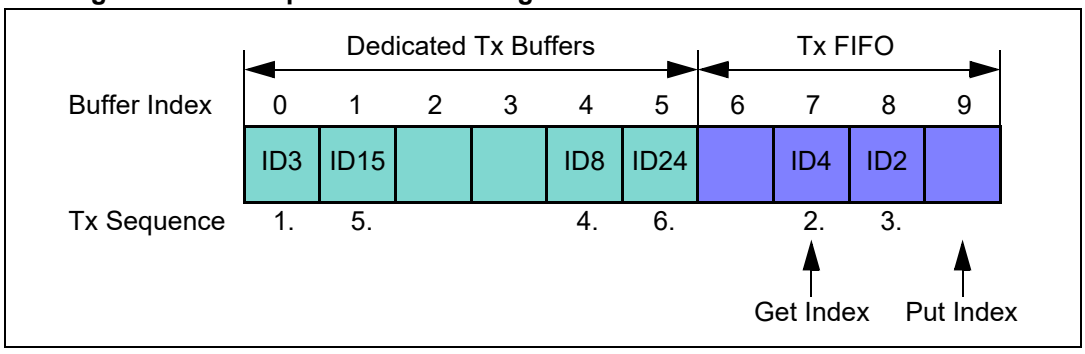
The application may use register TXBRP instead of the Put Index and may place messages to any Tx Buffer without pending transmission request.

A Tx Queue Buffer allocates Element Size 32-bit words in the Message RAM (see *Table 687*). Therefore the start address of the next available (free) Tx Queue Buffer is calculated by adding Tx FIFO/Queue Put Index TXFQS.TFQPI (0…31) × Element Size to the Tx Buffer Start Address TXBC.TBSA.

42.3.13.5 Mixed dedicated Tx Buffers / Tx FIFO

In this case the Tx Buffers section in the Message RAM is subdivided into a set of Dedicated Tx Buffers and a Tx FIFO. The number of Dedicated Tx Buffers is configured by TXBC[NDTB]. The number of Tx Buffers assigned to the Tx FIFO is configured by TXBC[TFQS]. In case TXBC[TFQS] is programmed to zero, only Dedicated Tx Buffers are used.

Figure 633. Example of mixed Configuration Dedicated Tx Buffers / Tx FIFO



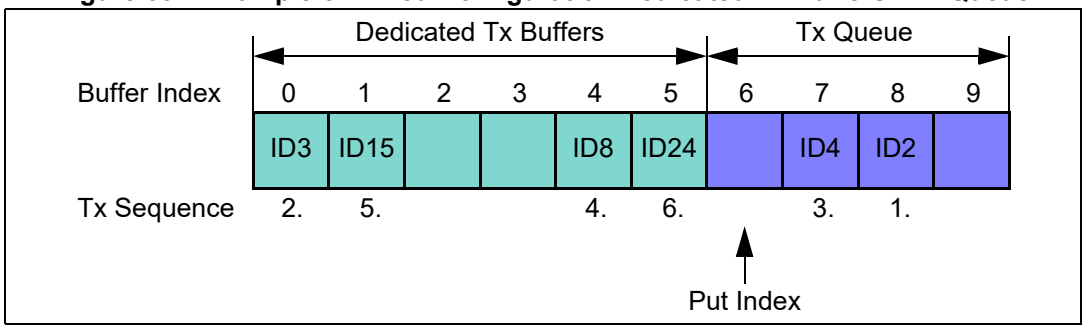
Tx prioritization:

- Scan Dedicated Tx Buffers and oldest pending Tx FIFO Buffer (referenced by TXFS[TFGI])
- Buffer with lowest Message ID gets highest priority and is transmitted next

42.3.13.6 Mixed dedicated Tx Buffers / Tx Queue

In this case the Tx Buffers section in the Message RAM is subdivided into a set of Dedicated Tx Buffers and a Tx Queue. The number of Dedicated Tx Buffers is configured by TXBC[NDTB]. The number of Tx Queue Buffers is configured by TXBC[TFQS]. In case TXBC[TFQS] is programmed to zero, only Dedicated Tx Buffers are used.

Figure 634. Example of mixed Configuration Dedicated Tx Buffers / Tx Queue



Tx prioritization:

- Scan all Tx Buffers with activated transmission request
- Tx Buffer with lowest Message ID gets highest priority and is transmitted next

42.3.13.7 Transmit cancellation

The M_CAN supports transmit cancellation. This feature is especially intended for gateway applications and AUTOSAR based applications. To cancel a requested transmission from a dedicated Tx Buffer or a Tx Queue Buffer the Host has to write a ‘1’ to the corresponding bit position (=number of Tx Buffer) of register TXBCR. Transmit cancellation is not intended for Tx FIFO operation.

Successful cancellation is signaled by setting the corresponding bit of register TXBCF to ‘1’.

In case a transmit cancellation is requested while a transmission from a Tx Buffer is already ongoing, the corresponding TXBRP bit remains set as long as the transmission is in progress. If the transmission was successful, the corresponding TXBTO and TXBCF bits are set. If the transmission was not successful, it is not repeated and only the corresponding TXBCF bit is set.

Note: In case a pending transmission is cancelled immediately before this transmission could have been started, there follows a short time window where no transmission is started even if another message is also pending in this node. This may enable another node to transmit a message which may have a lower priority than the second message in this node.

42.3.13.8 Tx Event handling

To support Tx event handling the M_CAN has implemented a Tx Event FIFO. After the M_CAN has transmitted a message on the CAN bus, Message ID and timestamp are stored

in a Tx Event FIFO element. To link a Tx event to a Tx Event FIFO element, the Message Marker from the transmitted Tx Buffer is copied into the Tx Event FIFO element.

The Tx Event FIFO can be configured to a maximum of 32 elements. The Tx Event FIFO element is described in the Note *Depending on the configuration of the element size (TXESC), between two and sixteen 32-bit words (Tn = 3 to 17) are used for storage of a CAN message's data field.*. When a Tx Event FIFO full condition is signaled by IR.TEFF, no further elements are written to the Tx Event FIFO until at least one element has been read out and the Tx Event FIFO Get Index has been incremented. In case a Tx event occurs while the Tx Event FIFO is full, this event is discarded and interrupt flag IR.TEFL is set.

To avoid a Tx Event FIFO overflow, the Tx Event FIFO watermark can be used. When the Tx Event FIFO fill level reaches the Tx Event FIFO watermark configured by TXEFC[EFWM], interrupt flag IR.TEFW is set.

When reading from the Tx Event FIFO, two times the Tx Event FIFO Get Index TXEFS[EFGI] has to be added to the Tx Event FIFO start address TXEFC[EFSA].

42.3.14 FIFO acknowledge handling

The Get Indices of Rx FIFO 0, Rx FIFO 1, and the Tx Event FIFO are controlled by writing to the corresponding FIFO Acknowledge Index (see *Section 42.3.5.2.28: Rx FIFO 0 Acknowledge Register (RXF0A)* and *Section 42.3.5.2.46: Tx Event FIFO Acknowledge Register (TXEFA)*). Writing to the FIFO Acknowledge Index will set the FIFO Get Index to the FIFO Acknowledge Index plus one and thereby updates the FIFO Fill Level. There are two use cases:

- When only a single element has been read from the FIFO (the one being pointed to by the Get Index), this Get Index value is written to the FIFO Acknowledge Index.
- When a sequence of elements has been read from the FIFO, it is sufficient to write the FIFO Acknowledge Index only once at the end of that read sequence (value: Index of the last element read), to update the FIFOs Get Index.

Due to the fact that the CPU has free access to the M_CAN's Message RAM, special care has to be taken when reading FIFO elements in an arbitrary order (Get Index not considered). This might be useful when reading a High Priority Message from one of the two Rx FIFOs. In this case the FIFOs Acknowledge Index should not be written because this would set the Get Index to a wrong position and also alters the FIFOs Fill Level. In this case some of the older FIFO elements would be lost.

Note: The application has to ensure that a valid value is written to the FIFO Acknowledge Index. The M_CAN does not check for erroneous values.

42.3.15 Clock Calibration on CAN Unit (CCCU)

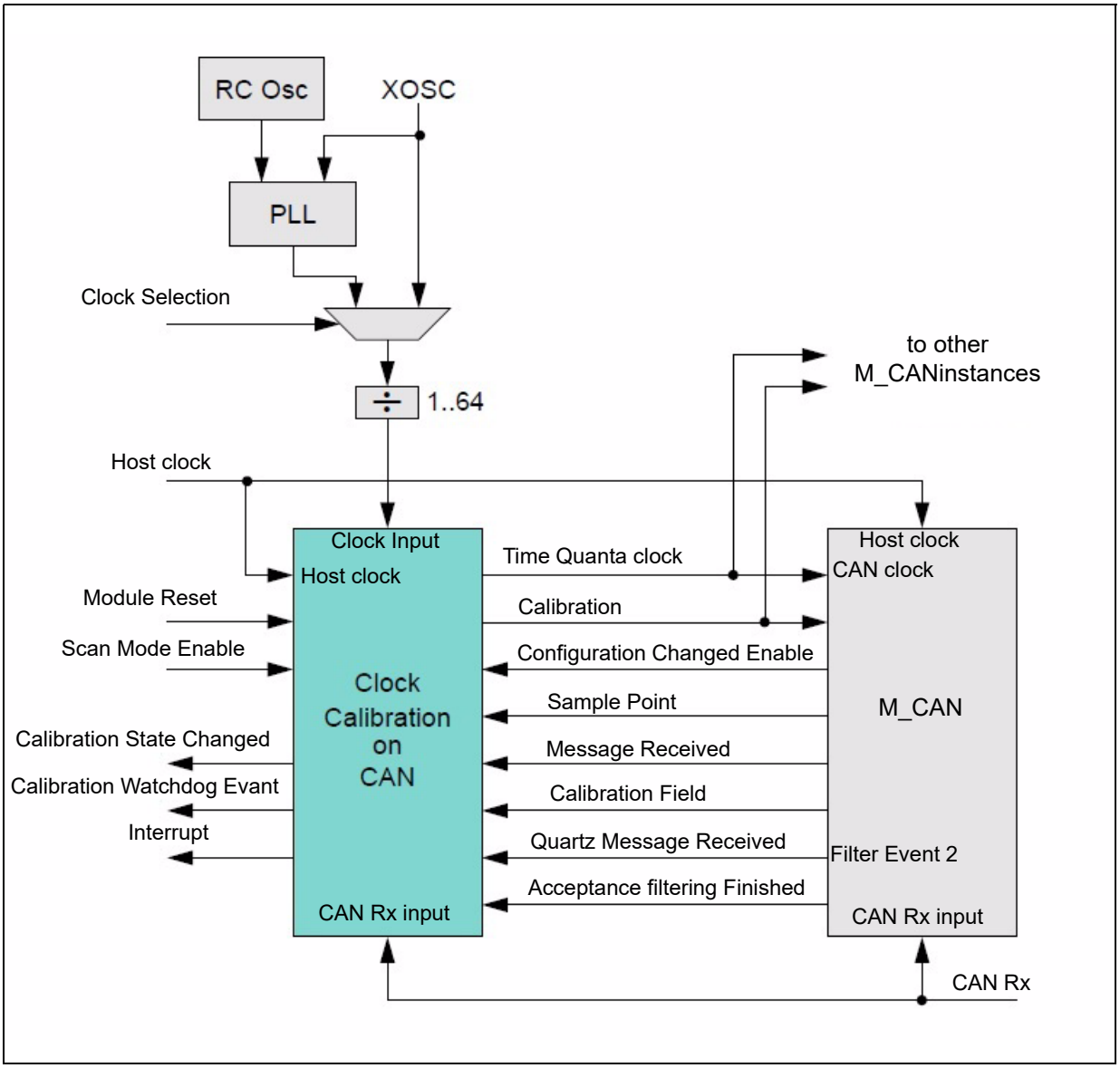
This section describes an add-on for the M_CAN modules that generates a calibrated CAN Time Quanta Clock from an on-chip RC-oscillator clock by evaluating CAN messages received by the attached M_CAN. Calibration on CAN is only possible when the M_CAN is operated in standard CAN mode.

Note: If calibration unit is intended to generate the protocol clock, then it is mandatory to configure calibration unit and MCAN1 before use the other CAN controller like MCAN2 and MCAN3, because the calibrated clock is generated through MCAN1.

Figure 635 shows the interface of the Clock Calibration on CAN Unit. The clock multiplexer controlled by clock input from Clock Select multiplexer controls whether the clock from the

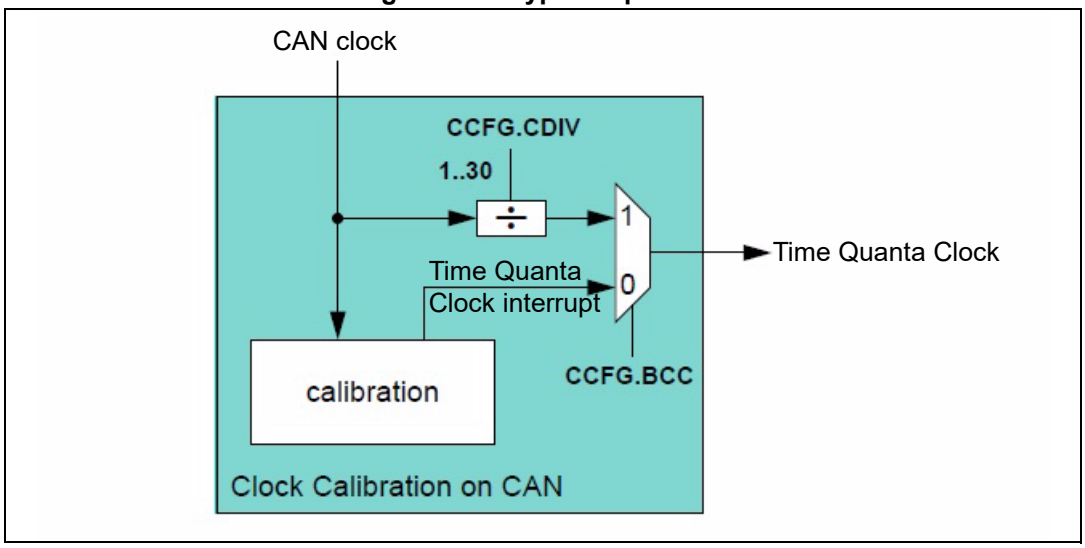
PLL or XOSC is routed to Clock Select multiplexer input. The calibrated Time Quanta Clock is routed to all CAN instances. In case of a successful calibration this is signaled via Calibration.

Figure 635. Connection of Clock Calibration on CAN Unit to M_CAN



CPU accesses to the Clock Calibration on CAN Unit use the Host clock connected to Host clock input from Clock Select Multiplexer. Clock calibration is bypassed when CCFG.BCC = '1'. In this case, the clock from Clock Select multiplexer is routed through a clock divider to Time Quanta Clock output (see *Figure 636*).

Figure 636. Bypass operation



42.3.15.1 Operating Conditions

The Clock Calibration on CAN Unit is designed to operate under the following conditions:

- Clock input from PLL (clock input from Clock Select multiplexer) between 80 MHz and 500 MHz
- M_CAN bit rates between 125 kbit/s and 1 Mbit/s
- Clock for Host accesses (Host clock) is 40/50 MHz

The Clock Calibration on CAN Unit generates a calibrated Time Quanta clock in the range from 0.5 MHz to 25 MHz.

Note: The M_CAN requires that the CAN clock is always below or equal to the Host clock (CAN clock ≤ Host clock). This has to be considered when the Clock Calibration on CAN Unit is bypassed (CCFG.BCC = '1').

42.3.15.2 Calibration Accuracy

The calibration accuracy in state Precision_Calibrated depends on:

- The dynamic clock tolerance at the clock input from clock select multiplexer
- The measurement error. For each bit sequence used for calibration measurement, there is a maximum error of one clock input from Clock Select multiplexer period. The number of bits used for measurement of the bit time is 32 or 64-bit, depending on configuration of CCFG.CFL.
- Tolerable error in calibration mechanism

The distance between two calibration messages has to be chosen to fit the clock tolerance requirements of the attached M_CANs.

Note: Dynamic clock tolerance is the clock frequency variation between two calibration messages, such as, caused by change of temperature or operating voltage.

42.3.15.3 Memory map and register description

42.3.15.3.1 Hardware Reset Description

After hardware reset, the registers of the Clock Calibration on CAN Unit hold the reset values listed in *Table 688*.

The Clock Calibration on CAN Unit allocates an address space of 16 bytes. All registers are organized as 32-bit registers. The Clock Calibration on CAN Unit is accessible by the Host CPU using a data width of 8-bit (byte access), 16-bit (half-word access), or 32-bit (word access). Write access by the Host CPU to registers/bits marked with “P=Protected Write” is possible only when Configuration Change Enable input = ‘1’. Configuration Change Enable signal is activated when the M_CAN control bits CCCR.CCE = ’1’ AND CCCR.INIT = ‘1’.

Table 688. CCCU memory map

Address Offset	Register Name	Access(1)	Reset value	Location
0x0000	Core Release Register (CREL)	R	0x1114_0729	*Section 42.3.15.3.2*
0x0004	Calibration Configuration Register (CCFG)	RP	0x0000_0404	*Section 42.3.15.3.3*
0x0008	Calibration Status Register (CSTAT)	R	0x0203_FFFF	*Section 42.3.15.3.4*
0x000C	Calibration Watchdog (CWD)	RP	0x0000_0000	*Section 42.3.15.3.5*
0x0010	CU Interrupt Register (CUIR)	R/W	0x0000_0000	*Section 42.3.15.3.6*
0x0014	CU Interrupt Enable (CUIE)	R/W	0x0000_0000	*Section 42.3.15.3.7*

1. R = Read, W = Write, P = Protected Write

42.3.15.3.2 Core Release Register (CREL)

Address: 0x0000 Access: Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	REL				STEP				SUBSTEP				YEAR			
W																
Reset(1)	-r	-r	-r	-r	-r	-r	-r	-r	-r	-r	-r	-r	-d	-d	-d	-d

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	MON								DAY							
W																
Reset(1)	-d	-d	-d	-d	-d	-d	-d	-d	-d	-d	-d	-d	-d	-d	-d	-d

1. -r = release, -d = time stamp, the coding of revisions depends on the module version used in the device. For the device reset value refer to *Table 688: CCCU memory map*.

Figure 637. Core Release Register (CREL)

Table 689. CREL field descriptions

Field	Description
0:3 REL	Core Release One digit, BCD-coded.
4:7 STEP	Step of Core Release One digit, BCD-coded.
8:11 SUBSTEP	Substep of Core Release One digit, BCD-coded.
12:15 YEAR	Time Stamp Year One digit, BCD-coded.
16:23 MON	Time Stamp Month Two digits, BCD-coded.
24:31 DAY	Time Stamp Day Two digits, BCD-coded.

Table 690. Example for Coding of Revisions

Release	Step	SubStep	Year	Month	Day	Name
1	0	0	2	02	20	Revision 1.0.0, Date 2012/02/20

42.3.15.3.3 Calibration Configuration Register (CCFG)

Address: 0x0004 Access: RP

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	SWR	0	0	0	0	0	0	0	0	0	0	0	CDIV(1)			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	OCPM(1)								CFL(1)	BCC(1)	0	TQBT(1)				
W																
Reset	0	0	0	0	0	1	0	0	0	0/1	0	0	0	1	0	0

1. Write access by the Host CPU to registers/bits marked with “P=Protected Write” is possible only when Configuration Change Enable input = ‘1’. Configuration Change Enable signal is activated when the M_CAN control bits CCCR.CCE = ‘1’ AND CCCR.INIT = ‘1’.

Figure 638. Calibration Configuration Register (CCFG)

Table 691. CCFG field descriptions

Field	Description
0 SWR	Software Reset Writing a ‘1’ to this bit will reset the calibration FSM to state Not_Calibrated (CSTAT.CALS = ‘00’). The Calibration Watchdog value CWD.WDV is also reset. Registers CCFG, CSTAT and the Calibration Watchdog configuration CWD.WDC are unchanged. The bit remains set until reset has completed.
12:15 CDIV	Clock Divider The clock divider has to be configured when the clock calibration is bypassed (BCC = ‘1’) to assure that the M_CAN requirement CAN clock <= Host clock is fulfilled. 0000 Divide by 1 0001 Divide by 2 0010 Divide by 4 0011 Divide by 6 0100 Divide by 8 0101 Divide by 10 0110 Divide by 12 0111 Divide by 14 1000 Divide by 16 1001 Divide by 18 1010 Divide by 20 1011 Divide by 22 1100 Divide by 24 1101 Divide by 26 1110 Divide by 28 1111 Divide by 30
16:23 OCPM	Oscillator Clock Periods Minimum (0x00–0xFF) Configures the minimum number of clock input from Clock Select multiplexer periods in two CAN bit times. OCPM is used in Basic Calibration to avoid false measurements in case of glitches on the bus line. The configured number of clock input from Clock Select multiplexer periods is OCPM × 32. The configuration depends on the clock input from Clock Select multiplexer frequency (80 MHz to 500 MHz) and the bit rate configured in the attached M_CANs (125 kbit/s to 1 Mbit/s). It is recommended to configure a value slightly below two CAN bit times. The reset value is 1.6 bit times at 80 MHz clock input from Clock Select multiplexer and 1 Mbit/s CAN bit rate.
24 CFL	Calibration Field Length 0 Calibragation field length is 32 bits. 1 Calibragation field length is 64 bits.

Table 691. CCFG field descriptions (continued)

Field	Description
25 BCC	Bypass Clock Calibration If this bit is set, the clock input clock input from Clock Select multiplexer is routed to the Time Quanta clock output through a clock divider configurable via CDIV, Calibration is always ‘1’. In this case the baud rate prescaler of the connected M_CANs has to be configured to generate the M_CAN internal time quanta clock. 0 Clock calibration unit generates Time Quanta clock 1 Clock calibration unit bypassed **Note:** As long as clock input from Clock Select multiplexer is equal or above 80 MHz the Clock Calibration on CAN Unit is functional, even when BCC = ‘1’. The calibration state can be read from register CSTAT.
27:31 TQBT	Time Quanta per Bit Time (0x04–0x19) Configures the number of time quanta per bit time. Same value as configured in the attached M_CANs. The range of the resulting Time Quanta clock is from 0,5 MHz (bit rate of 125 kbit/s with 4 tq per bit time) to 25 MHz (bit rate of 1 Mbit/s with 25 tq per bit time). Valid values are 4 to 25. Configured values below 4 are interpreted as 4, values above 25 are interpreted as 25.

42.3.15.3.4 Calibration Status Register (CSTAT)

Address: 0x0008 Access: Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	CALS		0		TQC										OCPC	
W																
Reset	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	1

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	OCPC															
W																
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Figure 639. Calibration Status Register (CSTAT)

Table 692. CSTAT field descriptions

Field	Description
0:1 CALS	Calibration State 00 Not_Calibrated 01 Basic_Calibrated 10 Precision_Calibrated 11 Reserved

Table 692. CSTAT field descriptions (continued)

Field	Description
3:13 TQC	Time Quanta Counter (0x000–0x3FF) Captured number of time quanta in calibration field (32 or 64 bits). Only valid when the clock calibration unit is in state Precision_Calibrated.
14:31 OCPC	Oscillator Clock Period Counter (0x00000–0x3FFFF) Captured number of oscillator clock periods in calibration field (32 or 64 bits). Only valid when the clock calibration unit is in state Precision_Calibrated.

42.3.15.3.5 Calibration Watchdog Register (CWD)

The calibration watchdog is started after the first falling edge when the calibration FSM is in state Not_Calibrated (CSTAT.CALS = ‘00’). In this state the calibration watchdog monitors the Message Received input. In case no message was received until the calibration watchdog has counted down to zero, the calibration FSM stays in state Not_Calibrated (CSTAT.CALS = ‘00’), the counter is reloaded with RWD.WDC and basic calibration is restarted after the next falling edge.

When in state Basic_Calibrated (CSTAT.CALS = ‘01’), the calibration watchdog is restarted with each Message Received = ‘1’). In case no message was received until the calibration watchdog has counted down to zero, the calibration FSM returns to state Not_Calibrated (CSTAT.CALS = ‘00’), the counter is reloaded with RWD.WDC and basic calibration is restarted after the next falling edge.

When a quartz message was received, state Precision_Calibrated (CSTAT.CALS = ‘10’) is entered and the calibration watchdog is restarted. In this state the calibration watchdog monitors the Quartz Message Received input. In case no message from a quartz controlled node was received by the attached M_CAN until the calibration watchdog has counted down to zero, the calibration FSM transits back to state Basic_Calibrated (CSTAT.CALS = ‘01’).

The Calibration Watchdog Counter is clocked by the Sample Point input. The duration of a cu_spt pulse is one Time Quanta clock period. The signal is active when the CAN protocol engine on the attached M_CAN is started, that is when the INIT bit is reset.

A calibration watchdog event is signaled by Calibration Watchdog Event output. The duration of aCalibration Watchdog Event pulse is one Host clock period.

A calibration watchdog event also sets interrupt flag **CUIR.CWE**. If enabled by **CUIE.CWEE**, Interrupt line is activated (set to high). Interrupt line remains active until interrupt flag **CUIR.CWE** is reset.

Address: 0x000C Access: RP

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	WDV															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	WDC(1)															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. Write access by the Host CPU to registers/bits marked with “P=Protected Write” is possible only when input Configuration Change Enable input = ‘1’. Configuration Change Enable inputsignal is activated when the M_CAN control bits CCCR.CCE = ‘1’ AND CCCR.INIT = ‘1’.

Figure 640. Calibration Watchdog (CWD)

Table 693. CWD field descriptions

Field	Description
0:15 WDV	Watchdog Value Actual Calibration Watchdog Counter Value.
16:31 WDC	Watchdog Configuration Start value of the CalibrationWatchdog Counter.With the reset value of ‘00’ the counter is disabled.

42.3.15.3.6 CU Interrupt Register (CUIR)

The flags are set when one of the listed conditions is detected (edge-sensitive). The flags remain set until the Host clears them. A flag is cleared by writing a ‘1’ to the corresponding bit position. Writing a ’0’ has no effect. A hard reset will clear the register. The configuration of CUIE controls whether an interrupt is generated.

Address: 0x0010 Access: R/W

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	CSC	CWE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 641. CU Interrupt Register (CUIR)

Table 694. CUIR field descriptions

Field	Description
30 CSC	Calibration State Changed 0 Calibration State unchanged 1 Calibration State has changed
31 CWE	Calibration Watchdog Event 0 No Calibration Watchdog Event 1 Calibration Watchdog Event occurred

42.3.15.3.7 CU Interrupt Enable Register (CUIE)

The settings in the CU Interrupt Enable register determine whether a status change in the CU Interrupt Register will be signalled on an interrupt line.

Address: 0x0014 Access: R/W

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	CSCE	CWEE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 642. CU Interrupt Enable Register (CUIE)

Table 695. CUIE field descriptions

Field	Description
30 CSCE	Calibration State Changed Enable 0 Interrupt disabled 1 Interrupt enabled
31 CWEE	Calibration Watchdog Event Enable 0 Interrupt disabled 1 Interrupt enabled

42.3.15.4 Functional Description

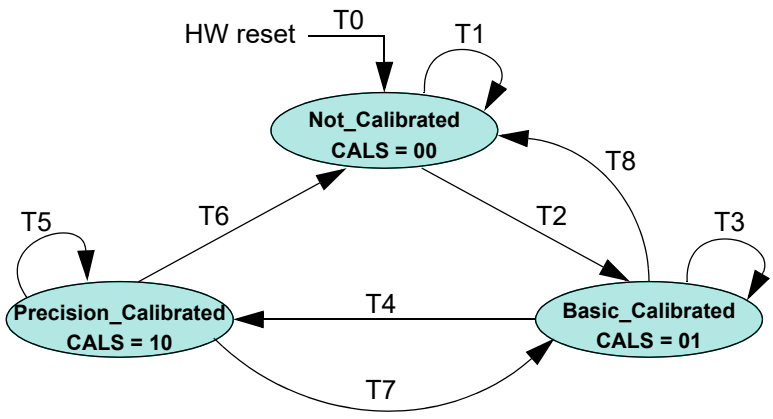
Calibration of the Time Quanta clock via CAN messages is performed by adapting a clock divider that generates the CAN protocol time quantum tq from the clock input from Clock Select multiplexer .

- First step: Basic Calibration–The minimum distance between two edges from recessive to dominant is measured, this time to be assumed two CAN bit times, counted in PLL clock periods. The clock divider is updated each time a new measurement finds a smaller distance between edges. Basic calibration is achieved when the CAN protocol

controller detects a valid CAN message. This is signaled to the Calibration on CAN unit by Message Received input signal.

- Second step: Precision Calibration – The calibration state machine measures the length of a longer bit sequence inside a CAN frame by counting the number of clock input from Clock Select multiplexer periods. The length of this bit sequence can be configured to 32 or 64 bits via CCFG.CLF. For a calibration field length of 32/64 bit a calibration message with at least 2/6 byte data field is required. The counter is controlled by Calibration Field and Sample Point input signals. The counter is enabled when Calibration Field is set and stops after the configured number of bits has been received. Received bits are signaled by the M_CAN by Sample Point signal. Precision calibration is based on the new clock divider value calculated from the measurement of the longer bit sequence.

Figure 643. Calibration FSM



T0: HW reset

T1: Check for minimum

T2: Message Received = ‘1’

T3: Evaluate messages

T4: Quartz Message Received = ‘1’

T5: Evaluate quartz messages

T6: Configuration Change Enabled = ‘1’

T7: Watchdog event: no quartz message received

T8: Watchdog event: no message received
or Configuration Change Enabled = ‘1’

A change of the calibration state is signaled via by Calibration State Changed output. The duration of a Calibration State Changed output pulse is one Host clock period.

A change of the calibration state also sets interrupt flag CUIR.CSC. If enabled by CUIE.CSCE, Interrupt line is activated (set to high). Interrupt line remains active until interrupt flag CUIR.CSC is reset.

Until precision calibration is achieved (Calibration=‘1’), the attached M_CANs operate in a restricted mode (no frame transmission, no error or overload flag transmission, no error counting). In case calibration of the PLL is done by software by evaluating the calibration status from register CSTAT, the attached M_CANs have to be set to restricted operation

mode (CCCR.ASM = ‘1’) until the Calibration on CAN unit is in state Precision_Calibrated (see *Section 42.3.15.4.3: Software Calibration*).

Precision calibration may be performed only on valid CAN frames transmitted by a node with a stable, quartz-controlled clock. Calibration frames are detected by the M_CAN’s acceptance filtering and are signaled to the Clock Calibration on CAN Unit by Quartz Message Received, to be connected to one of the M_CAN’s Filter Event outputs, signal. A filter element and an Rx Buffer have to be configured in the attached M_CAN to identify and store calibration messages. After reception of a calibration message the Rx Buffer’s new data flag has to be reset to enable signaling of the next calibration message.

In case there is only one CAN transmitter with a quartz clock in the network, this node has to transmit its first message after startup with at least one ‘1010’ sequence in the data field or in the identifier. This assures that the non-quartz nodes can enter state Basic_Calibrated and then acknowledge the quartz node’s messages.

Precision calibration must be repeated in predefined maximum intervals supervised by the calibration watchdog.

Note: When the Clock Calibration on CAN Unit transits from state Precision_Calibrated back to Basic_Calibrated, the calibration OK signal is deasserted, the attached M_CANs complete ongoing transmissions, and then enter restricted operation (no frame transmission, no error or overload flag transmission, no error counting).

42.3.15.4.1 Configuration

The Clock Calibration on CAN Unit is configured via register CCFG. The register is only writeable while Configuration Change Enable = ‘1’, that is when the attached M_CAN has CCCR.CCE and CCCR.INIT set.

For basic calibration the minimum number of oscillator periods between two consecutive falling edges at CAN receive input pin is measured. The number of clock periods depends on the clock frequency applied at clock input from clock select multiplexer. In case the measured number of clock periods is below the minimum configured by CCFG.OCPM, for example due to a glitch on CAN receive input, the value is discarded and measurement continues.

It is recommended to configure CCFG.OCPM slightly below two CAN bit times:

CCFG.OCPM < ((2 x CAN bit time) / Clock input from Clock Select multiplexer period) / 32

The length of the bit field used for precision calibration can be configured to 32 or 64 bits via CCFG.CFL. The number of bits used for precision calibration has an impact on calibration accuracy and the maximum distance between two calibration messages. The reception of the bits is signaled by the attached M_CAN by Sample Point input.

The number of time quanta per bit time configured by CCFG.TQBT is used together with the measured number of oscillator clock periods CSTAT.OCPC to define the number of oscillator clocks per bit time.

When the clock calibration is bypassed by configuring CCFG.BCC = ‘1’, the internal clock divider has to be configured via CCFG.CDIV to fulfill the condition CAN clock $\leq$ Host clock.

Note: When clock calibration on CAN is active (CCFG.BCC = ‘0’), the baud rate prescaler of the attached M_CANs has to be configured to inactive.

42.3.15.4.2 Status signaling

The status of the Clock Calibration on CAN Unit can be monitored by reading register CSTAT. When in state Precision_Calibrated the oscillator clock period counter CSTAT.OCPC signals the number of oscillator clock periods in the calibration field while CSTAT.TQC signals the number of time quanta in the calibration field.

The calibration state is monitored by CSTAT.CALS. A change of the calibration state is signalled at the module interface by Calibration State Changed. The duration of a Calibration State Changed pulse is one Host clock period. A change of the calibration state also sets interrupt flag CUIR.CSC. If enabled by CUIE.CSCE, Interrupt line is activated (set to high). Interrupt line remains active until interrupt flag CUIR.CSC is reset.

A calibration watchdog event is signalled by Calibration Watchdog Event output. The duration of a Calibration Watchdog Event output pulse is one Host clock period. A calibration watchdog event also sets interrupt flag CUIR.CWE. If enabled by CUIE.CWEE, Interrupt line is activated (set to high). Interrupt line remains active until interrupt flag CUIR.CWE is reset

42.3.15.4.3 Software Calibration

The Clock Calibration on CAN Unit also supports software calibration of the clock generated from an adjustable PLL, connected to the clock input, from Clock Select multiplexer. For this purpose the clock from Clock Select multiplexer has to be routed to Time Quanta clock output by programming CCFG.BCC = ‘1’. The Clock Calibration on CAN Unit’s internal clock divider has to be configured via CCFG.CDIV to meet the clock requirements of the attached M_CANs. All other configuration parameters have to be set via CCFG as if the Clock Calibration on CAN Unit is generating Time Quanta clock.

For startup the M_CANs attached to the Clock Calibration on CAN Unit have to be configured for Restricted Operation (CCCR.ASM = ‘1’). The PLL clock has to be adjusted until the Clock Calibration on CAN Unit has reached state Precision_Calibrated. Now the software has to reset CCCR.ASM and the M_CANs can start normal operation.

During operation the software has to check regularly whether the Clock Calibration on CAN Unit is still in state Precision_Calibrated. In case the Clock Calibration on CAN Unit has left state Precision_Calibrated due to drift of the PLL clock, the attached M_CANs have to be set into Restricted Operation Mode by programming CCCR.INIT, CCCR.CCE, and CCCR.ASM to ‘1’.

After the PLL clock has been adjusted successfully (Clock Calibration on CAN Unit is in state Precision_Calibrated), the M_CANs can be set to normal operation again.

42.4 CAN RAM arbiter

This block acts as a dynamic arbiter between the various CAN nodes and the CPU for granting access to the shared CAN memory. The dynamic arbitration ensures:

- 50% bandwidth for CPU, 50% shared by the CAN nodes when simultaneous access requested from all masters and CPU,
- Bandwidth dynamic upgrading to requesting masters if other masters do not request access.

Note: From the arbiter prospective the CPU bandwidth may be limited to 50%. The CPU may be waiting stated due to this bandwidth limitation when accessing the shared memory.

42.4.1 Features

- Active low asynchronous reset
- 1–CPU + up to 5–CAN arbiter
- Combined dynamic arbitration scheme: Bandwidth between CPU and M_CAN interfaces can be redistributed. If active, the CPU interface is granted a 50% portion of the available bandwidth
- Bandwidth dynamic upgrading to requesting masters if other masters do not request access
- The SRAM access by a single master needs two cycles (address and data) for each read/write command. The arbiter has a pseudo address pre-fetching mechanism. This mechanism acts when multiple masters are accessing the SRAM and it overlaps the address cycle of one master with the data cycle of another master. In case of multiple masters accessing the SRAM, the pseudo address pre-fetching scheme saves multiple clock cycles
- The peripheral GSI module enable acts as a request from each master. Each master will be made to wait at least one clock cycle before the grant is given. The arbiter ensures that the CPU does not wait for more than 1 clock cycle, hence ensuring a 50% guaranteed bandwidth
- Time slot is defined as one peripheral clock cycle

42.4.2 Functional overview using examples

The dynamic arbitration scheme is explained using the following examples.

- Example 1 (active CPU IF, 3 out of 3 CAN nodes are active):
  - CPU obtains every 2nd slot
  - CAN 1/2/3 obtains every 6th time slot
- Example 2 (inactive CPU IF, 4 out of 4 CAN nodes are active):
  - CPU obtains no time slots
  - CAN 1/2/3/4 obtains every 4th time slot
- Example 3 (active CPU IF, 2 out of 4 CAN node are active):
  - CPU obtains every 2nd time slot
  - Active CAN 1/2 obtains every 4th time slot
- Example 4 (inactive CPU IF, 2 out of 4 CAN node are active):
  - CPU obtains no time slots
  - Active CAN 1/2 obtains every 2nd time slot

42.5 SRAM interface and memory organization

The CAN subsystem will interface with an external RAM using this interface. The RAM sizes required for the CAN subsystem are detailed in *Section 42.8: Shared memory map*.

42.5.1 ECC controller

The RAM embeds ECC logic and code to provide Single Error Correction / Double Error Detection (SECCDED). The module does not guarantee any proper functionality if more than 2 bits are in errors. It supports only 32-bit write access and 8-/16-/32-bit read access.

The ECC error is reported to each CAN module as well as available at the CAN subsystem top, so that the same can be forwarded to the Error Management Module.

The module uses Hamming code for single error correction and double error detection logic. It involves transmitting data with multiple ecc_check bits and decoding the associated ecc_check bits when receiving data to detect errors. The SECDED Hamming code is not able to detect three bit errors. Rather, in the presence of a three bit error, a conventional SECDED code returns an error code that is indistinguishable from an error code resulting from a single bit error. Hence, this module does not guarantee any proper functionality when more than 2-bits are in error.

The ecc_check bits are parallel parity bits generated from XORing certain bits in the original data word. If bit error(s) are introduced in the codeword, several ecc_check bits show parity errors after decoding the retrieved codeword. The combination of these ecc_check bit errors display the nature of the error. In addition, the position of any single bit error is identified from the ecc_check bits.

Note:
The ECC error address is reported at the top of the subsystem using a 16 bit ECC error address signal. This address is valid only when the ECC bit error output is valid.

42.5.1.1 Features

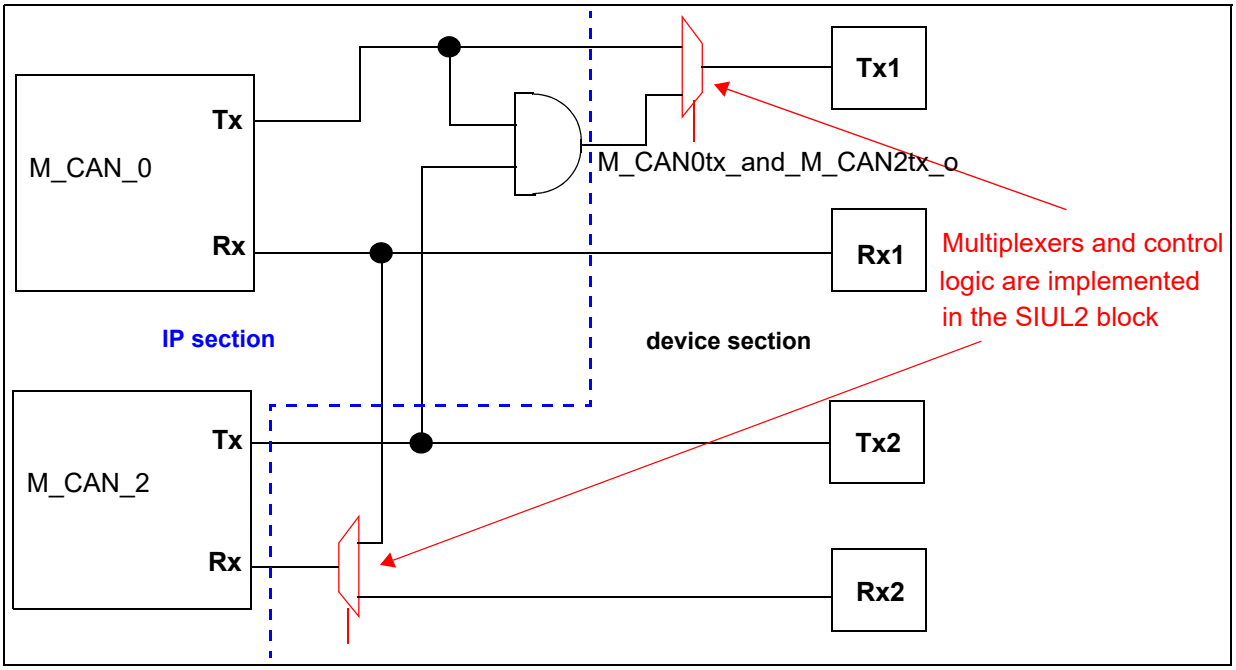
The ECC module supports the following:

- ecc_error = ‘00’ No error is detected in the read data from memory
- ecc_error = ‘01’ This indicates single bit error occurred within the 39-bit codeword. In addition, the error is corrected, and the forwarded data to the master is error free.
- ecc_error = ‘10’ This indicates that a two bit error has occurred within the codeword. In this case, no error correction is possible.
- ecc_error = ‘11’ This indicates that errors beyond the detection capability have occurred within the codeword and no error correction is possible. This is an invalid error type.

42.6 CAN nodes 0 and 2 I/Os sharing

Each CAN node has its Tx/Rx signals connected to separated GPIO pairs. It is required for M_CAN_0 and M_CAN_2 to share one common GPIO pair. This feature is described in *Figure 644*.

Figure 644. CAN Subsystem_0 M_CAN_0 and M_CAN_2 connections to GP I/Os



42.7 External signal description

The CAN subsystem has following external pins.

- M_CAN_x_RX: M_CAN x receive input
- M_CAN_x_TX: M_CAN x transmit output
- M_CAN_x_RXFD: M_CAN_x Receive Fast Data control signal
- M_CAN_x_TXFD: M_CAN_x Transmit Fast Data control signal

42.8 Shared memory map

Table 696 shows the shared memory map vs CAN subsystem transmit, receive and filters elements.

Table 696. Shared memory map versus CAN transmit, receive and filters elements

Start Address Offset (byte)	End Address Offset (byte)	CAN block/sub block		Comment
CAN Subsystem 0 - APB Bridge 0				
0x06E0	0x0EFF	M_CAN0	Standard Filters	This available physical memory space could be configured for Filters/ FIFO/Buffer. This memory space can be flexible, assigned to any of the implemented M_CAN modules.
			Extended Filters	
			Rx FIFO 0	
			Rx FIFO 2	
			Rx Buffer	
			Tx Event FIFO	
			Tx Buffers	
			Reserved	
0x1880	0x1CFF	M_CAN2	Standard Filters	This available physical memory space could be configured for Filters/ FIFO/Buffer. This memory space can be flexible, assigned to any of the implemented M_CAN modules.
			Extended Filters	
			Rx FIFO 0	
			Rx FIFO 2	
			Rx Buffer	
			Tx Event FIFO	
			Tx Buffers	
			Reserved	
CAN Subsystem 1 - APB Bridge 1				
0x0000	0x081F	M_CAN1	Standard Filters	This available physical memory space could be configured for Filters/ FIFO/Buffer. This memory space can be flexible, assigned to any of the implemented M_CAN modules.
			Extended Filters	
			Rx FIFO 0	
			Rx FIFO 1	
			Rx Buffer	
			Tx Event FIFO	
			Tx Buffers	
			Reserved	

Note: There is no write access protection from a CAN module to the message memory space. The write access protection is only for CPU access to the message memory.

The start address for every memory section is to be configured by the user in the relevant CAN modules registers. However there is no hardware check in the RAM interface for the start address consistency.

However the number of filters and the Tx/Rx buffers used, it is mandatory for the user to configure the start address registers, specified in Table 697, in order to fit the memory mapping specified in Table 696.

The CPU uses the peripheral interface to access the memory locations. It must be noted that there is no transfer error reported to the CPU in case of access to reserved memory locations.

Table 697. Detailed mapping in CAN subblocks

CAN subblock	Element name	Comment
Standard Filters	Standard Message ID filter 0 ... Standard Message ID filter 127	Address configured in SIDFC [FLSSA]
Extended Filters	Extended Message ID filter 0 ... Extended Message ID filter 63	Address configured in XIDFC [FLESA]
Rx FIFO0	Rx FIFO0 element 0 ... Rx FIFO0 element 63	Address configured in RXF0C[F0SA]
Rx FIFO1	Rx FIFO1 element 0 ... Rx FIFO1 element 63	Address configured in RXF1C[F1SA]
Rx Buffer	Rx Buffer element 0 ... Rx Buffer element 63	Address configured in RXBC[RBSA]
Tx Event FIFO	Tx event FIFO element 0 ... Tx event FIFO element 31	Address configured in TXEFC[EFSA]
Tx Buffers	Tx Buffer element 0 ... Tx Buffer element 31	Address configured in TXBC[TBSA]

43 Deserial Serial Peripheral Interface (DSPI)

43.1 Introduction

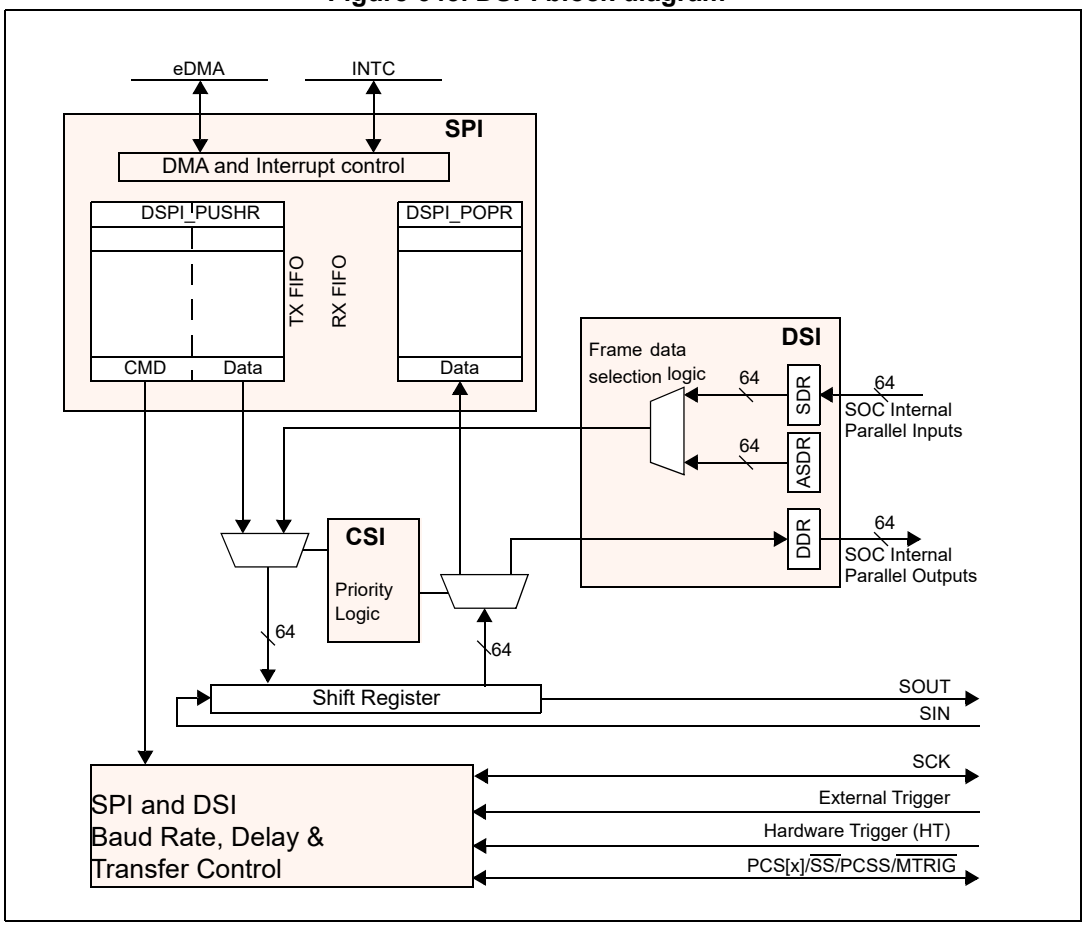
The Deserial Serial Peripheral Interface (DSPI) module provides a synchronous serial bus for communication between an MCU and an external peripheral device. The module supports MCU pin count reduction through serialization and deserialization of MCU internal signals transmitted over the SPI serial link.

Note: For the chip-specific implementation details of this module's instances, see Chapter 7: Device configuration: Section 7.9.2: DSPI configuration.

43.1.1 Block diagram

Figure 645 provides the block diagram of this module.

Figure 645. DSPI block diagram (1)



1. CSI and DSI features are not supported

43.1.2 Features

The DSPI supports these features:

- full-duplex, four-wire synchronous transfers
- Master and Slave modes: data streaming operation in slave mode with continuous slave selection
- buffered transmit operation using the TX FIFO with depth of 4 entries
- buffered receive operation using the RX FIFO with depth of 4 entries
- asynchronous clocking scheme for Register and Protocol interfaces
- TX and RX FIFOs can be disabled individually for low-latency updates to SPI queues
- visibility into TX and RX FIFOs for ease of debugging
- programmable transfer attributes on a per-frame basis:
  - 8 transfer attribute registers along with 8 extended transfer attribute registers
  - serial clock with programmable polarity and phase
  - various programmable delays
  - programmable serial frame size of 4 to 64 bits, expandable by software control: SPI frames longer than 32 bits can be supported using the continuous selection format
  - continuously held chip select capability
  - parity control
- 8 Peripheral Chip Selects, expandable to 256 with external demultiplexer
- deglitching support for up to 128 Peripheral Chip Selects with external demultiplexer
- DMA support for adding entries to TX FIFO and removing entries from RX FIFO:
  - TX FIFO is not full (TFFF)
  - RX FIFO is not empty (RFDF)
  - CMD FIFO is not full (CMDFFF)
- interrupt conditions:
  - End Of Queue reached (EOQF)
  - TX FIFO is not full (TFFF)
  - CMD FIFO is not full (CMDFFF)
  - Transfer of current frame Complete (TCF)
  - Transfers due from current Command frame Complete (CMDTCF)
  - Transfer of current SPI frame Complete (SPITCF)
  - attempt to transmit with an empty Transmit FIFO (TFUF)
  - RX FIFO is not empty (RFDF)
  - frame received while Receive FIFO is full (RFOF)
  - SPI Parity Error (SPEF)
  - data present in TX FIFO while CMD FIFO is empty (TFIWF)
- global interrupt request line
- modified SPI transfer formats for communication with slower peripheral devices
- power-saving architectural features
  - support for stop mode
  - support for doze mode

The DSPI also supports pin reduction through serialization and deserialization if enabled for the module.

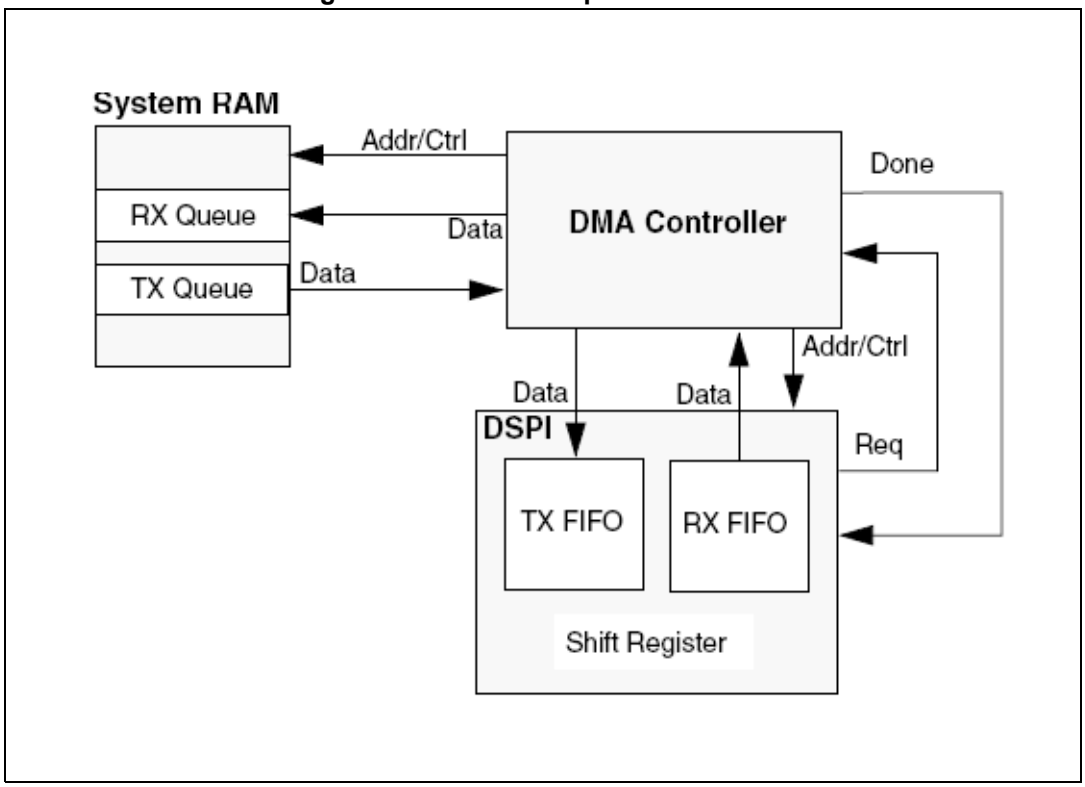
- Two sources of serialized data:
  - DSPI memory-mapped register
  - Parallel input signals
  - Programmable selection of source data on bit basis
- Deserialized data provided as Parallel Output signals and as bits in a memory-mapped register
- interrupt conditions:
  - Deserialized data matches pre-programmed pattern (DDIF)
- DMA request support for following conditions: deserialized data matches pre-programmed pattern (DDIF)
- Transfer initiation conditions:
  - Continuous
  - Edge sensitive hardware trigger
  - Change in data
- Support for parallel and serial chaining of several DSPI modules inside the MCU
- Pin serialization/deserialization with interleaved SPI frames for control and diagnostics
- Continuous serial communications clock

43.1.3 SPI configuration

The SPI configuration allows the DSPI to send and receive serial data. This configuration allows the DSPI to operate as a basic SPI block with internal FIFOs supporting external queues operation. Transmit data and received data reside in separate FIFOs. The host CPU or a DMA controller read the received data from the receive FIFO and write transmit data to the transmit FIFO.

For queued operations, the SPI queues can reside in system RAM which is external to the DSPI. Data transfers between the queues and the DSPI FIFOs are accomplished by a DMA controller or host CPU. *Figure 646* shows a system example with DMA, DSPI and external queues in system RAM.

Figure 646. DSPI with queues and DMA



43.1.4 Modes of operation

The DSPI modes of operation that can be divided into two categories:

- Module-specific modes
  - Master mode
  - Slave mode
  - Module Disable mode
- MCU-specific modes
  - External Stop mode
  - Debug mode

The DSPI enters module-specific modes when the host writes a DSPI register. The MCU-specific modes are controlled by signals external to the DSPI. The MCU-specific modes are modes that an MCU may enter in parallel to the DSPI block-specific modes.

43.1.4.1 Master mode

Master mode allows the DSPI to initiate and control serial communication. In this mode, the SCK signal and the PCS[x] signals are controlled by the DSPI and configured as outputs.

43.1.4.2 Slave mode

Slave mode allows the DSPI to communicate with SPI bus master. In this mode, the DSPI responds to externally controlled serial transfers. The SCK signal and the PCS[0]/SS signals are configured as inputs and driven by a SPI bus master.

43.1.4.3 Module Disable mode

The Module Disable mode can be used for MCU power management. The clock to the non-memory-mapped logic in the DSPI can be stopped while in Module Disable mode.

43.1.4.4 External Stop mode

External Stop mode is used for MCU power management. The DSPI supports the Peripheral Bus Stop mode mechanism. When a request is made to enter External Stop mode, the DSPI block acknowledges the request and completes the transfer in progress. When the DSPI reaches the frame boundary it signals that the protocol clock to the DSPI module may be shut off.

43.1.4.5 Debug mode

Debug mode is used for system development and debugging. The MCR[FRZ] bit controls DSPI behavior in the debug mode. If the bit is set, the DSPI stops all serial transfers, when the MCU is in Debug mode. If the bit is cleared, the MCU Debug mode has no effect on the DSPI.

43.2 DSPI signal description

This section provides the DSPI signals description.

Table 698 lists the signals that may connect off chip depending on device implementation.

Table 698. DSPI signal description

Signal	Description	I/O
PCS0/ $\overline{SS}$	Master mode: Peripheral Chip Select 0 output Slave mode: Slave Select input	I/O
PCS[1:3]	Master mode: Peripheral Chip Select 1–3 Slave mode: Unused	O
PCS4/ $\overline{MTRIG}$	Master mode: Peripheral Chip Select 4 Slave mode: Master Trigger	O
PCS5/ $\overline{PCSS}$	Master mode: Peripheral Chip Select 5 / Peripheral Chip Select Strobe Slave mode: Unused	O
PCS[6:7]	Master mode: Peripheral Chip Select 6–7 Slave mode: Unused	O
SIN	Serial Data In	I
SOUT	Serial Data Out	O
SCK	Master mode: Serial Clock (output) Slave mode: Serial Clock (input)	I/O

43.2.1 PCS0/$\overline{SS}$ — Peripheral Chip Select 0 / Slave Select

In master mode, the PCS0 signal is a Peripheral Chip Select output that selects which slave device the current transmission is intended for.

In slave mode, the active low $\overline{SS}$ signal is a Slave Select input signal that allows a SPI master to select the DSPI as the target for transmission.

43.2.2 PCS[1:3] — Peripheral Chip Selects 1 – 3

PCS1 – PCS3 are Peripheral Chip Select output signals in master mode.

In slave mode, these signals are unused.

43.2.3 PCS4/$\overline{MTRIG}$ — Peripheral Chip Select 4 / Master Trigger

In master mode, PCS4 is a Peripheral Chip Select output signal.

In slave mode, the active low $\overline{MTRIG}$ is an output trigger signal that indicates that a change in data to be serialized has occurred. The $\overline{MTRIG}$ pulse is four protocol clock cycles in duration. If the DSPI is in slave mode and the MTO is disabled, the PCS4/$\overline{MTRIG}$ signal is unused.

43.2.4 PCS5/$\overline{PCSS}$ — Peripheral Chip Select 5 / Peripheral Chip Select Strobe

PCS5 is a Peripheral Chip Select output signal. When the DSPI is in master mode and the MCR[PCSSE] bit is cleared, this signal selects which slave device the current transfer is intended for.

When the DSPI is in master mode and the MCR[PCSSE] bit is set, the $\overline{PCSS}$ signal acts as a strobe to an external peripheral chip select demultiplexer, which decodes the PCS0 – PCS4 and PCS6 – PCS7 signals, preventing glitches on the demultiplexer outputs.

This signal is not used in slave mode.

43.2.5 PCS[6:7] — Peripheral Chip Selects 6 – 7

PCS[6] – PCS[7] are Peripheral Chip Select output signals in master mode. In slave mode, these signals are not used.

43.2.6 SIN — Serial Input

SIN is a serial data input signal.

43.2.7 SOUT — Serial Output

SOUT is a serial data output signal.

43.2.8 SCK — Serial Clock

SCK is a serial communication clock signal. In master mode, the DSPI generates the SCK. In slave mode, SCK is an input from an external bus master.

43.2.9 Memory map/register definition

Register accesses to memory addresses that are reserved or undefined result in a transfer error. Write access to the DSPI*x*_POPR also results in a transfer error.

Note: For the chip-specific implementation details of this module's registers, see Chapter 7: Device configuration: Section 7.9.2.3: DSPI registers.

Table 699. DSPI memory map

Offset (hex)	Register name	Location
0x0000	DSPI Module Configuration Register (DSPI*x*_MCR)	*43.2.9.1 on page 1257*
0x0004-0x0007	Reserved	
0x0008	DSPI Transfer Count Register (DSPI*x*_TCR)	*43.2.9.2 on page 1261*
0x000C	DSPI Clock and Transfer Attributes Register 0 (In Master Mode) (DSPI*x*_CTAR0)	*43.2.9.3 on page 1262*
0x000C	DSPI Clock and Transfer Attributes Register 0 (In Slave Mode) (DSPI*x*_CTAR0_SLAVE)	*43.2.9.4 on page 1267*
0x0010	DSPI Clock and Transfer Attributes Register 1 (In Master Mode) (DSPI*x*_CTAR1)	*43.2.9.3 on page 1262*
0x0010	DSPI Clock and Transfer Attributes Register 1 (In Slave Mode) (DSPI*x*_CTAR1_SLAVE)	*43.2.9.4 on page 1267*
0x0014	DSPI Clock and Transfer Attributes Register 2 (In Master Mode) (DSPI*x*_CTAR2)	*43.2.9.3 on page 1262*
0x0018	DSPI Clock and Transfer Attributes Register 3 (In Master Mode) (DSPI*x*_CTAR3)	*43.2.9.3 on page 1262*
0x001C	DSPI Clock and Transfer Attributes Register 4 (In Master Mode) (DSPI*x*_CTAR4)	*43.2.9.3 on page 1262*
0x0020	DSPI Clock and Transfer Attributes Register 5 (In Master Mode) (DSPI*x*_CTAR5)	*43.2.9.3 on page 1262*
0x0024	DSPI Clock and Transfer Attributes Register 6 (In Master Mode) (DSPI*x*_CTAR6)	*43.2.9.3 on page 1262*
0x0028	DSPI Clock and Transfer Attributes Register 7 (In Master Mode) (DSPI*x*_CTAR7)	*43.2.9.3 on page 1262*
0x002C	DSPI Status Register (DSPI*x*_SR)	*43.2.9.6 on page 1269*
0x0030	DSPI DMA/Interrupt Request Select and Enable Register (DSPI*x*_RSER)	*43.2.9.7 on page 1271*
0x0034	DSPI PUSH FIFO Register In Master Mode (DSPI*x*_PUSHR)	*43.2.9.8 on page 1274*
0x0034	DSPI PUSH FIFO Register In Slave Mode (DSPI*x*_PUSHR_SLAVE)	*43.2.9.9 on page 1277*
0x0038	DSPI POP FIFO Register (DSPI*x*_POPR)	*43.2.9.10 on page 1277*
TX FIFO registers (n=0 to 3)		
0x003C + n*0x4	DSPI Transmit FIFO Register n (DSPI*x*_TXFR*n*)	*43.2.9.11 on page 1278*
0x004C–0x007B	Reserved	

Table 699. DSPI memory map (continued)

Offset (hex)	Register name	Location
RX FIFO registers (n=0 to 3)		
0x007C + n*0x4	DSPI Receive FIFO Register n (DSPI*x*_RXFR*n*)	*43.2.9.12 on page 1279*
0x00BC–0x011B	Reserved	
Extended transfer attributes registers (n=0 to 7)		
0x011C + n*0x4	DSPI Clock and Transfer Attributes Register Extended n (DSPI*x*_CTARE*n*)	*43.2.9.13 on page 1279*
0x013C	DSPI Status Register Extended (DSPI*x*_SREX)	*43.2.9.14 on page 1280*

43.2.9.1 DSPI Module Configuration Register (DSPI*x*_MCR)

The DSPI*x*_MCR contains bits to configure various attributes associated with DSPI operations. The HALT and MDIS bits can be changed at any time, but they only take effect on the next frame boundary. Only the HALT and MDIS bits in the MCR can be changed while the DSPI is in the Running state.

Offset: 0x0000 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	MSTR	CONT_SCKE	DCONF		FRZ	MTFE	PCSSE	ROOE	PCSIS7	PCSIS6	PCSIS5	PCSIS4	PCSIS3	PCSIS2	PCSIS1	PCSIS0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	DOZE	MDIS	DIS_TXF	DIS_RXF	0	0	SMPL_PT		0	0	0	0	XSPI	FCPCS	PES	HALT
W					CLR_TXF	CLR_RXF										
Reset	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Figure 647. DSPI Module Configuration Register (DSPI*x*_MCR)

Table 700. DSPI*x*_MCR field descriptions

Field	Description
0 MSTR	Master/Slave Mode Select Configures the DSPI for either master mode or slave mode. 0 DSPI is in slave mode. 1 DSPI is in master mode.
1 CONT_SCKE	Continuous SCK Enable Enables the Serial Communication Clock (SCK) to run continuously. 0 Continuous SCK disabled. 1 Continuous SCK enabled.
2:3 DCONF	DSPI Configuration Selects among the different configurations of the DSPI. 00 SPI 01 Reserved 10 Reserved 11 Reserved
4 FRZ	Freeze Enables the DSPI transfers to be stopped on the next frame boundary when the device enters Debug mode. 0 Do not halt serial transfers in debug mode. 1 Halt serial transfers in debug mode.
5 MTFE	Modified Timing Format Enable Enables a modified transfer format to be used. 0 Modified SPI transfer format disabled. 1 Modified SPI transfer format enabled.
6 PCSSE	Peripheral Chip Select Strobe Enable Enables the PCS[5]/$\overline{\text{PCSS}}$ to operate as a PCS Strobe output signal. 0 PCS[5] is used as the Peripheral Chip Select[5] signal. 1 PCS[5]/$\overline{\text{PCSS}}$ is used as an active-low PCS Strobe signal.
7 ROOE	Receive FIFO Overflow Overwrite Enable In the RX FIFO overflow condition, configures the DSPI to ignore the incoming serial data or overwrite existing data. If the RX FIFO is full and new data is received, the data from the transfer, generating the overflow, is ignored or shifted into the shift register. 0 Incoming data is ignored. 1 Incoming data is shifted into the shift register.
8 PCSIS7	Peripheral Chip Select 7 Inactive State Determines the inactive state of PCS7 when DSPI is in Master Mode. This field has no effect when DSPI is in Slave Mode. The Slave Select input to DSPI in slave mode is always Active Low. 0 The inactive state of PCS7 is low. 1 The inactive state of PCS7 is high.
9 PCSIS6	Peripheral Chip Select 6 Inactive State Determines the inactive state of PCS6 when DSPI is in Master Mode. This field has no effect when DSPI is in Slave Mode. The Slave Select input to DSPI in slave mode is always Active Low. 0 The inactive state of PCS6 is low. 1 The inactive state of PCS6 is high.

Table 700. DSPI*x*_MCR field descriptions (continued)

Field	Description
10 PCSIS5	Peripheral Chip Select 5 Inactive State Determines the inactive state of PCS5 when DSPI is in Master Mode. This field has no effect when DSPI is in Slave Mode. The Slave Select input to DSPI in slave mode is always Active Low. 0 The inactive state of PCS5 is low. 1 The inactive state of PCS5 is high.
11 PCSIS4	Peripheral Chip Select 4 Inactive State Determines the inactive state of PCS4 when DSPI is in Master Mode. This field has no effect when DSPI is in Slave Mode. The Slave Select input to DSPI in slave mode is always Active Low. 0 The inactive state of PCS4 is low. 1 The inactive state of PCS4 is high.
12 PCSIS3	Peripheral Chip Select 3 Inactive State Determines the inactive state of PCS3 when DSPI is in Master Mode. This field has no effect when DSPI is in Slave Mode. The Slave Select input to DSPI in slave mode is always Active Low. 0 The inactive state of PCS3 is low. 1 The inactive state of PCS3 is high.
13 PCSIS2	Peripheral Chip Select 2 Inactive State Determines the inactive state of PCS2 when DSPI is in Master Mode. This field has no effect when DSPI is in Slave Mode. The Slave Select input to DSPI in slave mode is always Active Low. 0 The inactive state of PCS2 is low. 1 The inactive state of PCS2 is high.
14 PCSIS1	Peripheral Chip Select 1 Inactive State Determines the inactive state of PCS1 when DSPI is in Master Mode. This field has no effect when DSPI is in Slave Mode. The Slave Select input to DSPI in slave mode is always Active Low. 0 The inactive state of PCS1 is low. 1 The inactive state of PCS1 is high.
15 PCSIS0	Peripheral Chip Select 0 Inactive State Determines the inactive state of PCS0 when DSPI is in Master Mode. This field has no effect when DSPI is in Slave Mode. The Slave Select input to DSPI in slave mode is always Active Low. 0 The inactive state of PCS0 is low. 1 The inactive state of PCS0 is high.
16 DOZE	Doze Enable Provides support for an externally controlled Doze mode power-saving mechanism. 0 Doze mode has no effect on DSPI. 1 Doze mode disables DSPI.

Table 700. DSPI*x*_MCR field descriptions (continued)

Field	Description
17 MDIS	Module Disable Allows the clock to be stopped to the non-memory-mapped logic in the DSPI effectively putting the DSPI in a software controlled power-saving state. The reset value of the MDIS bit is parameterized, with a default reset value of 1. When DSPI is used in Slave Mode, it is recommended to leave this bit set to '0', since a slave does not have control over master transactions. 0 Enable DSPI clocks. 1 Allow external logic to disable DSPI clocks.
18 DIS_TXF	Disable Transmit FIFO When the TX FIFO is disabled, the transmit part of the DSPI (both TXFIFO and CMD FIFO) operates as a simplified double-buffered SPI. This bit can only be written when the MDIS bit is cleared. 0 Tx FIFO is enabled. 1 Tx FIFO is disabled.
19 DIS_RXF	Disable Receive FIFO When the RX FIFO is disabled, the receive part of the DSPI operates as a simplified double-buffered SPI. This bit can only be written when the MDIS bit is cleared. 0 Rx FIFO is enabled. 1 Rx FIFO is disabled.
20 CLR_TXF	Clear TX FIFO Flushes the TX FIFO and the CMD FIFO. Writing a '1' to CLR_TXF clears the TX FIFO and CMD FIFO Counters. The CLR_TXF bit is always read as zero. 0 Do not clear the Tx FIFO and CMD FIFO counters. 1 Clears the Tx FIFO and CMD FIFO counters.
21 CLR_RXF	Flushes the RX FIFO Writing a '1' to CLR_RXF clears the RX Counter. The CLR_RXF bit is always read as zero. 0 Do not clear the Rx FIFO counter. 1 Clear the Rx FIFO counter.
22:23 SMPL_PT	Sample Point Controls when the DSPI master samples SIN in Modified Transfer Format. This field is valid only when CPHA bit in CTAR is 0. 00 0 protocol clocks between SCK edge and SIN sample 01 1 protocol clock between SCK edge and SIN sample 10 2 protocol clocks between SCK edge and SIN sample 11 Reserved
24:27	Reserved This read-only bitfield is reserved and always has the value zero.

Table 700. DSPI*x*_MCR field descriptions (continued)

Field	Description
28 XSPI	Extended SPI Mode This bit enables usage of CTARE (Command and Transfer Attribute Register Extended) Registers. CTARE registers allow the user to send up to 32-bit SPI frames. Command Cycling is also enabled which allows the user to send multiple Data Frames using a single Command Frame. When MCR[DIS_TXF] is asserted, the Extended SPI Mode cannot be used to transmit SPI frames which are more than 16 bits in size. 0 Normal SPI Mode. Frame size can be up to 16 bits. Command Cycling is not available in this mode. 1 Extended SPI Mode. Up to 32-bit SPI Frames along with Command Cycling Enabled
29 FCPCS	Fast Continuous PCS Mode This bit enables the masking of “After SCK (t_{ASC})” and “PCS to SCK (t_{CSC})” delays when operating in Continuous PCS mode. This masking is not available if Continuous SCK mode is enabled. The individual delay masks are selected via bits MASC and MCSC of the PUSHR. The firmware should select appropriate masks when providing continuous frames via the PUSHR. 0 Normal or Slow Continuous PCS mode. Masking of delays is disabled. 1 Fast Continuous PCS mode. Delays masked via control bits in PUSHR.
30 PES	Parity Error Stop Controls SPI operation when a parity error is detected in a received SPI frame. 0 SPI frame transmission continues. 1 SPI frame transmission stops.
31 HALT	Halt Starts and stops DSPI transfers. 0 Start transfers. 1 Stop transfers.

43.2.9.2 DSPI Transfer Count Register (DSPI*x*_TCR)

The DSPI*x*_TCR contains a counter that indicates the number of SPI transfers made. The transfer counter is intended to assist in queue management. Do not write the TCR when the DSPI is in the Running state.

Offset: 0x0008 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	SPI_TCNT															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

Figure 648. DSPI Transfer Count Register (DSPI*x*_TCR)

Table 701. DSPI*x*_TCR field descriptions

Field	Description
0:15 SPI_TCNT	SPI Transfer Counter Counts the number of SPI transfers the DSPI makes. The SPI_TCNT field increments every time the last bit of a SPI frame is transmitted. A value written to SPI_TCNT presets the counter to that value. SPI_TCNT is reset to zero at the beginning of the frame when the CTCNT field is set in the executing SPI command. The Transfer Counter wraps around; incrementing the counter past 65535 resets the counter to zero.
16:31	Reserved This read-only bitfield is reserved and always has the value zero.

43.2.9.3 DSPI Clock and Transfer Attributes Register (In Master Mode) (DSPI*x*_CTAR*n*)

CTARs are used to define different transfer attributes. Do not write to the CTARs while the DSPI is in the Running state.

In master mode, the CTARs define combinations of transfer attributes such as frame size, clock phase and polarity, data bit ordering, baud rate, and various delays. In slave mode, a subset of the bitfields in CTAR0 and CTAR1 are used to set the slave transfer attributes.

When the DSPI is configured as an SPI master, the CTAS field in the command portion of the TX FIFO entry selects which of the CTARs is used. When the DSPI is configured as an SPI bus slave, the CTAR0 register is used.

Offset: 0x000C (CTAR0), 0x0010 (CTAR1), 0x0014 (CTAR2), 0x0018 (CTAR3), 0x001C (CTAR4), 0x0020 (CTAR5), 0x0024 (CTAR6), 0x0028 (CTAR7)

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	DBR	FMSZ				CPOL	CPHA	LSBFE	PCSSCK		PASC		PDT		PBR	
Reset	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	CSSCK				ASC				DT				BR			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 649. DSPI Clock and Transfer Attributes Register (In Master Mode) (DSPI*x*_CTAR*n*)

Table 702. DSPI*x*_CTAR*n* field descriptions

Field	Description
0 DBR	Double Baud Rate Doubles the effective baud rate of the Serial Communications Clock (SCK). This field is used only in master mode. It effectively halves the Baud Rate division ratio, supporting faster frequencies, and odd division ratios for the Serial Communications Clock (SCK). When the DBR bit is set, the duty cycle of the Serial Communications Clock (SCK) depends on the value in the Baud Rate Prescaler and the Clock Phase bit as listed in *Table 703*. See the BR field description for details on how to compute the baud rate. Refer to *Table 703* for DSPI SCK duty cycle values. 0 The baud rate is computed normally with a 50/50 duty cycle. 1 The baud rate is doubled with the duty cycle depending on the Baud Rate Prescaler.
1:4 FMSZ	Frame Size The number of bits transferred per frame is equal to the FMSZ field value plus 1. The minimum valid FMSZ field value is 3.There is a constraint on the minimum allowable Frame size, which depends on the Register Read/Write clock to the Protocol Clock Ratio. The minimum Frame size (FMSZ + 1) is decided by the following equation. Upper Ceiling must be applied for non-integer values. Min. Frame Size = ((4 x f_p) + (3 x f_r)) / (n x f_r) f_p = Protocol Clock Frequency f_r = Register Read/Write Clock Frequency n = Multiple of protocol clock required to get Baud Clock - Given by (PBR x BR) / (1 + DBR) However, the minimum frame size can never be less than 4. There is no constraint on the maximum programmable frame size. Note that 'f_p' can be equal to, less than or greater than 'f_r' purely based on user requirement.

Table 702. DSPI*x*_CTAR*n* field descriptions (continued)

Field	Description
5 CPOL	Clock Polarity Selects the inactive state of the Serial Communications Clock (SCK). This bit is used in both master and slave mode. For successful communication between serial devices, the devices must have identical clock polarities. When the Continuous Selection Format is selected, switching between clock polarities without stopping the DSPI can cause errors in the transfer due to the peripheral device interpreting the switch of clock polarity as a valid clock edge. 0 The inactive state value of SCK is low. 1 The inactive state value of SCK is high.
6 CPHA	Clock Phase Selects which edge of SCK causes data to change and which edge causes data to be captured. This bit is used in both master and slave mode. For successful communication between serial devices, the devices must have identical clock phase settings. In Continuous SCK mode, the bit value is ignored and the transfers are done as if the CPHA bit is set to '1'. 0 Data is captured on the leading edge of SCK and changed on the following edge. 1 Data is changed on the leading edge of SCK and captured on the following edge.
7 LSBFE	LSB First Specifies whether the LSB or MSB of the frame is transferred first. 0 Data is transferred MSB first. 1 Data is transferred LSB first.
8:9 PCSSCK	PCS to SCK Delay Prescaler Selects the prescaler value for the delay between assertion of PCS and the first edge of the SCK. See the CSSCK field description for information on how to compute the PCS to SCK Delay. Refer to *Section 43.4.3.2: PCS to SCK delay (t_{CSC})* for more details. 00 PCS to SCK Prescaler value is 1. 01 PCS to SCK Prescaler value is 3. 10 PCS to SCK Prescaler value is 5. 11 PCS to SCK Prescaler value is 7.
10:11 PASC	After SCK Delay Prescaler Selects the prescaler value for the delay between the last edge of SCK and the negation of PCS. See the ASC field description for information on how to compute the After SCK Delay. 00 Delay after Transfer Prescaler value is 1. 01 Delay after Transfer Prescaler value is 3. 10 Delay after Transfer Prescaler value is 5. 11 Delay after Transfer Prescaler value is 7.
12:13 PDT	Delay after Transfer Prescaler Selects the prescaler value for the delay between the negation of the PCS signal at the end of a frame and the assertion of PCS at the beginning of the next frame. The PDT field is only used in master mode. See the DT field description for details on how to compute the Delay after Transfer. Refer to *Section 43.4.3.4: Delay after transfer (t_{DT})* for more details. 00 Delay after Transfer Prescaler value is 1. 01 Delay after Transfer Prescaler value is 3. 10 Delay after Transfer Prescaler value is 5. 11 Delay after Transfer Prescaler value is 7.

Table 702. DSPI*x*_CTAR*n* field descriptions (continued)

Field	Description
14:15 PBR	Baud Rate Prescaler Selects the prescaler value for the baud rate. This field is used only in master mode. The baud rate is the frequency of the SCK. The protocol clock is divided by the prescaler value before the baud rate selection takes place. See the BR field description for details on how to compute the baud rate. 00 Baud Rate Prescaler value is 2. 01 Baud Rate Prescaler value is 3. 10 Baud Rate Prescaler value is 5. 11 Baud Rate Prescaler value is 7.
16:19 CSSCK	PCS to SCK Delay Scaler Selects the scaler value for the PCS to SCK delay. This field is used only in master mode. The PCS to SCK Delay is the delay between the assertion of PCS and the first edge of the SCK. The delay is a multiple of the protocol clock period, and it is computed according to the following equation: $t_{CSC} = (1/f_p) \times PCSSCK \times CSSCK$ 0000 Delay scaler value = 2 0001 Delay scaler value = 4 0010 Delay scaler value = 8 0011 Delay scaler value = 16 0100 Delay scaler value = 32 0101 Delay scaler value = 64 0110 Delay scaler value = 128 0111 Delay scaler value = 256 1000 Delay scaler value = 512 1001 Delay scaler value = 1024 1010 Delay scaler value = 2048 1011 Delay scaler value = 4096 1100 Delay scaler value = 8192 1101 Delay scaler value = 16384 1110 Delay scaler value = 32768 1111 Delay scaler value = 65536 Refer to *Section 43.4.3.2: PCS to SCK delay (t_{CSC})* for more details.
20:23 ASC	After SCK Delay Scaler Selects the scaler value for the After SCK Delay. This field is used only in master mode. The After SCK Delay is the delay between the last edge of SCK and the negation of PCS. The delay is a multiple of the protocol clock period, and it is computed according to the following equation: $t_{ASC} = (1/f_p) \times PASC \times ASC$ See Delay Scaler Encoding table in CTAR*n*[CSSCK] bit field description for scaler values. Refer to *Section 43.4.3.3: After SCK delay (t_{ASC})* for more details.

Table 702. DSPI*x*_CTAR*n* field descriptions (continued)

Field	Description
24:27 DT	Delay After Transfer Scaler Selects the Delay after Transfer Scaler. This field is used only in master mode. The Delay after Transfer is the time between the negation of the PCS signal at the end of a frame and the assertion of PCS at the beginning of the next frame. In the Continuous Serial Communications Clock operation, the DT value is fixed to one SCK clock period, The Delay after Transfer is a multiple of the protocol clock period, and it is computed according to the following equation: $t_{DT} = (1/f_p) \times PDT \times DT$ See Delay Scaler Encoding table in CTAR*n*[CSSCK] bit field description for scaler values.
28:31 BR	Baud Rate Scaler Selects the scaler value for the baud rate. This field is used only in master mode. The prescaled protocol clock is divided by the Baud Rate Scaler to generate the frequency of the SCK. The baud rate is computed according to the following equation: SCK baud rate = $(f_p/PBR) \times [(1+DBR)/BR]$ 0000 Baud rate scaler = 2 0001 Baud rate scaler = 4 0010 Baud rate scaler = 6 0011 Baud rate scaler = 8 0100 Baud rate scaler = 16 0101 Baud rate scaler = 32 0110 Baud rate scaler = 64 0111 Baud rate scaler = 128 1000 Baud rate scaler = 256 1001 Baud rate scaler = 512 1010 Baud rate scaler = 1024 1011 Baud rate scaler = 2048 1100 Baud rate scaler = 4096 1101 Baud rate scaler = 8192 1110 Baud rate scaler = 16384 1111 Baud rate scaler = 32768

Table 703. DSPI SCK duty cycle

DBR	CPHA	PBR	SCK duty cycle
0	any	any	50/50
1	0	00	50/50
1	0	01	33/66
1	0	10	40/60
1	0	11	43/57
1	1	00	50/50
1	1	01	66/33
1	1	10	60/40
1	1	11	57/43

43.2.9.4 DSPI Clock and Transfer Attributes Register (In Slave Mode) (DSPI*x*_CTAR0_SLAVE)

When the DSPI is configured as an SPI bus slave, the CTAR0 register is used.

Offset: 0x000C Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	FMSZ					CPOL	CPHA	PE	PP	0	0	0	0	0	0	0
W																
Reset	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0

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

Figure 650. DSPI Clock and Transfer Attributes Register (In Slave Mode) (DSPI*x*_CTAR0_SLAVE)

Table 704. DSPI*x*_CTAR0_SLAVE field descriptions

Field	Description
0:4 FMSZ	Frame Size The number of bits transferred per frame is equal to the concatenated field {CTAR0_SLAVE[FMSZ[5]], CTAR0_SLAVE[FMSZ]} value plus 1. The minimum valid FMSZ field value is 3 (when FMSZ6 is 0).
5 CPOL	Clock Polarity Selects the inactive state of the Serial Communications Clock (SCK). 0 The inactive state value of SCK is low. 1 The inactive state value of SCK is high.
6 CPHA	Clock Phase Selects which edge of SCK causes data to change and which edge causes data to be captured. This bit is used in both master and slave mode. For successful communication between serial devices, the devices must have identical clock phase settings. In Continuous SCK mode, the bit value is ignored and the transfers are done as the CPHA bit is set to ‘1’. 0 Data is captured on the leading edge of SCK and changed on the following edge. 1 Data is changed on the leading edge of SCK and captured on the following edge.
7 PE	Parity Enable Enables parity bit transmission and reception for the frame. 0 No parity bit included/checked. 1 Parity bit is transmitted instead of last data bit in frame, parity checked for received frame.
8 PP	Parity Polarity Controls polarity of the parity bit transmitted and checked. 0 Even Parity: the number of 1 bits in the transmitted frame is even. The SR[SPEF] bit is set if the number of 1 bits is odd in the received frame. 1 Odd Parity: the number of 1 bits in the transmitted frame is odd. The SR[SPEF] bit is set if the number of 1 bits is even in the received frame.
9:31	Reserved This read-only bitfield is reserved and always has the value zero.

43.2.9.5 DSPI Clock and Transfer Attributes Register (In Slave Mode) (DSPI*x*_CTAR1_SLAVE)

Offset: 0x0010 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	FMSZ					CPOL	CPHA	PE	PP	0	0	0	0	0	0	0
W																
Reset	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0

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

Figure 651. DSPI Clock and Transfer Attributes Register (In Slave Mode) (DSPI*x*_CTAR1_SLAVE)

Table 705. DSPI*x*_CTAR1_SLAVE field descriptions

Field	Description
0:4 FMSZ	Frame Size The number of bits transferred per frame is equal to the concatenated field {CTAR1_SLAVE[FMSZ[5]], CTAR1_SLAVE[FMSZ]} value plus 1. The minimum valid FMSZ field value is 3 (when FMSZ6 is 0).
5 CPOL	Clock Polarity Selects the inactive state of the Serial Communications Clock (SCK). 0 The inactive state value of SCK is low. 1 The inactive state value of SCK is high.
6 CPHA	Clock Phase Selects which edge of SCK causes data to change and which edge causes data to be captured. This bit is used in both master and slave mode. For successful communication between serial devices, the devices must have identical clock phase settings. In Continuous SCK mode, the bit value is ignored and the transfers are done as the CPHA bit is set to ‘1’. 0 Data is captured on the leading edge of SCK and changed on the following edge. 1 Data is changed on the leading edge of SCK and captured on the following edge.
7 PE	Parity Enable Enables parity bit transmission and reception for the frame. 0 No parity bit included/checked. 1 Parity bit is transmitted instead of last data bit in frame, parity checked for received frame.
8 PP	Parity Polarity Controls polarity of the parity bit transmitted and checked. 0 Even Parity: the number of 1 bits in the transmitted frame is even. The SR[SPEF] bit is set if the number of 1 bits is odd in the received frame. 1 Odd Parity: the number of 1 bits in the transmitted frame is odd. The SR[SPEF] bit is set if the number of 1 bits is even in the received frame.
9:31	Reserved This read-only bitfield is reserved and always has the value zero.

43.2.9.6 DSPI Status Register (DSPI*x*_SR)

SR contains status and flag bits. The bits reflect the status of the DSPI and indicate the occurrence of events that can generate interrupt or DMA requests. Software can clear most of the flag bits in the SR by writing a '1' to them. Writing a 0 to a flag bit has no effect. This register may not be writable in module disable mode due to the use of power saving mechanisms.

Offset: 0x002C Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	TCF	TXRXS	0	EOQF	TFUF	0	TFFF	BSYF	CMDTCF	0	SPEF	0	RFOF	TFIWF	RFDF	CMDFFF
W	w1c	w1c		w1c	w1c		w1c		w1c		w1c		w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TXCTR				TXNXTPTR				RXCTR				POPNXTPTR			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 652. DSPI Status Register (DSPI*x*_SR)

Table 706. DSPI*x*_SR field descriptions

Field	Description
0 TCF	Transfer Complete Flag Indicates that all bits in a frame have been shifted out. TCF remains set until it is cleared by writing a '1' to it. 0 Transfer not complete. 1 Transfer complete.
1 TXRXS	TX and RX Status Reflects the run status of the DSPI. 0 Transmit and receive operations are disabled (DSPI is in stopped state). 1 Transmit and receive operations are enabled (DSPI is in running state).
2	Reserved Write of any value has no effect; read value is always 0.
3 EOQF	End of Queue Flag Indicates that the last entry in a queue has been transmitted when the DSPI is in master mode. The EOQF bit is set when the TX FIFO entry has the EOQ bit set in the command halfword and the end of the transfer is reached. The EOQF bit remains set until cleared by writing a '1' to it. When the EOQF bit is set, the TXRXS bit is automatically cleared. 0 EOQ is not set in the executing command. 1 EOQ is set in the executing SPI command.

Table 706. DSPI*x*_SR field descriptions (continued)

Field	Description
4 TFUF	Transmit FIFO Underflow Flag Indicates an underflow condition in the TX FIFO. The transmit underflow condition is detected only for DSPI blocks operating in slave mode and SPI configuration. TFUF is set when the TX FIFO of a DSPI operating in SPI slave mode is empty and an external SPI master initiates a transfer. The TFUF bit remains set until cleared by writing '1' to it. 0 No Tx FIFO underflow. 1 Tx FIFO underflow has occurred.
5	Reserved Write of any value has no effect; read value is always 0.
6 TFFF	Transmit FIFO Fill Flag Provides a method for the DSPI to request more entries to be added to the TX FIFO. The TFFF bit is set while the TX FIFO is not full. The TFFF bit can be cleared by writing '1' to it or by acknowledgement from the DMA controller to the TX FIFO full request. 0 Tx FIFO is full. 1 Tx FIFO is not full.
7 BSYF	Busy Flag This bit is valid only when DSPI_MCR[XSPI] is enabled. Indicates that the current Command Frame is being used for transmitting multiple data frames. This bit is not set for the last Data Frame of a Cyclic command Transfer or when DSPI_CTARE[DTCP] = 1. Refer to *Section 43.4.2.5: Command First In First Out (CMD FIFO) buffering mechanism* for more details. 0 No Cyclic Command Transfer in Progress. 1 Cyclic Command Transfer is in progress. Current Data Frame is not the last data frame for current cyclic command transfer.
8 CMDTCF	Command Transfer Complete Flag Indicates that the last Data frame for the current Cyclic Command has been transmitted. Hence this bit is set only for the last Data Frame of a Cyclic Command Transfer or when DSPI_CTARE[DTCP] = 1. CMDTCF remains set until it is cleared by writing a '1' to it. 0 Data Transfer by current Command not complete. 1 Data Transfer by current Command complete.
9	Reserved Write of any value has no effect; read value is always 0.
10 SPEF	SPI Parity Error Flag Indicates that a SPI frame with parity error had been received. The bit remains set until it is cleared by writing a '1' to it. 0 No parity error. 1 Parity error has occurred.
11	Reserved Write of any value has no effect; read value is always 0.
12 RFOF	Receive FIFO Overflow Flag Indicates an overflow condition in the RX FIFO. The bit is set when the RX FIFO and shift register are full and a transfer is initiated. The bit remains set until it is cleared by writing a '1' to it. 0 No Rx FIFO overflow. 1 Rx FIFO overflow has occurred.

Table 706. DSPI*x*_SR field descriptions (continued)

Field	Description
13 TFIWF	Transmit FIFO Invalid Write Flag Indicates Data Write on TX FIFO while CMD FIFO is empty. Without a Command, the Data entries present in TXFIFO are invalid. 0 No Invalid Data present in TX FIFO 1 Invalid Data present in TX FIFO since CMD FIFO is empty
14 RFDF	Receive FIFO Drain Flag Provides a method for the DSPI to request that entries be removed from the RX FIFO. The bit is set while the RX FIFO is not empty. The RFDF bit can be cleared by writing 1 to it or by acknowledgement from the DMA controller when the RX FIFO is empty. 0 Rx FIFO is empty. 1 Rx FIFO is not empty.
15 CMDFFF	Command FIFO Fill Flag Provides a method for the DSPI to request more entries to be added to the CMD FIFO. The CMDFFF bit is set while the CMD FIFO is not full.The CMDFFF is cleared by writing a ‘1’ to it or by acknowledgement from the DMA controller to the CMD FIFO full request. 0 CMD FIFO is full. 1 CMD FIFO is not full.
16:19 TXCTR	TX FIFO Counter Indicates the number of valid entries in the TX FIFO. The TXCTR is incremented every time the data part of PUSHR is written. The TXCTR is decremented every time the SPI data is transferred to the shift register.
20:23 TXNXTPTR	Transmit Next Pointer Indicates which TX FIFO Entry is transmitted during the next transfer. The TXNXTPTR field is updated every time SPI data is transferred from the TX FIFO to the shift register.
24:27 RXCTR	RX FIFO Counter Indicates the number of entries in the RX FIFO. The RXCTR is decremented every time the POPR is read. The RXCTR is incremented every time data is transferred from the shift register to the RX FIFO.
28:31 POPNXTPTR	Pop Next Pointer Contains a pointer to the RX FIFO entry to be returned when the POPR is read. The POPNXTPTR is updated when the POPR is read.

43.2.9.7 DSPI DMA/Interrupt Request Select and Enable Register (DSPI*x*_RSER)

The DSPI*x*_RSER controls DMA and interrupt requests. Do not write to the RSER while the DSPI is in the Running state.

Offset: 0x0030 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	TCF_RE	CMDFFF_RE	0	EOQF_RE	TFUF_RE	0	TFFF_RE	TFFF_DIRS	CMDTCF_RE	0	SPEF_RE	0	RFOF_RE	TFIWF_RE	RFDF_RE	RFDF_DIRS
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	CMDFFF_DIRS	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 653. DSPI DMA/Interrupt Request Select and Enable Register (DSPI*x*_RSER)

Table 707. DSPI*x*_RSER field descriptions

Field	Description
0 TCF_RE	Transmission Complete Request Enable Enables TCF flag in the SR to generate an interrupt request. 0 TCF interrupt requests are disabled. 1 TCF interrupt requests are enabled.
1 CMDFFF_RE	Command FIFO Fill Flag Request Enable Enables the CMDFFF flag in the SR to generate a request. The CMDFFF_DIRS bit selects between generating an interrupt request or a DMA request. 0 CMDFFF interrupts or DMA requests are disabled. 1 CMDFFF interrupts or DMA requests are enabled.
2	Reserved Write of any value has no effect; read value is always 0.
3 EOQF_RE	DSPI Finished Request Enable Enables the EOQF flag in the SR to generate an interrupt request. 0 EOQF interrupt requests are disabled. 1 EOQF interrupt requests are enabled.
4 TFUF_RE	Transmit FIFO Underflow Request Enable Enables the TFUF flag in the SR to generate an interrupt request. 0 TFUF interrupt requests are disabled. 1 TFUF interrupt requests are enabled.
5	Reserved Write of any value has no effect; read value is always 0.

Table 707. DSPI*x*_RSER field descriptions (continued)

Field	Description
6 TFFF_RE	Transmit FIFO Fill Request Enable Enables the TFFF flag in the SR to generate a request. The TFFF_DIRS bit selects between generating an interrupt request or a DMA request. 0 TFFF interrupts or DMA requests are disabled. 1 TFFF interrupts or DMA requests are enabled.
7 TFFF_DIRS	Transmit FIFO Fill DMA or Interrupt Request Select Selects between generating a DMA request or an interrupt request. When the TFFF flag bit in the SR is set and the TFFF_RE bit in the RSER is set, this bit selects between generating an interrupt request or a DMA request. 0 TFFF flag generates interrupt requests. 1 TFFF flag generates DMA requests.
8 CMDTCF_RE	Command Transmission Complete Request Enable The CMDTCF_RE bit enables CMDTCF flag in the SR to generate an interrupt request. 0 CMDTCF interrupt requests are disabled. 1 CMDTCF interrupt requests are enabled.
9	Reserved Write of any value has no effect; read value is always 0.
10 SPEF_RE	SPI Parity Error Request Enable Enables the SPEF flag in the SR to generate an interrupt request. 0 SPEF interrupt requests are disabled. 1 SPEF interrupt requests are enabled.
11	Reserved Write of any value has no effect; read value is always 0.
12 RFOF_RE	Receive FIFO Overflow Request Enable Enables the RFOF flag in the SR to generate an interrupt request. 0 RFOF interrupt requests are disabled. 1 RFOF interrupt requests are enabled.
13 TFIWF_RE	Transmit FIFO Invalid Write Request Enable Enables the TFIWF flag in the SR to generate an interrupt request. 0 TFIWF interrupt requests are disabled. 1 TFIWF interrupt requests are enabled.
14 RFDF_RE	Receive FIFO Drain Request Enable Enables the RFDF flag in the SR to generate a request. The RFDF_DIRS bit selects between generating an interrupt request or a DMA request. 0 RFDF interrupt or DMA requests are disabled 1 RFDF interrupt or DMA requests are enabled
15 RFDF_DIRS	Receive FIFO Drain DMA or Interrupt Request Select Selects between generating a DMA request or an interrupt request. When the RFDF flag bit in the SR is set, and the RFDF_RE bit in the RSER is set, the RFDF_DIRS bit selects between generating an interrupt request or a DMA request. 0 Interrupt request. 1 DMA request.

Table 707. DSPI*x*_RSER field descriptions (continued)

Field	Description
16 CMDFFF_DIRS	Command FIFO FIll DMA or Interrupt Request Select Selects between generating a DMA request or an interrupt request. When the CMDFFF flag bit in the SR is set, and the CMDFFF_RE bit in the RSER is set, the CMDFFF_DIRS bit selects between generating an interrupt request or a DMA request. 0 Interrupt request. 1 DMA request.
17:31	Reserved This read-only bitfield is reserved and always has the value zero.

43.2.9.8 DSPI PUSH FIFO Register In Master Mode (DSPI*x*_PUSHR)

PUSHR provides the means to write to the TX FIFO and CMD FIFO. Data written to this register is transferred to the TX FIFO and/or CMD FIFO. Eight- or sixteen-bit write accesses to the Data Field of PUSHR transfers the 16-bit Data field of PUSHR to the TX FIFO. Write accesses to the Command Field of PUSHR transfers the 16-bit Command Field of PUSHR to the CMD FIFO. The register structure is different in master and slave modes. In master mode, the register provides 16-bit command to the CMD FIFO and 16-bit data to the TX FIFO. In slave mode, CMD FIFO is unused and hence the 16-bit Command Field of PUSHR is reserved.

Unless Extended SPI Mode is being used, the TX FIFO and CMD FIFO must be filled simultaneously. In other words, write accesses must be given to both Data and Command fields of PUSHR for every PUSHR operation performed when MCR[XSPI] is 0. When Extended SPI Mode is disabled, since both TX FIFO and CMD FIFO are written to and read from simultaneously, they can be thought of as a single 32-bit FIFO.

When Extended SPI Mode is enabled, TX FIFO and CMD FIFO can be written independently. A PUSHR Read Operation returns the topmost TX FIFO and CMD FIFO entries concatenated.

When DSPI Module is disabled, any writes to this register will not update the FIFO. Hence any reads performed during Module disable mode will return the last PUSHR write performed when Module was enabled.

Offset: 0x0034 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	CONT	CTAS			EOQ	CTCNT	PE_MASC	PP_MCSC	PCS7	PCS6	PCS5	PCS4	PCS3	PCS2	PCS1	PCS0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	TXDATA															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 654. DSPI PUSH FIFO Register In Master Mode (DSPI*x*_PUSHR)

Table 708. DSPI*x*_PUSHR field descriptions

Field	Description
0 CONT	Continuous Peripheral Chip Select Enable Selects a Continuous Selection Format. The bit is used in SPI master mode. The bit enables the selected PCS signals to remain asserted between transfers. PUSHR[EOQ] and PUSHR[CONT] bits cannot be asserted together in any Command Frame, since EOQ signifies the last frame in a queue and the Last frame in a Continuous Transfer must have PUSHR[CONT] as 0. 0 Return PCSn signals to their inactive state between transfers. 1 Keep PCSn signals asserted between transfers.
1:3 CTAS	Clock and Transfer Attributes Select Selects which CTAR (and corresponding CTARE register) to use in master mode to specify the transfer attributes for the associated SPI frame. In SPI slave mode, CTAR0 is used. See *Chapter 7: Device configuration* to determine how many CTARs this device has. You should not program a value in this field for a register that is not present. 000 CTAR0/CTARE0 001 CTAR1/CTARE1 010 CTAR2/CTARE2 011 CTAR3/CTARE3 100 CTAR4/CTARE4 101 CTAR5/CTARE5 110 CTAR6/CTARE6 111 CTAR7/CTARE7
4 EOQ	End Of Queue Host software uses this bit to signal to the DSPI that the current SPI transfer is the last in a queue. At the end of the transfer, the EOQF bit in the SR is set. 0 The SPI data is not the last data to transfer. 1 The SPI data is the last data to transfer.
5 CTCNT	Clear Transfer Counter Clears the SPI_TCNT field in the TCR. The SPI_TCNT field is cleared before the DSPI starts transmitting the current SPI frame. Hence the current SPI frame will cause this counter to increment by 1. 0 Do not clear the TCR[SPI_TCNT] field. 1 Clear the TCR[SPI_TCNT] field.
6 PE_MASC	Parity Enable or Mask t_{ASC} delay in the current frame PE—This bit enables parity bit transmission and parity reception check for the SPI frame. MASC—The current frame has the “after SCK” delay masked if this bit is asserted. See *Section 43.4.4.6: Fast Continuous Selection Format* for more details. **Note:** This bit is used as Mask t_{ASC} in the Fast Continuous PCS mode when MCR[FCPCS] is set. 0 PE - No parity bit included/checked. MASC - TASC delay is not masked and the current frame has the after SCK delay. 1 PE - Parity bit is transmitted instead of the last data bit in the frame; parity is checked for the received frame. MASC - TASC delay is masked in the current frame.

Table 708. DSPI*x*_PUSHR field descriptions (continued)

Field	Description
7 PP_MCSC	Parity Polarity or Mask t_{CSC} delay in the next frame PP—It controls the polarity of the parity bit transmitted and checked. MCSC—The next frame has the “PCS to SCK” delay masked if this bit is asserted. See *Section 43.4.4.6: Fast Continuous Selection Format* for more details. **Note:** This bit is used as Mask t_{CSC} in the Fast Continuous PCS mode when MCR[FCPCS] is set. 0 PP - Even Parity: the number of 1 bits in the transmitted frame is even. The SR[SPEF] bit is set if the number of 1 bits is odd in the received frame. MCSC - TCSC delay is not masked and the next frame has the PCS to SCK delay. 1 PP - Odd Parity: the number of 1 bits in the transmitted frame is odd. The SR[SPEF] bit is set if the number of 1 bits is even in the received frame. MCSC - TCSC delay is masked in the next frame.
8 PCS7	Peripheral Chip Select Selects which PCS signals are to be asserted for the transfer. Refer to *Chapter 7: Device configuration* for the number of PCS signals used in this MCU. 0 Negate the PCS[7] signal. 1 Assert the PCS[7] signal.
9 PCS6	Peripheral Chip Select Selects which PCS signals are to be asserted for the transfer. Refer to *Chapter 7: Device configuration* for the number of PCS signals used in this MCU. 0 Negate the PCS[6] signal. 1 Assert the PCS[6] signal.
10 PCS5	Peripheral Chip Select Selects which PCS signals are to be asserted for the transfer. Refer to *Chapter 7: Device configuration* for the number of PCS signals used in this MCU. 0 Negate the PCS[5] signal. 1 Assert the PCS[5] signal.
11 PCS4	Peripheral Chip Select Selects which PCS signals are to be asserted for the transfer. Refer to *Chapter 7: Device configuration* for the number of PCS signals used in this MCU. 0 Negate the PCS[4] signal. 1 Assert the PCS[4] signal.
12 PCS3	Peripheral Chip Select Selects which PCS signals are to be asserted for the transfer. Refer to *Chapter 7: Device configuration* for the number of PCS signals used in this MCU. 0 Negate the PCS[3] signal. 1 Assert the PCS[3] signal.
13 PCS2	Peripheral Chip Select Selects which PCS signals are to be asserted for the transfer. Refer to *Chapter 7: Device configuration* for the number of PCS signals used in this MCU. 0 Negate the PCS[2] signal. 1 Assert the PCS[2] signal.
14 PCS1	Peripheral Chip Select Selects which PCS signals are to be asserted for the transfer. Refer to *Chapter 7: Device configuration* for the number of PCS signals used in this MCU. 0 Negate the PCS[1] signal. 1 Assert the PCS[1] signal.

Table 708. DSPI*x*_PUSHR field descriptions (continued)

Field	Description
15 PCS0	Peripheral Chip Select Selects which PCS signals are to be asserted for the transfer. Refer to *Chapter 7: Device configuration* for the number of PCS signals used in this MCU. 0 Negate the PCS[0] signal. 1 Assert the PCS[0] signal.
16:31 TXDATA	Transmit Data Holds SPI data to be transferred according to the associated SPI command.

43.2.9.9 DSPI PUSH FIFO Register In Slave Mode (DSPI*x*_PUSHR_SLAVE)

PUSHR provides the means to write to the TX FIFO. Data written to this register is transferred to the TX FIFO. Eight or 16-bit write accesses to the Data Field of PUSHR transfers the 16-bit Data Field of PUSHR to the TX FIFO. The register structure is different in master and slave modes. In master mode, the register provides 16-bit command to the CMD FIFO and 16-bit data to the TX FIFO. In slave mode, CMD FIFO is unused and hence the 16-bit Command Field of PUSHR is reserved.

In Slave mode, up to 32-bit SPI frames may be queued for transmission/reception by enabling MCR[XSPI] mode.

Offset: 0x0034 Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TXDATA															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 655. DSPI PUSH FIFO Register In Slave Mode (DSPI*x*_PUSHR_SLAVE)

Table 709. DSPI*x*_PUSHR_SLAVE field descriptions

Field	Description
0:15	Reserved This read-only bitfield is reserved and always has the value zero.
16:31 TXDATA	Transmit Data Holds SPI data to be transferred.

43.2.9.10 DSPI POP FIFO Register (DSPI*x*_POPR)

POPR is used to read the RX FIFO. Eight- or sixteen-bit read accesses to the POPR have the same effect on the RX FIFO as 32-bit read accesses. A write to this register will generate a Transfer Error.

Offset: 0x0038 Access: User read-only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	RXDATA															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	RXDATA															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 656. DSPI POP FIFO Register (DSPI*x*_POPR)

Table 710. DSPI*x*_POPR field descriptions

Field	Description
0:31 RXDATA	Received Data Contains the SPI data from the RX FIFO entry to which the Pop Next Data Pointer points.

43.2.9.11 DSPI Transmit FIFO Registers (DSPI*x*_TXFRn)

TXFRn provide visibility into the TX FIFO for debugging purposes. Each register is an entry in the TX FIFO. The registers are read-only and cannot be modified. Reading the TXFRn registers does not alter the state of the TX FIFO. Reading DSPI*x*_TXFRn registers is invalid for a DSPI Master when being used in the Extended SPI (XSPI) Mode.

Offset: 0x003C +n*0x4 (n=0 to 3) Access: User read-only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	TXCMD_TXDATA															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TXDATA															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 657. DSPI Transmit FIFO Registers (DSPI*x*_TXFRn)

Table 711. DSPI*x*_TXFRn field descriptions

Field	Description
0:15 TXCMD_TXDATA	Transmit Command or Transmit Data In master mode, the TXCMD field contains the command that sets the transfer attributes for the SPI data. In slave mode, the TXDATA contains 16 MSB bits of the SPI data to be shifted out.
16:31 TXDATA	Transmit Data Contains the SPI data to be shifted out.

43.2.9.12 DSPI Receive FIFO Registers (DSPI*x*_RXFRn)

RXFRn provide visibility into the RX FIFO for debugging purposes. Each register is an entry in the RX FIFO. The RXFRs are read-only. Reading the RXFRn registers does not alter the state of the RX FIFO. The field MCR[MDIS] must be 0 when RXFR is read.

Offset: 0x007C +n*0x4 (n=0 to 3) Access: User read-only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	RXDATA															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	RXDATA															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 658. DSPI Receive FIFO Registers (DSPI*x*_RXFRn)

Table 712. DSPI*x*_RXFRn field descriptions

Field	Description
0:31 RXDATA	Receive Data Contains the received SPI data.

43.2.9.13 DSPI Clock and Transfer Attributes Register Extended (DSPI*x*_CTARE*n*)

CTARE registers are used to define the extended transfer attributes for an SPI frame. These registers are valid only when DSPI*x*_MCR[XSPI] is set.

When the DSPI is configured as an SPI master, the CTAS field in CMD FIFO entry selects which of the CTARE registers is used. When the DSPI is configured as an SPI bus slave, the CTARE0 register is used.

Offset: 0x011C +n*0x4 (n=0 to 7) Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	FMSZE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	DTCP										
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Figure 659. DSPI Clock and Transfer Attributes Register Extended (DSPI*x*_CTARE*n*)

Table 713. DSPI*x*_CTARE*n* field descriptions

Field	Description
15 FMSZE	Frame Size Extended This field is valid only when DSPI*x*_MCR[XSPI] is set. This field concatenated with DSPI*x*_CTAR[FMSZ] defines the frame size of the SPI frames to be transmitted. Effective frame size is the concatenation of {DSPI*x*_CTAR[FMSZ], DSPI*x*_CTARE[FMSZE]} plus 1. 0 Default Mode. Up to 16-bit SPI frames can be transferred. 1 Up to 32-bit SPI frames can be transferred. Each frame transfer will be a result of 2 TX FIFO Pops.
21:31 DTCP	Data Transfer Count Preload This field is valid only when DSPI*x*_MCR[XSPI] is set. This field defines the number of data frames (whose size is defined by CTARE[FMSZE] and CTAR[FMSZ]) to be transmitted using the Command frame that selected this DSPI*x*_CTARE register. The value 0 is reserved and should not be written in this field. The default value of this field is ‘1’.

43.2.9.14 DSPI Status Register Extended (DSPI*x*_SREX)

The DSPI*x*_SREX contains status fields. The fields reflect the status of the DSPI and indicate the occurrence of events. This register is not writable.

Offset: 0x013C Access: User read-only

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	CMDCTR				CMDNXTPTR			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 660. DSPI Status Register Extended (DSPI*x*_SREX)

Table 714. DSPI*x*_SREX field descriptions

Field	Description
24:27 CMDCTR	CMD FIFO Counter Indicates the number of entries in the CMD FIFO. The CMDCTR is incremented every time the command part of PUSHR is written. The CMDCTR is decremented every time a SPI command is executed (all data frames due to current command frame have been transmitted).
28:31 CMDNXTPTR	Command Next Pointer Indicates which CMD FIFO Entry is used during the next transfer. The CMDNXTPTR field is updated every time SPI data due to current command have been transmitted.

43.3 Register classification for safety

The registers of this module are classified as per safety requirements as follows:

- **Safety Critical:** There are no safety critical registers in this module.
- **Safety Relevant:** The register bits/fields in this category are safety relevant as any random error in these bits will lead to unwanted interference to CPU. The registers containing these fields must be protected by periodic CRC or equivalent check implemented at chip level. The following register bits/fields are classified in this category:
  - DSPI Module Configuration Register (DSPI*x*_MCR)
  - DSPI Transfer Count Register (DSPI*x*_TCR)
  - DSPI Clock and Transfer Attributes Register (In Master Mode) (DSPI*x*_CTAR*n*)
  - DSPI Clock and Transfer Attributes Extended Register (In Master Mode) (DSPI*x*_CTARE*n*)
  - DSPI DMA/Interrupt Request Select and Enable Register (DSPI*x*_RSER)
  - DSPI PUSH FIFO Register In Master Mode (DSPI*x*_PUSHR) [0:15]
- **Safety Latent:** The register bits/fields in this category should be protected by LBIST at chip level. The following register bits/fields are classified in this category:
  - DSPI Status Register (DSPI*x*_SR) [0:15] (for stuck at 0)
  - DSPI Status Register Extended (DSPI*x*_SREX) [0:8] (for stuck at 0)
- **Non-Safety:** All other registers and bits fall in this category.

43.4 Functional description

The Deserial Serial Peripheral Interface (DSPI) block supports full-duplex, synchronous serial communications between MCUs and peripheral devices. The DSPI can also be used to reduce the number of pins required for I/O by serializing and deserializing up to 64 parallel input/output signals. All communications are done with SPI-like protocol.

SPI — DSPI operates as a basic SPI or a queued SPI.

The DCONF field in the DSPI Module Configuration Register (MCR) determines the DSPI configuration. See *Section 43.2.9.1: DSPI Module Configuration Register (DSPIx_MCR)* for the DSPI configuration values.

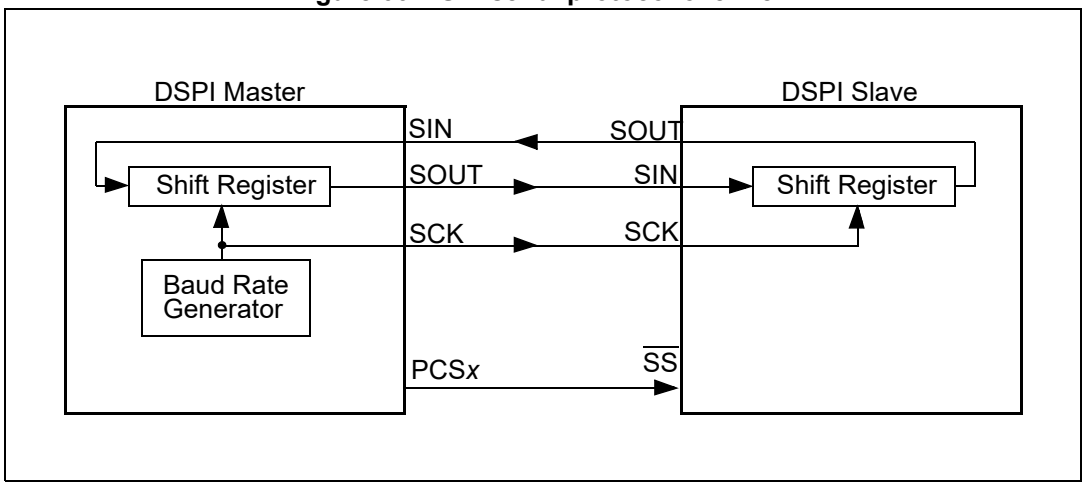
The CTAR*n* registers hold clock and transfer attributes. The SPI configuration allows to select which CTAR to use on a frame by frame basis by setting a field in the SPI command. The Extended SPI Mode (DSPI*x*_MCR[XSPI]) further allows the usage of CTARE*n* (CTAR*n* Extended) registers which allows the user to send multiple Data frames (of up to 32-bit frame size) using a single Command frame.

See DSPI Clock and Transfer Attributes Registers for information on the fields of the CTARs.

See DSPI Clock and Transfer Attributes Registers Extended for information on the fields of the CTARE registers.

Typical master to slave connections are shown in *Figure 661*. When a data transfer operation is performed, data is serially shifted a predetermined number of bit positions. Because the modules are linked, data is exchanged between the master and the slave. The data that was in the master shift register is now in the shift register of the slave, and vice versa. At the end of a transfer, the TCF bit in the SR is set to indicate a completed transfer.

Figure 661. SPI serial protocol overview



Generally more than one slave device can be connected to the DSPI master.

Eight Peripheral Chip Select (PCS) signals of the DSPI masters can be used to select which of the slaves to communicate with. Refer to *Chapter 7: Device configuration* for the number of PCS signals used in this MCU.

43.4.1 Start and Stop of DSPI transfers

The DSPI has two operating states: STOPPED and RUNNING. The states are independent of DSPI configuration. The default state of the DSPI is STOPPED. In the STOPPED state no serial transfers are initiated in master mode and no transfers are responded to in slave mode. The STOPPED state is also a safe state for writing the various configuration registers of the DSPI without causing undetermined results. In the RUNNING state serial transfers take place.

The TXRXS bit in the SR indicates what state the DSPI in. The bit is set if the module in RUNNING state.

The DSPI is started (DSPI transitions to RUNNING) when all of the following conditions are true:

- SR[EOQF] bit is clear
- MCU is not in the debug mode or the MCR[FRZ] bit is clear
- MCR[HALT] bit is clear

The DSPI stops (transitions from RUNNING to STOPPED) after the current frame when any one of the following conditions exist:

- SR[EOQF] bit is set
- MCU in the debug mode and the MCR[FRZ] bit is set
- MCR[HALT] bit is set

State transitions from RUNNING to STOPPED occur on the next frame boundary if a transfer is in progress, or immediately if no transfers are in progress.

43.4.2 Serial Peripheral Interface (SPI) configuration

The SPI configuration transfers data serially using a shift register and a selection of programmable transfer attributes. The DSPI is in SPI configuration when the DCONF field in the MCR is 0b00. The SPI frames can be 32 bits long. The host CPU or a DMA controller transfers the SPI data from the external to DSPI RAM queues to a transmit FIFO (TX FIFO) buffer. The received data is stored in entries in the Receive FIFO (RX FIFO) buffer. The host CPU or the DMA controller transfers the received data from the RX FIFO to memory external to the DSPI. The FIFO buffers operation is described in *Section 43.4.2.4: Transmit First In First Out (TX FIFO) buffering mechanism*, *Section 43.4.2.5: Command First In First Out (CMD FIFO) buffering mechanism* and *Section 43.4.2.6: Receive First In First Out (RX FIFO) Buffering Mechanism*. The interrupt and DMA request conditions are described in *Section 43.4.8*.

The SPI configuration supports two block-specific modes —master mode and slave mode. In master mode, the DSPI initiates and controls the transfer according to the fields of the executing CMD FIFO entry. In slave mode, the DSPI only responds to transfers initiated by a bus master external to the DSPI and the SPI command field space is reserved.

43.4.2.1 Master Mode

In SPI master mode the DSPI initiates the serial transfers by controlling the Serial Communications Clock (SCK) and the Peripheral Chip Select (PCS) signals. The executing SPI CMD FIFO entry determines which CTARs will be used to set the transfer attributes and which PCS signals to assert. The command field also contains various bits that help with queue management and transfer protocol. See DSPI PUSH FIFO Register (PUSHR) for details on the SPI command fields. The data in the executing TX FIFO entry is loaded into the shift register and shifted out on the Serial Out (SOUT) pin. In SPI master mode, each SPI frame to be transmitted has a command associated with it allowing for transfer attribute control on a frame by frame basis. In Extended SPI Master Mode, multiple SPI frames can have a single command associated with them allowing for efficient SPI frame transfers requiring common transfer attributes. In addition, the Extended SPI Mode allows for larger frame sizes of up to 32 bits.

43.4.2.2 Slave Mode

In SPI slave mode the DSPI responds to transfers initiated by a SPI bus master. The DSPI does not initiate transfers. Certain transfer attributes such as clock polarity, clock phase and

frame size must be set for successful communication with a SPI master. The SPI slave mode transfer attributes are set in the CTAR0 and CTARE0. The data is shifted out with MSB first. Shifting out of LSB is not supported in this mode.

43.4.2.3 FIFO Disable Operation

The FIFO disable mechanisms allow SPI transfers without using the TX FIFO, CMD FIFO or RX FIFO. The DSPI operates as a double-buffered simplified SPI when the FIFOs are disabled. The Transmit and Receive side of the FIFOs are disabled separately; setting the MCR[DIS_TXF] bit disables the TX FIFO and CMD FIFO, and setting the MCR[DIS_RXF] bit disables the RX FIFO.

The FIFO Disable mechanisms are transparent to the user and to host software; Transmit data and commands are written to the PUSHR and received data is read from the POPR.

When the TX FIFO is disabled (CMD FIFO is also disabled), the TFFF, TFUF, CMDFFF, CMDCTR and TXCTR fields in SR, SREX behave as if there is a one-entry FIFO but the contents of the TXFR registers, TXNXTPTR and CMDNXTPTR are undefined. Likewise, when the RX FIFO is disabled, the RFDF, RFOF and RXCTR fields in the SR behave as if there is a one-entry FIFO, but the contents of the RXFR registers and POPNXTPTR are undefined.

43.4.2.4 Transmit First In First Out (TX FIFO) buffering mechanism

The TX FIFO functions as a buffer of SPI data for transmission. The TX FIFO holds 4 words, each consisting of SPI data. The number of entries in the TX FIFO is device-specific. SPI data is added to the TX FIFO by writing to the Data field of DSPI PUSH FIFO Register (PUSHR). TX FIFO entries can only be removed from the TX FIFO by being shifted out or by flushing the TX FIFO.

The TX FIFO Counter field (TXCTR) in the DSPI Status Register (SR) indicates the number of valid entries in the TX FIFO. The TXCTR is updated every time an 8 or 16-bit write takes place to the Data field of DSPI_PUSHR or SPI data is transferred into the shift register from the TX FIFO.

The TXNXTPTR field indicates which TX FIFO Entry will be transmitted during the next transfer. The TXFRn Registers are invalid in the Extended SPI Mode, since the TX FIFO and CMD FIFO are used separately. The TXNXTPTR field is incremented every time SPI data is transferred from the TX FIFO to the shift register. The maximum value of the field is equal to the maximum implemented TXFR register number and it rolls over after reaching the maximum.

43.4.2.4.1 Filling the TX FIFO

Host software or other intelligent blocks can add (push) entries to the TX FIFO and CMD FIFO by writing to the PUSHR. When the TX FIFO is not full, the TX FIFO Fill Flag (TFFF) in the SR is set. The TFFF bit is cleared when TX FIFO is full and the DMA controller indicates that a write to PUSHR is complete. Writing a ‘1’ to the TFFF bit also clears it. The TFFF can generate a DMA request or an interrupt request. See *Section 43.4.8.2: Transmit FIFO Fill interrupt or DMA request* for details.

The DSPI ignores attempts to push data to a full TX FIFO, the state of the TX FIFO does not change and no error condition is indicated.

43.4.2.4.2 Draining the TX FIFO

The TX FIFO entries are removed (drained) by shifting SPI data out through the shift register. Entries are transferred from the TX FIFO to the shift register and shifted out as long as there are valid entries in the TX FIFO. Every time an entry is transferred from the TX FIFO to the shift register, the TX FIFO Counter decrements by one. When Extended SPI Mode (DSPI*x*_MCR[XSPI]) is enabled, if the frame size of SPI Data to be transmitted is more than 16 bits, then it causes two Data entries to be popped from TX FIFO simultaneously which are transferred to the shift register. The first of the two popped entries forms the 16 LSB bits of the SPI frame. Such an operation also causes TX FIFO Counter to decrement by two. At the end of a transfer, the TCF bit in the SR is set to indicate the completion of a transfer. The TX FIFO is flushed by writing a '1' to the CLR_TXF bit in MCR.

If an external bus master initiates a transfer with a DSPI slave while the slave's DSPI TX FIFO is empty, the Transmit FIFO Underflow Flag (TFUF) in the slave's SR is set. See *Section 43.4.8.7: Transmit FIFO Underflow interrupt request* for details.

43.4.2.5 Command First In First Out (CMD FIFO) buffering mechanism

The CMD FIFO functions as a buffer of SPI command used for SPI data transmission. When Extended SPI Mode (DSPI*x*_MCR[XSPI]) is disabled, the TX FIFO and CMD FIFO must be filled together. When Extended SPI Mode (DSPI*x*_MCR[XSPI]) is enabled, the TX FIFO and CMD FIFO can be filled independently.

The CMD FIFO holds four entries, each representing command fields. The number of entries in the CMD FIFO is device-specific. SPI Command is added to the CMD FIFO by writing to the command field of DSPI PUSH FIFO Register (PUSHR). CMD FIFO entries can only be removed from the CMD FIFO by being shifted out (to help transmit SPI data) or by flushing the CMD FIFO.

When Extended SPI Mode (DSPI*x*_MCR[XSPI]) is disabled, every CMDFIFO entry has a corresponding single TXFIFO entry attached to it.

When Extended SPI Mode (DSPI*x*_MCR[XSPI]) is enabled, every CMDFIFO entry can have multiple TXFIFO entries attached to it. Thus a single CMD FIFO entry can be used to transmit multiple TX FIFO entries. The CTARE[DTCP] field decides the number of SPI Data Frames having frame size as {FMSZE, FMSZ} to be transmitted using the current Command Entry. The CTAR/CTARE registers pointed by the CTAS field in the Command frame gives FMSZ and FMSZE fields respectively. The time for which a command entry is in use is known as a Command Cycle. The Busy Flag SR[BSYF] is asserted for the duration of the Command Cycle except for the last SPI frame in the Command Cycle.

The CMD FIFO Counter field (CMDCTR) in the DSPI Status Register (SR) indicates the number of valid entries in the CMD FIFO. The TXCTR is updated every time an 8 or 16-bit write takes place on the lower half of DSPI_PUSHR or SPI data is transferred into the shift register from the TX FIFO.

The CMDNXTPTR field indicates which CMD FIFO Entry will be used during the next command cycle. The TXFRn Registers are invalid in the Extended SPI Mode, since the TX FIFO and CMD FIFO are used separately. The CMDNXTPTR field is incremented every time the last SPI data in the command cycle is transferred from the TX FIFO to the shift register. The maximum value of the field is equal to the maximum implemented TXFR register number and it rolls over after reaching the maximum.

43.4.2.6 Receive First In First Out (RX FIFO) Buffering Mechanism

The RX FIFO functions as a buffer for data received on the SIN pin. The RX FIFO holds four received SPI data frames. The number of entries in the RX FIFO is device-specific. SPI data is added to the RX FIFO at the completion of a transfer when the received data in the shift register is transferred into the RX FIFO. SPI data are removed (popped) from the RX FIFO by reading the DSPI POP FIFO Register (POPR). RX FIFO entries can only be removed from the RX FIFO by reading the POPR or by flushing the RX FIFO.

The RX FIFO Counter field (RXCTR) in the DSPI Status Register (SR) indicates the number of valid entries in the RX FIFO. The RXCTR is updated every time the POPR is read or SPI data is copied from the shift register to the RX FIFO.

The POPNXTPTR field in the SR points to the RX FIFO entry that is returned when the POPR is read. The POPNXTPTR contains the positive offset from RXFR0 in number of 32-bit registers. For example, POPNXTPTR equal to two means that the RXFR2 contains the received SPI data that will be returned when POPR is read. The POPNXTPTR field is incremented every time the POPR is read. The maximum value of the field is equal to the maximum implemented RXFR register number and it rolls over after reaching the maximum.

43.4.2.6.1 Filling the RX FIFO

The RX FIFO is filled with the received SPI data from the shift register. While the RX FIFO is not full, SPI frames from the shift register are transferred to the RX FIFO. Every time a SPI frame is transferred to the RX FIFO the RX FIFO Counter is incremented by one.

If the RX FIFO and shift register are full and a transfer is initiated, the RFOF bit in the SR is set indicating an overflow condition. Depending on the state of the ROOE bit in the MCR, the data from the transfer that generated the overflow is either ignored or shifted in to the shift register. If the ROOE bit is set, the incoming data is shifted in to the shift register. If the ROOE bit is cleared, the incoming data is ignored.

43.4.2.6.2 Draining the RX FIFO

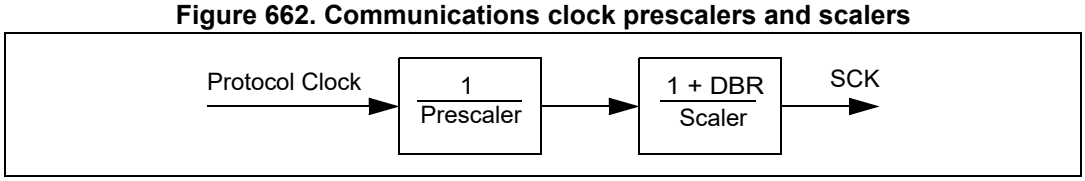
Host CPU or a DMA can remove (pop) entries from the RX FIFO by reading the DSPI POP FIFO Register (POPR). A read of the POPR decrements the RX FIFO Counter by one. Attempts to pop data from an empty RX FIFO are ignored and the RX FIFO Counter remains unchanged. The data, read from the empty RX FIFO, is undetermined.

When the RX FIFO is not empty, the RX FIFO Drain Flag (RFDF) in the SR is set. The RFDF bit is cleared when the RX_FIFO is empty and the DMA controller indicates that a read from POPR is complete or by writing a ‘1’ to it.

43.4.3 DSPI baud rate and clock delay generation

The SCK frequency and the delay values for serial transfer are generated by dividing the protocol clock frequency by a prescaler and a scaler with the option for doubling the baud rate. *Figure 662* shows conceptually how the SCK signal is generated.

Figure 662. Communications clock prescalers and scalers



43.4.3.1 Baud rate generator

The baud rate is the frequency of the Serial Communication Clock (SCK). The protocol clock is divided by a prescaler (PBR) and scaler (BR) to produce SCK with the possibility of halving the scaler division. The DBR, PBR and BR fields in the CTARs select the frequency of SCK by the formula in the BR field description. *Table 715* shows an example of how to compute the baud rate.

Table 715. Baud rate computation example

f_p	PBR	Prescaler	BR	Scaler	DBR	Baud rate
100 MHz	0b00	2	0b0000	2	0	25 Mb/s
20 MHz	0b00	2	0b0000	2	1	10 Mb/s

Note: The clock frequencies mentioned in Table 715 are given as an example. Refer to the clocking chapter for the frequency used to drive this module in the device.

43.4.3.2 PCS to SCK delay (t_{CSC})

The PCS to SCK delay is the length of time from assertion of the PCS signal to the first SCK edge. See *Table 716* for an illustration of the PCS to SCK delay. The PCSSCK and CSSCK fields in the CTAR*n* registers select the PCS to SCK delay by the formula in the CSSCK field description. *Table 716* shows an example of how to compute the PCS to SCK delay.

Table 716. PCS to SCK delay computation example

f_p	PCSSCK	Prescaler	CSSCK	Scaler	PCS to SCK delay
100 MHz	0b01	3	0b0100	32	0.96 µs

Note: The clock frequency mentioned in Table 716 is given as an example. Refer to the clocking chapter for the frequency used to drive this module in the device.

43.4.3.3 After SCK delay (t_{ASC})

The After SCK delay is the length of time between the last edge of SCK and the negation of PCS. See *Table 717* for an illustration of the After SCK delay. The PASC and ASC fields in the CTAR*n* registers select the After SCK Delay by the formula in the ASC field description. *Table 717* shows an example of how to compute the After SCK delay.

Table 717. After SCK delay computation example

f_p	PASC	Prescaler	ASC	Scaler	After SCK delay
100 MHz	0b01	3	0b0100	32	0.96 µs

Note: The clock frequency mentioned in Table 717 is given as an example. Refer to the clocking chapter for the frequency used to drive this module in the device.

43.4.3.4 Delay after transfer (t_{DT})

The delay after transfer is the minimum time between negation of the PCS signal for a frame and the assertion of the PCS signal for the next frame. See *Table 718* for an illustration of the delay after transfer. the pdt and dt fields in the CTAR*n* registers select the Delay after Transfer by the formula in the DT field description. *Table 718* shows an example of how to compute the delay after transfer.

Table 718. Delay after transfer computation example

f_p	PDT	Prescaler	DT	Scaler	Delay after transfer
100 MHz	0b01	3	0b1110	32768	0.98 ms

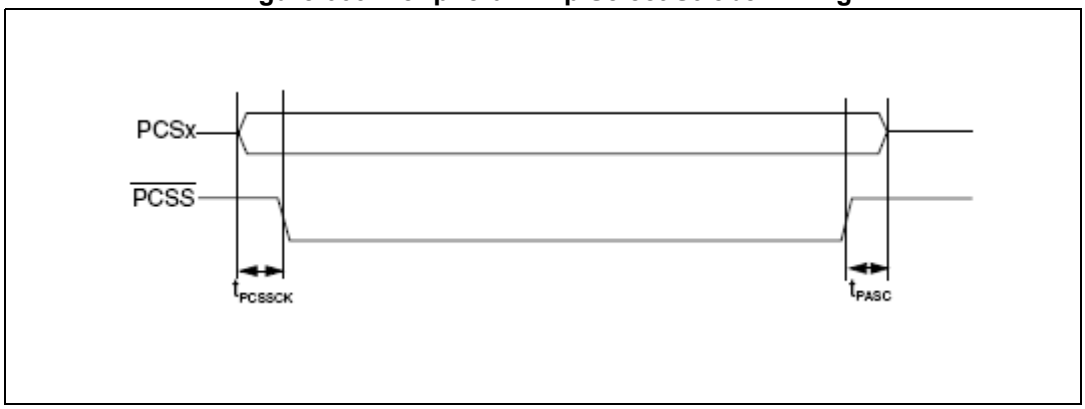
Note: The clock frequency mentioned in Table 718 is given as an example. Refer to the clocking chapter for the frequency used to drive this module in the device.

When in non-continuous clock mode the t_{DT} delay is configured according to the equation specified in the CTAR[DT] bitfield description. When in continuous clock mode is not enabled, the delay is fixed at 1 SCK period.

43.4.3.5 Peripheral Chip Select Strobe Enable ($\overline{PCSS}$)

The PCSS signal provides a delay to allow the PCS signals to settle after a transition occurs thereby avoiding glitches. When the DSPI is in master mode and the PCSSE bit is set in the MCR, $\overline{PCSS}$ provides a signal for an external demultiplexer to decode the PCS[0]–PCS[4] and PCS[6]–PCS[7] signals into as many as 128 glitch-free PCS signals. *Figure 663* shows the timing of the $\overline{PCSS}$ signal relative to PCS signals.

Figure 663. Peripheral Chip Select Strobe Timing



The delay between the assertion of the PCS signals and the assertion of PCSS is selected by the PCSSCK field in the CTAR based on the following formula:

Equation 26

$$t_{PCSSCK} = \frac{1}{f_P} \times PCSSCK$$

At the end of the transfer the delay between $\overline{PCSS}$ negation and PCS negation is selected by the PASC field in the CTAR based on the following formula:

Equation 27

$$t_{PASC} = \frac{1}{f_P} \times PASC$$

Table 719 shows an example of how to compute the t_{PCSSCK} delay.

Table 719. Peripheral Chip Select Strobe Assert Computation Example

f_p	PCSSCK	Prescaler	Delay before transfer
100 MHz	0b11	7	70.0 ns

Table 720 shows an example of how to compute the t_{PASC} delay.

Table 720. Peripheral Chip Select Strobe Negate Computation Example

f_p	PASC	Prescaler	Delay after transfer
100 MHz	0b11	7	70.0 ns

The $\overline{PCSS}$ signal is not supported when Continuous Serial Communication SCK is enabled.

Note: The clock frequency mentioned in the preceding Table 719 and Table 720 is given as an example. Refer to the clocking chapter for the frequency used to drive this module in the device.

43.4.4 Transfer formats

The SPI serial communication is controlled by the Serial Communications Clock (SCK) signal and the PCS signals. The SCK signal provided by the master device synchronizes shifting and sampling of the data on the SIN and SOUT pins. The PCS signals serve as enable signals for the slave devices.

In master mode, the CPOL and CPHA bits in the Clock and Transfer Attributes Registers (CTAR*n*) select the polarity and phase of the serial clock, SCK.

- CPOL—Selects the idle state polarity of the SCK
- CPHA—Selects if the data on SOUT is valid before or on the first SCK edge

Even though the bus slave does not control the SCK signal, in slave mode these values must be identical to the master device settings to ensure proper transmission. In SPI slave mode, only CTAR0 is used.

The DSPI supports four different transfer formats:

- Classic SPI with CPHA = 0
- Classic SPI with CPHA = 1
- Modified Transfer format with CPHA = 0
- Modified Transfer format with CPHA = 1

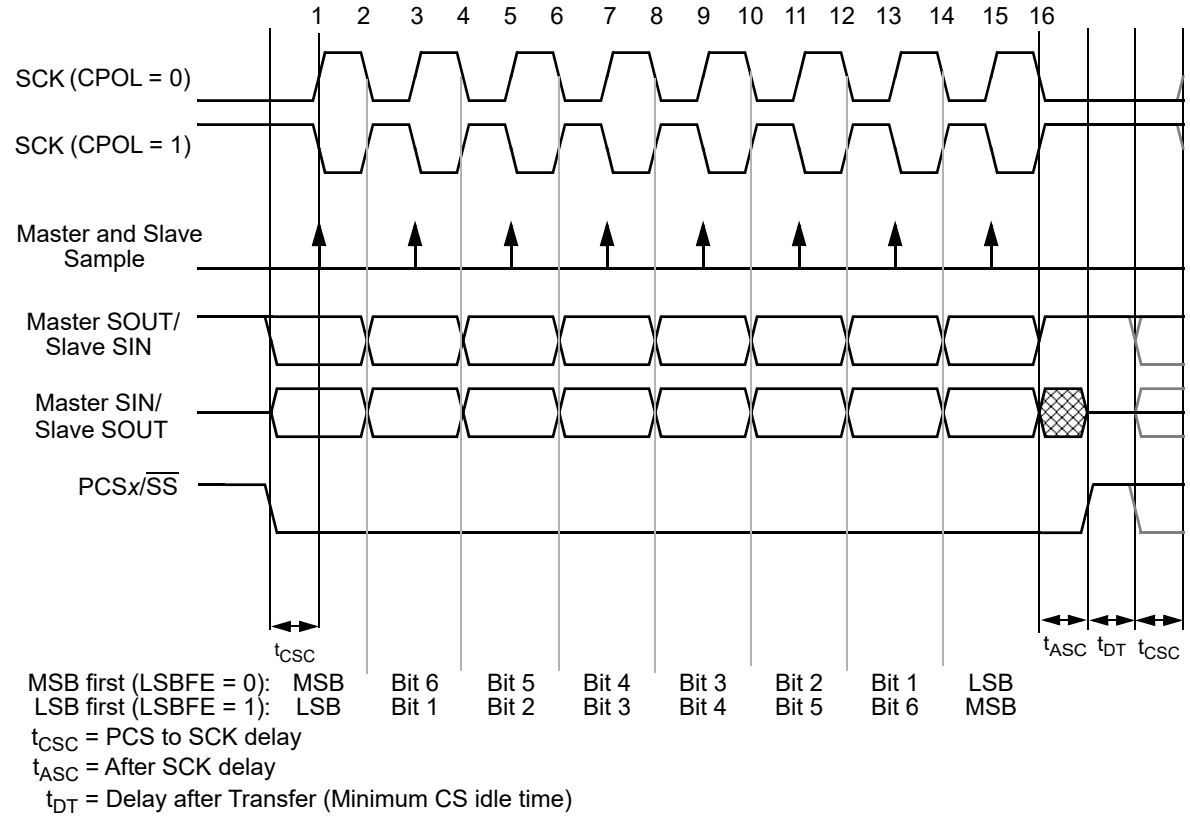
A modified transfer format is supported to allow for high-speed communication with peripherals that require longer setup times. The DSPI can sample the incoming data later than halfway through the cycle to give the peripheral more setup time. The MTFE bit in the MCR selects between Classic SPI Format and Modified Transfer Format.

In the SPI configuration, the DSPI provides the option of keeping the PCS signals asserted between frames. See *Section 43.4.4.5: Continuous Selection Format* for details.

43.4.4.1 Classic SPI Transfer Format (CPHA = 0)

The transfer format shown in *Figure 664* is used to communicate with peripheral SPI slave devices where the first data bit is available on the first clock edge. In this format, the master and slave sample their SIN pins on the odd-numbered SCK edges and change the data on their SOUT pins on the even-numbered SCK edges.

Figure 664. DSPI transfer timing diagram (MTFE = 0, CPHA = 0, FMSZ = 8)

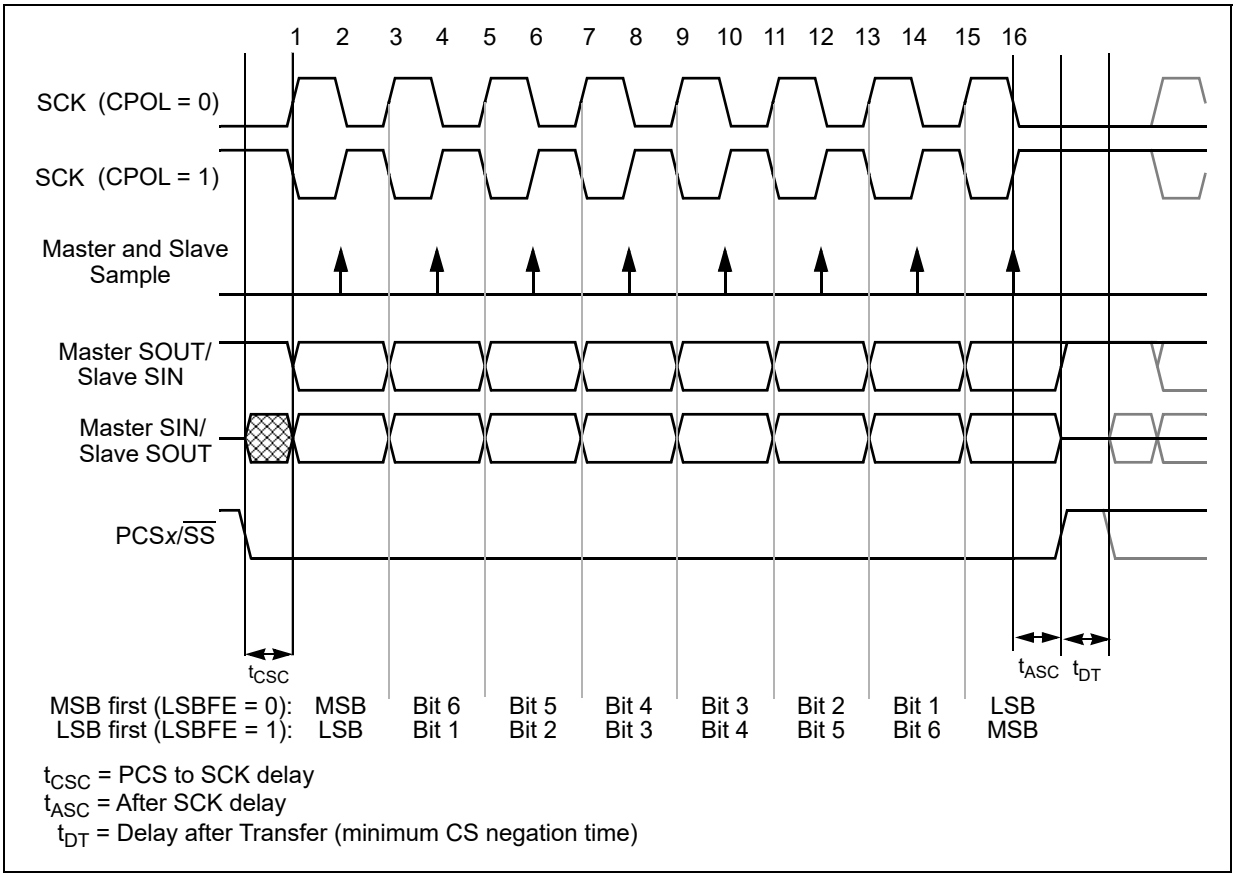


The master initiates the transfer by placing its first data bit on the SOUT pin and asserting the appropriate peripheral chip select signals to the slave device. The slave responds by placing its first data bit on its SOUT pin. After the t_{CSC} delay elapses, the master outputs the first edge of SCK. The master and slave devices use this edge to sample the first input data bit on their serial data input signals. At the second edge of the SCK the master and slave devices place their second data bit on their serial data output signals. For the rest of the frame the master and the slave sample their SIN pins on the odd-numbered clock edges and changes the data on their SOUT pins on the even-numbered clock edges. After the last clock edge occurs a delay of t_{ASC} is inserted before the master negates the PCS signals. A delay of t_{DT} is inserted before a new frame transfer can be initiated by the master.

43.4.4.2 Classic SPI Transfer Format (CPHA = 1)

This transfer format shown in *Figure 665* is used to communicate with peripheral SPI slave devices that require the first SCK edge before the first data bit becomes available on the slave SOUT pin. In this format the master and slave devices change the data on their SOUT pins on the odd-numbered SCK edges and sample the data on their SIN pins on the even-numbered SCK edges.

Figure 665. DSPI transfer timing diagram (MTFE = 0, CPHA = 1, FMSZ = 8)



The master initiates the transfer by asserting the PCS signal to the slave. After the t_{CSC} delay has elapsed, the master generates the first SCK edge and at the same time places valid data on the master SOUT pin. The slave responds to the first SCK edge by placing its first data bit on its slave SOUT pin.

At the second edge of the SCK the master and slave sample their SIN pins. For the rest of the frame the master and the slave change the data on their SOUT pins on the odd-numbered clock edges and sample their SIN pins on the even-numbered clock edges. After the last clock edge occurs a delay of t_{ASC} is inserted before the master negates the PCS signal. A delay of t_{DT} is inserted before a new frame transfer can be initiated by the master.

43.4.4.3 Modified SPI Transfer Format (MTFE = 1, CPHA = 0)

In this Modified Transfer Format both the master and the slave sample later in the SCK period than in Classic SPI mode to allow the logic to tolerate more delays in device pads and board traces. These delays become a more significant fraction of the SCK period as the SCK period decreases with increasing baud rates.

The master and the slave place data on the SOUT pins at the assertion of the PCS signal. After the PCS to SCK delay has elapsed the first SCK edge is generated. The slave samples the master SOUT signal on every odd numbered SCK edge. The DSPI in the slave mode when the MTFE bit is set also places new data on the slave SOUT on every odd numbered clock edge. Regular external slave, configured with CPHA=0 format drives its SOUT output at every even numbered SCK clock edge.

The DSPI master places its second data bit on the SOUT line one protocol clock after odd numbered SCK edge if the protocol clock frequency to SCK frequency ratio is higher than three. If this ratio is below four the master changes SOUT at odd numbered SCK edge. The point where the master samples the SIN is selected by the DSPI_MCR[SMPL_PT] field. The master sample point can be delayed by one or two protocol clock cycles. The SMPL_PT field should be set to 0 if the protocol to SCK frequency ratio is less than 4. However if this ratio is less than 4, the actually sample point is delayed by 1 protocol clock cycle automatically by the design.

The following timing diagrams illustrate the DSPI operation with MTFE = 1. Timing delays shown are:

- t_{CSC} – PCS to SCK assertion delay
- t_{ACS} – After SCK PCS negation delay
- t_{su_ms} – master SIN setup time
- t_{hd_ms} – master SIN hold time
- t_{vd_sl} – slave data output valid time, time between slave data output SCK driving edge and data becomes valid.
- t_{su_sl} – data setup time on slave data input
- t_{hd_sl} – data hold time on slave data input
- t_p – protocol clock period

Figure 666 shows the modified transfer format for CPHA = 0 and f_p/f_{sck} = 4. Only the condition where CPOL = 0 is illustrated. Solid triangles show the data sampling clock edges. The two possible slave behaviors are shown.

- Signal, marked “SOUT of Ext Slave”, presents regular SPI slave serial output.
- Signal, marked “SOUT of DSPI Slave”, presents DSPI in the slave mode with MTFE bit set.

Other MTFE = 1 diagrams show DSPI SIN input as being driven by a regular external SPI slave, configured according DSPI master CPHA programming.

Note: In the following diagrams, f_p represents the protocol clock frequency from which the Baud frequency f_{sck} is derived.

Figure 666. DSPI Modified Transfer Format (MTFE = 1, CPHA = 0, $f_{sck} = f_p/4$)

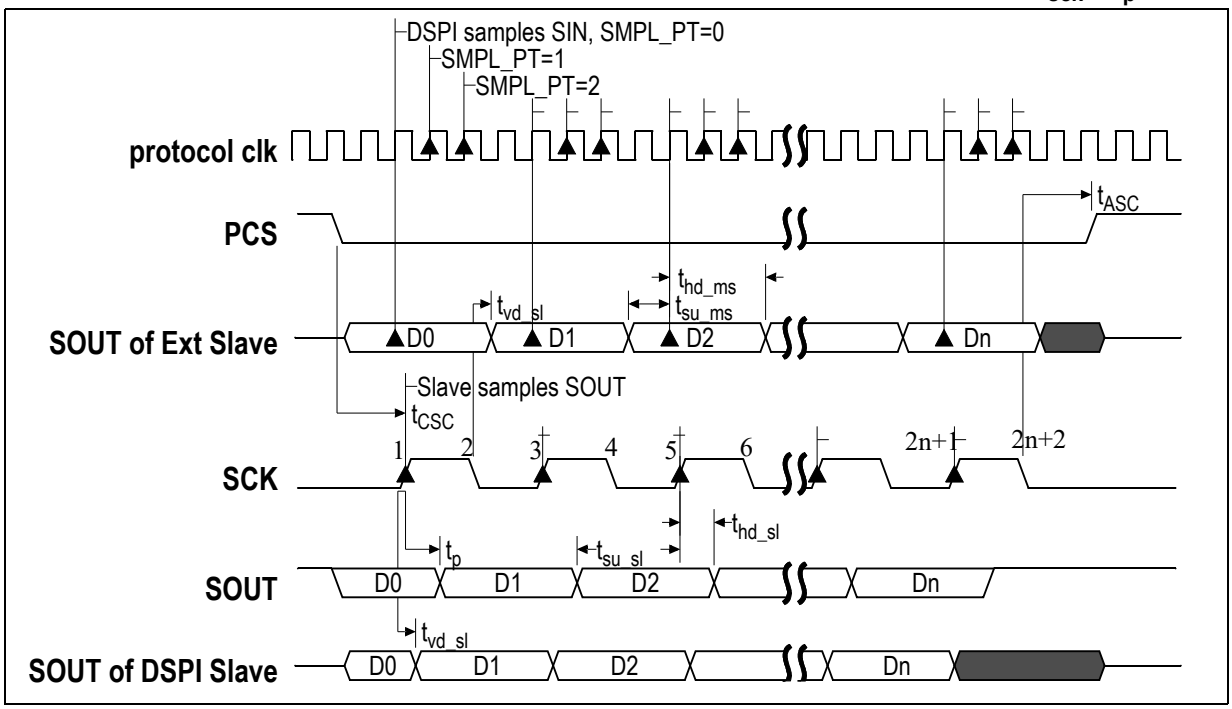


Figure 667. DSPI Modified Transfer Format (MTFE = 1, CPHA = 0, $f_{sck} = f_p/2$)

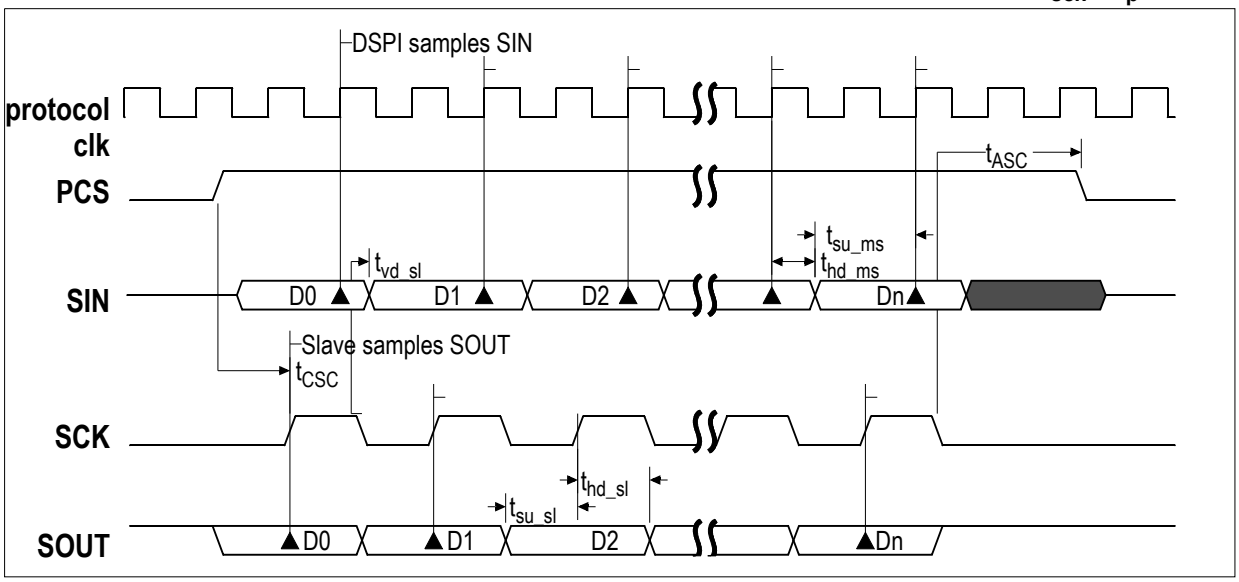
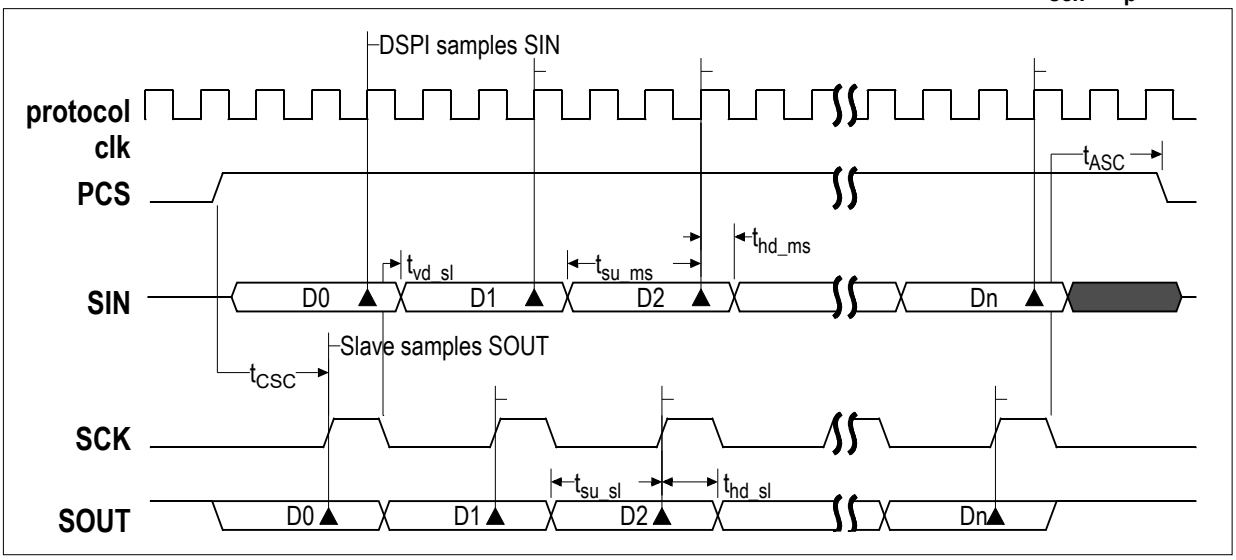


Figure 668. DSPI Modified Transfer Format (MTFE = 1, CPHA = 0, $f_{sck} = f_p/3$)



43.4.4.4 Modified SPI Transfer Format (MTFE = 1, CPHA = 1)

The following figures show the Modified Transfer Format for CPHA = 1. Only the condition, where CPOL = 0 is shown. At the start of a transfer the DSPI asserts the PCS signal to the slave device. After the PCS to SCK delay has elapsed the master and the slave put data on their SOUT pins at the first edge of SCK. The slave samples the master SOUT signal on the even numbered edges of SCK. The master samples the slave SOUT signal on the odd numbered SCK edges starting with the third SCK edge. The slave samples the last bit on the last edge of the SCK. The master samples the last slave SOUT bit one half SCK cycle after the last edge of SCK. No clock edge will be visible on the master SCK pin during the sampling of the last bit. **The SCK to PCS delay must be greater or equal to half of the SCK period.**

Figure 669. DSPI Modified Transfer Format (MTFE = 1, CPHA = 1, $f_{sck} = f_p/2$)

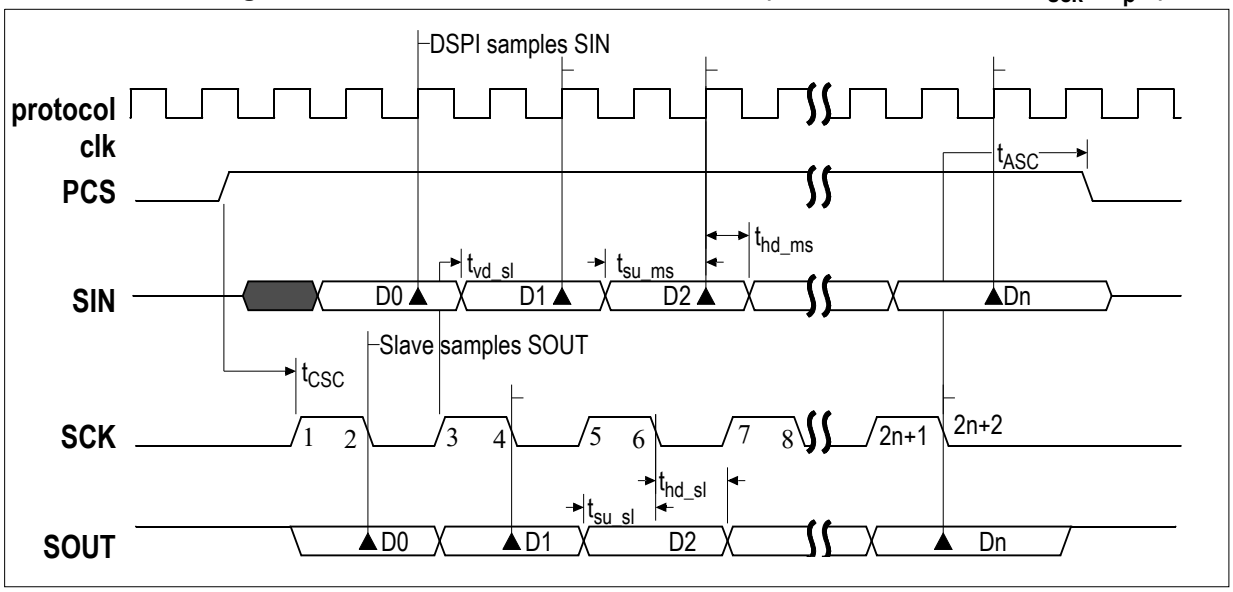


Figure 670. DSPI Modified Transfer Format (MTFE = 1, CPHA = 1, $f_{sck} = f_p/3$)

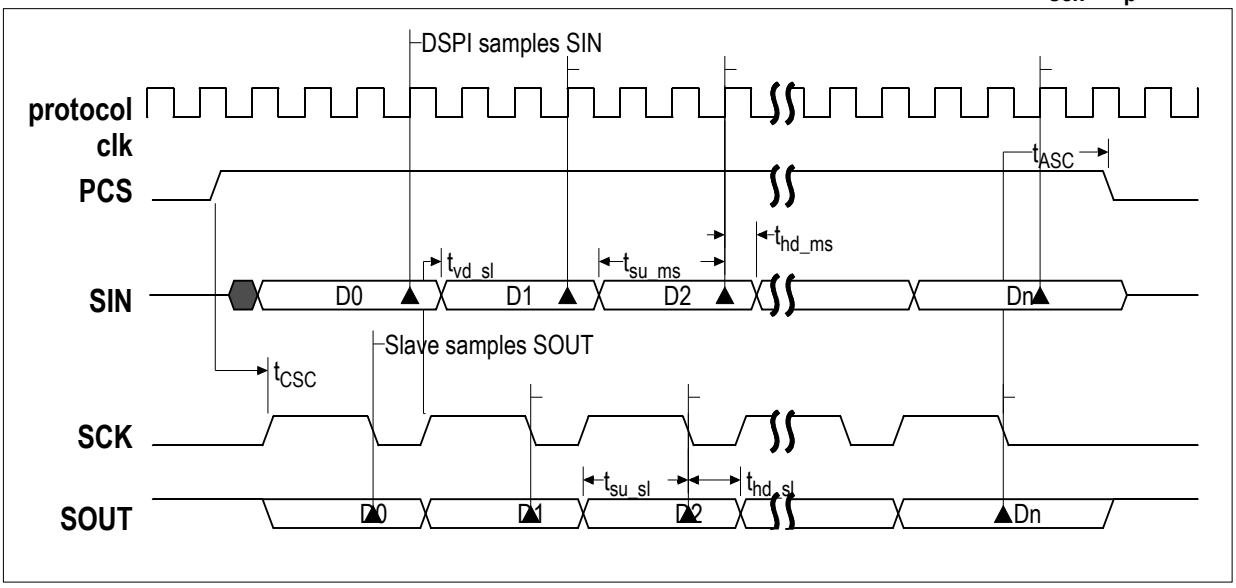
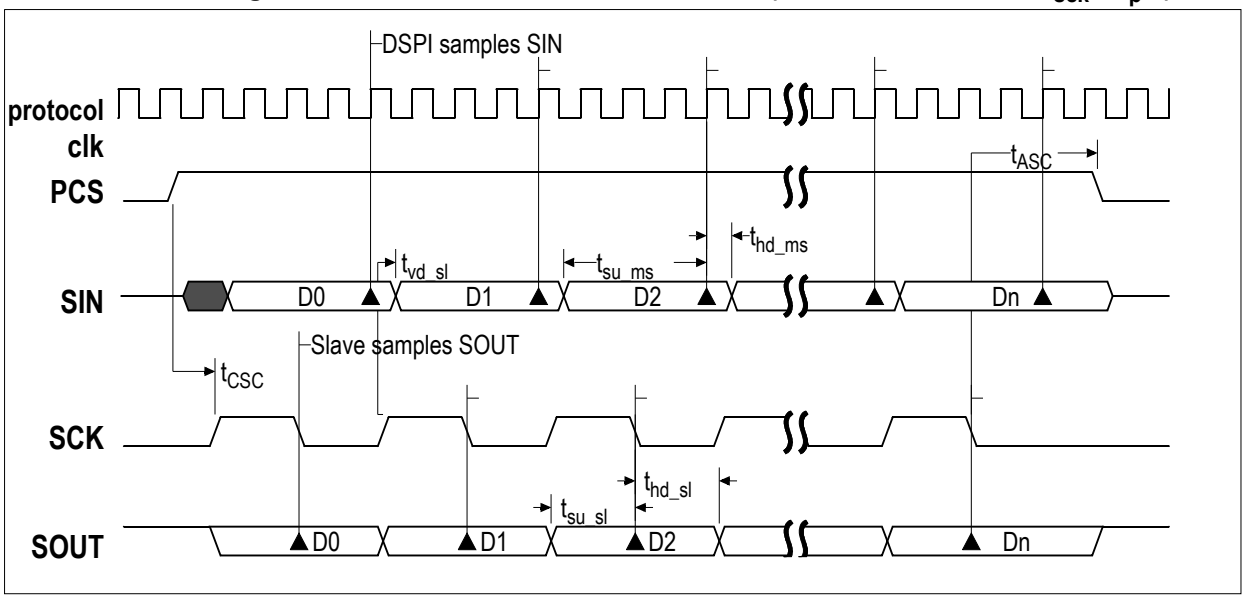


Figure 671. DSPI Modified Transfer Format (MTFE = 1, CPHA = 1, $f_{sck} = f_p/4$)

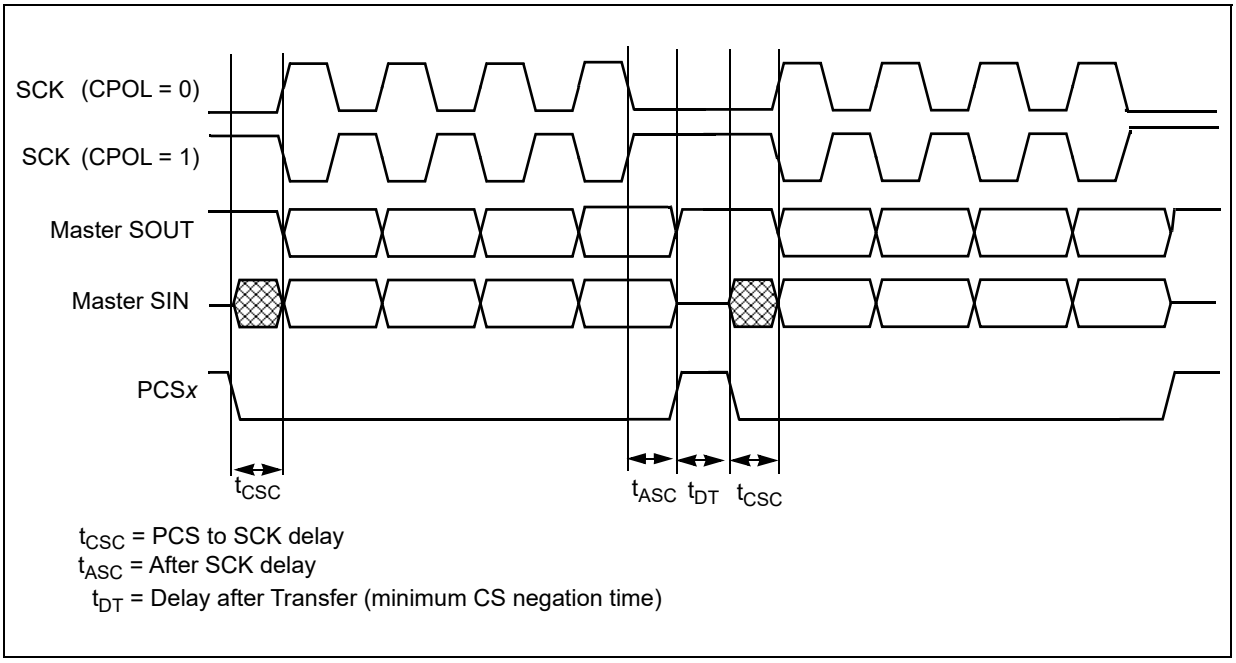


43.4.4.5 Continuous Selection Format

Some peripherals must be deselected between every transfer. Other peripherals must remain selected between several sequential serial transfers. The Continuous Selection Format provides the flexibility to handle both the following cases. The Continuous Selection Format is enabled for the SPI configuration by setting the CONT bit in the SPI command.

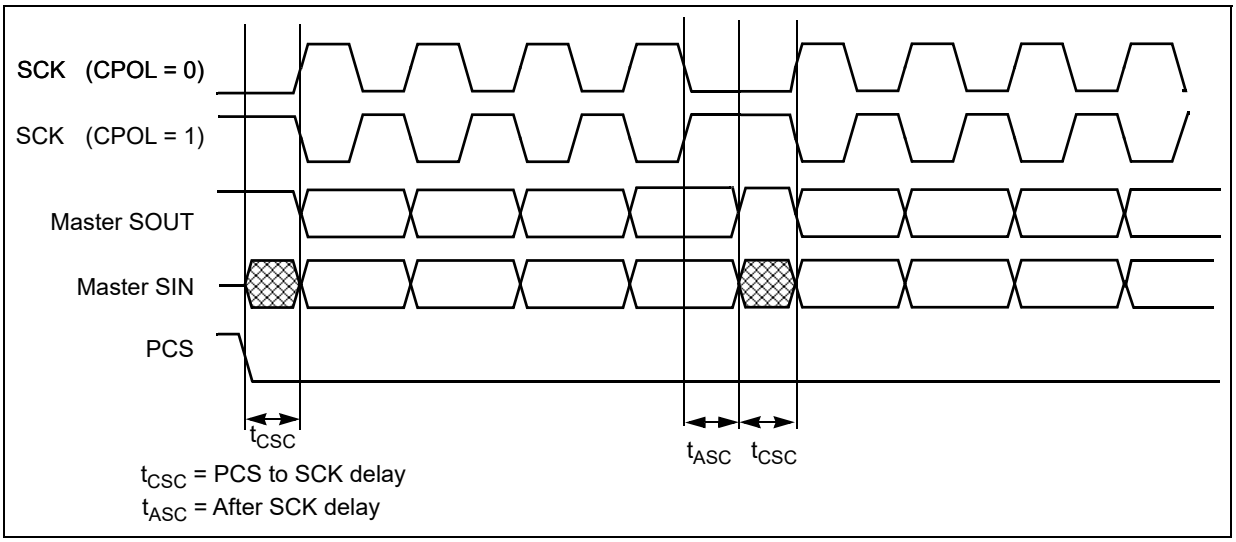
The behavior of the PCS signals in the two configurations is identical so only SPI configuration will be described. When the CONT bit = 0, the DSPI drives the asserted Chip Select signals to their idle states in between frames. The idle states of the Chip Select signals are selected by the PCSISn bits in the MCR. The timing diagram in *Figure 672* is for two four-bit transfers with CPHA = 1 and CONT = 0.

Figure 672. Example of non-continuous format (CPHA = 1, CONT = 0)



When the CONT bit = 1, the PCS signal remains asserted for the duration of the two transfers. The Delay between Transfers (t_{DT}) is not inserted between the transfers. *Figure 673* shows the timing diagram for two four-bit transfers with CPHA = 1 and CONT = 1.

Figure 673. Example of continuous transfer (CPHA = 1, CONT = 1)



When using DSPI with continuous selection follow these rules:

- All transmit commands must have the same PCSn bits programming.
- The CTARs, selected by transmit commands, must be programmed with the same transfer attributes. Only FMSZ field can be programmed differently in these CTARs.
- When transmitting multiple frames in this mode, the user software must ensure that the last frame has the PUSHR[CONT] bit de-asserted (in master mode) and the user

software must provide sufficient frames in the TX_FIFO to be sent out (in slave mode) and the master de-asserts the PCSn at end of transmission of last frame.

- The PUSHR[CONT] bit must be de-asserted before asserting MCR[HALT] bit (in master mode). This will make sure that the PCSn signals are deasserted. Asserting MCR[HALT] bit during continuous transfer will cause the PCSn signals to remain asserted and hence Slave Device cannot transition from RUNNING to STOPPED state.

Note: You must fill the TXFIFO with the number of entries that will be concatenated together under one PCS assertion for both master and slave before the TXFIFO becomes empty.

When operating in slave mode, ensure that when the last-entry in the TXFIFO is completely transmitted (that is the corresponding TCF flag is asserted and TXFIFO is empty), the slave is deselected for any further serial communication; otherwise, an underflow error occurs.

43.4.4.6 Fast Continuous Selection Format

The Fast Continuous Selection Format functions similar to the Continuous Selection Format except that the inter command delays, t_{ASC} and t_{CSC}, can be masked out and are not inserted by the hardware.

Note: The Fast Continuous Selection Format is available in the SPI configuration only and when Continuous Serial Communication Clock mode is disabled.

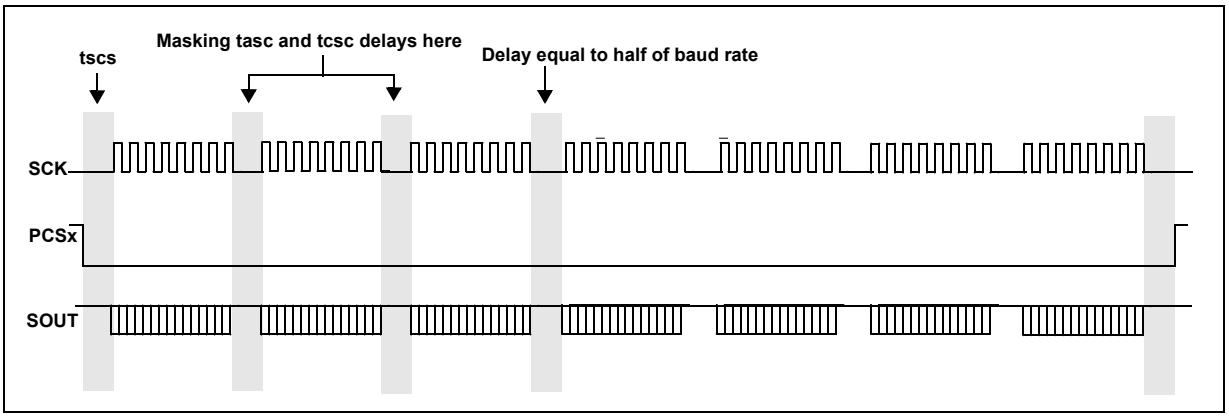
The Fast Continuous Selection Format is enabled by writing '1' to bit MCR[FCPCS]. When this bit is asserted, MASC and MCSC bits of the PUSHR perform the function of mask bits for the transmit frame. These bits individually mask the t_{ASC} and t_{CSC} delays as programmed by the user software. A normal Continuous Selection Format has these two delays for each frame that is transmitted with the CONT bit asserted. In order to avoid these delays and speed up the transfer process, the software can simply mask these delays while programming the command in the PUSHR.

While masking the delays, the software must follow the following masking rules, else correct operation is not guaranteed.

- MASC bit masks the "After SCK" delay for the current frame.
- MCSC bit masks the "PCS to SCK" delay for the next frame.
- "After SCK" (t_{ASC}) delay must not be masked when the current frame is the last frame in the continuous selection format.
- The "PCS to SCK" delay for the first frame in the continuous selection format cannot be masked.
- Masking of only t_{ASC} is not allowed. If t_{ASC} is masked then t_{CSC} must be masked too.
- Masking of both t_{ASC} and t_{CSC} delays is allowed. In this case, the delay between two frames is equal to half the baud rate set by the user software.
- Masking of only t_{CSC} is allowed. In this case, the delay between two frames is equal to the t_{ASC} time and thus the user software must ensure that the t_{ASC} time is greater than the baud rate.
- The user software must not mask these delays if the continuous selection format is not used and MCR[FCPCS] is asserted.
- Rules applicable to the Continuous Selection Format are applicable here, too.

Figure 674 shows the timing for a Fast Continuous Selection Format transfer. Here seven frames are transferred with both t_{ASC} and t_{CSC} delays masked except for the last frame that terminated the transfer. The last frame has t_{ASC} delay at its end.

Figure 674. Example of Fast Continuous Selection Format



In case any chip select is to be changed, then the fast continuous selection format should be terminated and then the chips selects should change and appropriate delays must be introduced.

43.4.5 Continuous Serial Communications Clock

The DSPI provides the option of generating a continuous SCK signal for slave peripherals that require a continuous clock.

Continuous SCK is enabled by setting bit MCR[CONT_SCKE]. Enabling this bit generates the Continuous Serial Communications Clock regardless of the MCR[HALT] bit status. Continuous SCK is valid in all configurations.

Continuous SCK is only supported for CPHA = 1. Clearing CPHA is ignored if the CONT_SCKE bit is set. Continuous SCK is supported for Modified Transfer Format.

Clock and transfer attributes for the Continuous SCK mode are set according to the following rules:

- When the DSPI is in SPI configuration, CTAR0 is used initially. At the start of each SPI frame transfer, the CTAR specified by the CTAS for the frame is used.
- In SPI configuration, the currently selected CTAR remains in use until the start of a frame with a different CTAR specified, or the Continuous SCK mode is terminated.

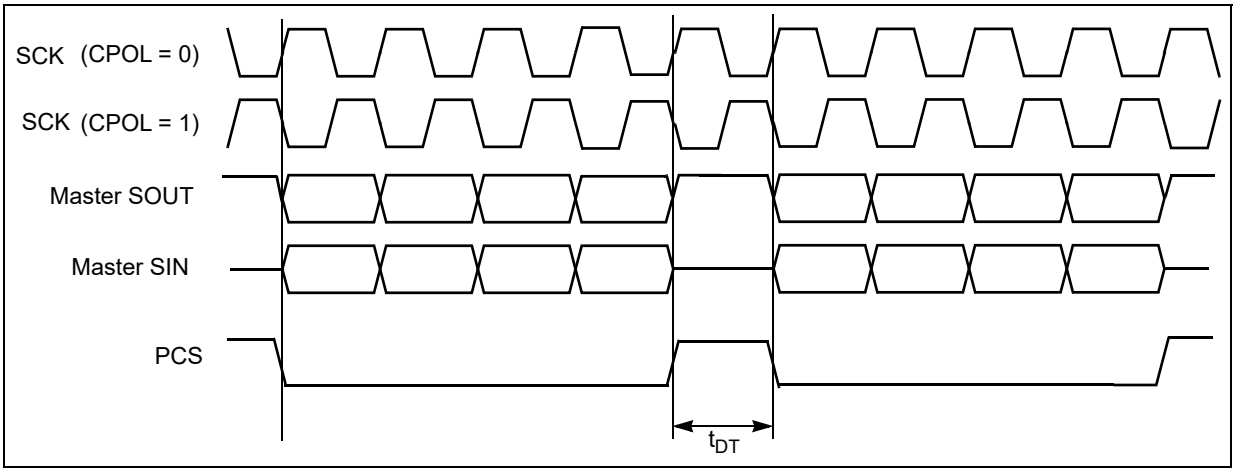
It is recommended to keep the baud rate the same while using the Continuous SCK. Switching clock polarity between frames while using Continuous SCK can cause errors in the transfer. Continuous SCK operation is not guaranteed if the DSPI is put into the External Stop mode or Module Disable mode.

Enabling Continuous SCK disables the PCS to SCK delay and the Delay after Transfer (t_{DT}) is fixed to one SCK cycle.

Figure 675 is the timing diagram for Continuous SCK format with Continuous Selection disabled.

Note: When in Continuous SCK mode, for the SPI transfer CTAR0 should always be used, and the TXFIFO must be cleared using the MCR[CLR_TXF] field before initiating transfer.

Figure 675. Continuous SCK timing diagram (CONT = 0)

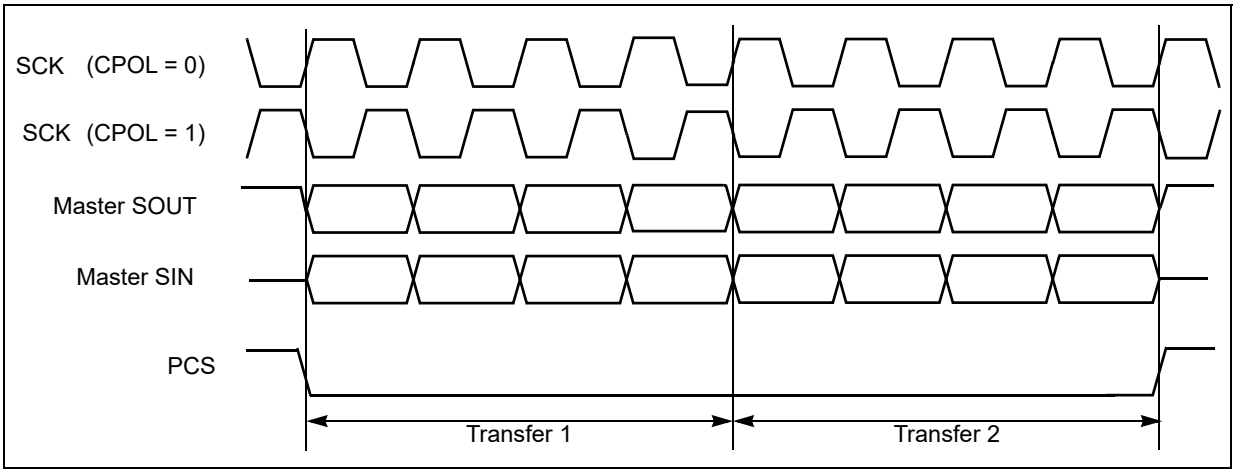


If the CONT bit in the TX FIFO entry is set, PCS remains asserted between the transfers. Under certain conditions, SCK can continue with PCS asserted, but with no data being shifted out of SOUT (SOUT pulled high). This can cause the slave to receive incorrect data. Those conditions include:

- Continuous SCK with CONT bit set, but no data in the transmit FIFO.
- Continuous SCK with CONT bit set and entering STOPPED state (refer to *Section 43.4.1*.
- Continuous SCK with CONT bit set and entering Stop mode or Module Disable mode.

Figure 676 shows the timing diagram for Continuous SCK format with Continuous Selection enabled.

Figure 676. Continuous SCK Timing Diagram (CONT = 1)



43.4.6 Slave mode operation constraints

Slave mode logic shift register is buffered. This allows data streaming operation, when the DSPI is permanently selected and data is shifted in with a constant rate.

The transmit data is transferred at second SCK clock edge of the each frame to the shift register if the $\overline{SS}$ signal is asserted and any time when transmit data is ready and SS signal is negated.

Received data is transferred to the receive buffer at last SCK edge of each frame, defined by frame size programmed to the CTAR0/1 register. Then the data from the buffer is transferred to the RXFIFO or DDR1/0 register.

If the $\overline{SS}$ negates before that last SCK edge, the data from shift register is lost.

43.4.7 Parity generation and check

The DSPI module can generate and check parity in the serial frame. The parity bit replaces the last transmitted bit in the frame. The parity is calculated for all transmitted data bits in frame, not including the last data bit that would be transmitted. The parity generation/control is done on frame basis. The register fields setting frame size defines the total number of bits in the frame, including the parity bit. Thus, to transmit/receive the same number of data bits with parity check, increase the frame size by one versus the same data size frame without the parity check.

Parity can be selected as odd or even. Parity Errors in the received frame set Parity Error flags in the Status register. The Parity Error Interrupt Requests are generated if enabled. The DSPI module can be programmed to stop SPI frame transmission in case of a frame reception with parity error.

43.4.7.1 Parity for SPI frames

When the DSPI is in the master mode the parity generation is controlled by PE and PP bits of the TX FIFO entries (PUSHR). Setting the PE bit enables parity generation for transmitted SPI frames and parity check for received frames. PP bit defines polarity of the parity bit.

When continuous PCS selection is used to transmit SPI data, two parity generation scenarios are available:

- Generate/check parity for the whole frame
- Generate/check parity for each subframe separately.

To generate/check parity for the whole frame set PE bit only in the last command/TX FIFO entry, forming this frame (with the PUSHR).

To generate/check parity for each subframe set PE bit in each command/TX FIFO entry, forming this frame.

If the parity error occurs for received SPI frame, the SR[SPEF] bit is set. If MCR[PES] bit is set, the DSPI stops SPI frames transmission. To resume SPI operation clear the SR[SPEF] or the MCR[PES] bits.

In slave mode, the parity is controlled by the PE and PP bits of the CTAR0 register similar to the master mode parity generation without continuous PCS selection.

43.4.8 Interrupts/DMA requests

Certain DSPI conditions can generate only interrupt requests and other DSPI conditions can generate either interrupt requests or DMA requests. *Table 721* lists these conditions.

Table 721. Interrupt and DMA request conditions

DSPI condition	Flag	Request type	
		Interrupt	DMA
End of Queue (EOQ)	EOQF	Yes	—
TX FIFO Fill	TFFF	Yes	Yes
CMD FIFO Fill	CMDFFF	Yes	Yes
TX FIFO Invalid Write	TFIWF	Yes	—
Transfer Complete	TCF	Yes	—
CMD Transfer Complete	CMDTCF	Yes	—
SPI Transfer Complete	SPITCF	Yes	—
TX FIFO Underflow	TFUF	Yes	—
RX FIFO Drain	RFDF	Yes	Yes
RX FIFO Overflow	RFOF	Yes	—
SPI Parity Error	SPEF	Yes	—

Each condition has a flag bit in the DSPI Status Register (SR) and a Request Enable bit in the DSPI DMA/Interrupt Request Select and Enable Register (RSER). The TX FIFO Fill Flag (TFFF) and RX FIFO Drain Flag (RFDF) generate interrupt requests or DMA requests depending on the TFFF_DIRS and RFDF_DIRS bits in the RSER.

The DSPI module also provides a global interrupt request line, which is asserted when any of individual interrupt requests lines is asserted.

43.4.8.1 End of Queue interrupt request

The End of Queue Request indicates that the end of a transmit queue is reached. The End of Queue Request is generated when the EOQ bit in the executing SPI command is set and RSER[EOQF_RE] is set.

When MCR[XSPI] is enabled and the EOQ bit in the executing SPI command is set, the End of Queue Request will only be generated once the last Data frame in the Command Cycle has been transmitted.

Note: This interrupt request is generated when the last bit of the SPI frame with EOQ bit set is transmitted.

43.4.8.2 Transmit FIFO Fill interrupt or DMA request

The Transmit FIFO Fill Request indicates that the TX FIFO is not full. The Transmit FIFO Fill Request is generated when the number of entries in the TX FIFO is less than the maximum number of possible entries, and the TFFF_RE bit in the RSER is set. The TFFF_DIRS bit in the RSER selects whether a DMA request or an interrupt request is generated.

Note: TFFF flag clears automatically when DMA is used to fill TXFIFO.

To clear TFFF when not using DMA, follow these steps for every PUSH performed using CPU to fill TXFIFO:

- *Wait until TFFF = 1*

- *Write data to PUSHR using CPU.*
- *Clear TFFF by writing a '1' to its location. If TX FIFO is not full, this flag will not clear.*

43.4.8.3 Command FIFO Fill interrupt or DMA request

The Command FIFO Fill Request indicates that the CMD FIFO is not full. The Command FIFO Fill Request is generated when the number of entries in the CMD FIFO is less than the maximum number of possible entries, and the CMDFFF_RE bit in the RSER is set. The CMDFFF_DIRS bit in the RSER selects whether a DMA request or an interrupt request is generated.

This Request is useful when MCR[XSPI] is enabled and hence TX FIFO and CMD FIFO can be filled independently. If MCR[XSPI] is disabled, then 'TX FIFO Fill Interrupt or DMA Request' will suffice to fill both FIFO's since both FIFO's must be filled simultaneously.

Note: CMDFFF flag clears automatically when DMA is used to fill CMD FIFO.

To clear CMDFFF when not using DMA, follow these steps for every PUSH performed using CPU to fill CMD FIFO:

- *Wait until CMDFFF = 1*
- *Write data to Command field of PUSHR using CPU.*
- *Clear CMDFFF by writing a '1' to its location. If CMD FIFO is not full, this flag will not clear.*

43.4.8.4 Transmit FIFO Invalid Write interrupt request

The Transfer FIFO Invalid Write Request is valid only when MCR[XSPI] is enabled. This Request indicates that Data exists in the TX FIFO while the CMD FIFO is empty. Since no Command Fields are associated with the Data present in TX FIFO, this data is considered invalid until a Command Entry becomes available. The Transfer FIFO Invalid Write Request is generated for the above condition when RSER[TFIWF_RE] is set.

43.4.8.5 Transfer Complete interrupt request

The Transfer Complete Request indicates the end of the transfer of a serial frame. The Transfer Complete Request is generated at the end of each frame transfer when RSER[TCF_RE] is set.

43.4.8.6 Command Transfer Complete interrupt request

The Command Transfer Complete Request indicates the end of transfer of the last SPI frame in a Command Cycle. The Transfer Complete Request is generated for the above condition when RSER[CMDTCF_RE] is set.

43.4.8.7 Transmit FIFO Underflow interrupt request

The Transmit FIFO Underflow request indicates that an underflow condition in the TX FIFO has occurred. The transmit underflow condition is detected only for the DSPI, operating in slave mode and SPI configuration. The TFUF bit is set when the TX FIFO of a DSPI is empty, and a transfer is initiated from an external SPI master. If the TFUF bit is set while the TFUF_RE bit in the RSER is set, an interrupt request is generated.

43.4.8.8 Receive FIFO Drain interrupt or DMA Request

The Receive FIFO Drain Request indicates that the RX FIFO is not empty. The Receive FIFO Drain Request is generated when the number of entries in the RX FIFO is not zero, and the RFDF_RE bit in the RSER is set. The bit RSER[RFDF_DIRS] selects whether a DMA request or an interrupt request is generated.

43.4.8.9 Receive FIFO Overflow Interrupt Request

The Receive FIFO Overflow Request indicates that an overflow condition in the RX FIFO has occurred. A Receive FIFO Overflow request is generated when RX FIFO and shift register are full and a transfer is initiated. The bit RSER[RFOF_RE] must be set for the interrupt request to be generated.

Depending on the state of the bit MCR[ROOE], the data from the transfer that generated the overflow is either ignored or shifted in to the shift register. If the ROOE bit is set, the incoming data is shifted in to the shift register. If the ROOE bit is cleared, the incoming data is ignored.

43.4.8.10 SPI Frame Parity Error Interrupt Request

The SPI Frame Parity Error Flag indicates that a SPI frame with parity error had been received. The SPEF_RE bit in the RSER must be set for the interrupt request to be generated.

43.4.9 Power saving features

The DSPI supports following power-saving strategies:

- External Stop mode
- Module Disable mode – Clock gating of non-memory-mapped logic

43.4.9.1 Stop mode (External Stop Mode)

The DSPI supports the stop mode protocol. When a request is made to enter external stop mode, the DSPI block acknowledges the request. If a serial transfer is in progress, the DSPI waits until it reaches the frame boundary before it is ready to have its clocks shut off. While the clocks are shut off, the DSPI memory-mapped logic is not accessible. This also puts the DSPI in STOPPED state. The SR[TXRXS] bit is cleared to indicate STOPPED state. The states of the interrupt and DMA request signals cannot be changed while in External Stop mode.

43.4.9.2 Module disable mode

Module disable mode is a block-specific mode that the DSPI can enter to save power. The host CPU can initiate the module disable mode by setting MCR[MDIS]. The module disable mode can also be initiated by hardware. A power management block can initiate the module disable mode by asserting the DOZE mode signal while MCR[DOZE] is set.

When MCR[MDIS] is set or the DOZE mode signal is asserted while MCR[DOZE] is set, the DSPI negates Clock Enable signal at the next frame boundary. Once the Clock Enable signal is negated, DSPI is said to have entered Module Disable Mode. This also puts the DSPI in STOPPED state. The SR[TXRXS] bit is cleared to indicate STOPPED state. If implemented, the Clock Enable signal can stop the clock to the non-memory-mapped logic. When Clock Enable is negated, the DSPI is in a dormant state, but the memory-mapped registers are still accessible. Certain read or write operations have a different effect when

the DSPI is in the module disable mode. Reading the RX FIFO Pop Register does not change the state of the RX FIFO. Likewise, writing to the FIFO Push Register does not change the state of the TX FIFO or CMD FIFO. Clearing of the FIFOs has no effect in the module disable mode. Changes to MCR[DIS_TXF] and MCR[DIS_RXF] have no effect in the module disable mode. In the module disable mode, all status bits and register flags in the DSPI return the correct values when read, but writing to them has no effect. Writing to the TCR during module disable mode has no effect. Interrupt and DMA request signals cannot be cleared while in the module disable mode.

43.5 Initialization/application information

This section describes how to initialize the DSPI module.

43.5.1 Managing DSPI queues

The queues are not part of the DSPI, but the DSPI includes features in support of queue management. Queues are primarily supported in SPI configuration.

1. When DSPI executes last command word from a queue, the EOQ bit in the command word is set to indicate to the DSPI that this is the last entry in the queue.
2. At the end of the transfer, corresponding to the command word with EOQ set is sampled, the EOQ flag (EOQF) in the SR is set.
3. The setting of the EOQF flag disables serial transmission and reception of data, putting the DSPI in the STOPPED state. The TXRXS bit is cleared to indicate the STOPPED state.
4. The DMA can continue to fill TX FIFO until it is full or step 5 occurs.
5. Disable DSPI DMA transfers by disabling the DMA enable request for the DMA channel assigned to TX FIFO (and CMD FIFO) and RX FIFO. This is done by clearing the corresponding DMA enable request bits in the DMA Controller.
6. Ensure all received data in RX FIFO has been transferred to memory receive queue by reading the RXCNT in SR or by checking RFDF in the SR after each read operation of the POPR.
7. Modify DMA descriptor of TX and RX channels for new queues
8. Flush TX FIFO (and CMD FIFO) by writing a ‘1’ to MCR[CLR_TXF]. Flush RX FIFO by writing a ‘1’ to MCR[CLR_RXF].
9. Clear transfer count either by setting CTCNT bit in the command word of the first entry in the new queue or via CPU writing directly to TCR[SPI_TCNT].
10. Enable DMA channel by enabling the DMA enable request for the DMA channel assigned to the DSPI TX FIFO, (and CMD FIFO) and RX FIFO by setting the corresponding DMA set enable request bit.
11. Enable serial transmission and serial reception of data by clearing the EOQF bit.

43.5.2 Switching master and slave mode

When changing modes in the DSPI, follow the steps below to guarantee proper operation.

1. Halt the DSPI by setting MCR[HALT].
2. Clear the transmit and receive FIFOs by writing a ‘1’ to the CLR_TXF and CLR_RXF bits in MCR.
3. Set the appropriate mode in MCR[MSTR] and enable the DSPI by clearing MCR[HALT].

43.5.3 Initializing DSPI in Master/Slave Modes

Once the appropriate mode in MCR[MSTR] is configured, the DSPI is enabled by clearing MCR[HALT]. It should be ensured that DSPI Slave is enabled before enabling DSPI Master. This ensures the Slave is ready to be communicated with, before Master initializes communication.

43.5.4 Baud rate settings

Table 722 shows the baud rate that is generated based on the combination of the baud rate prescaler PBR and the baud rate scaler BR in the CTARs. The values calculated assume a 100 MHz protocol frequency and the double baud rate DBR bit is clear.

Note: The protocol clock frequency mentioned above is given as an example in this chapter. Refer to the Chapter 24: Clocking for the frequency used to drive this module in the device.

Table 722. Baud rate values (bps)

Baud Rate scaler values (DSPI*n*_CTAR[BR])	Baud rate divider prescaler values (DSPI*n*_CTAR[PBR])				
		2	3	5	7
	2	25.0 MHz	16.7 MHz	10.0 MHz	7.14 MHz
	4	12.5 MHz	8.33 MHz	5.00 MHz	3.57 MHz
	6	8.33 MHz	5.56 MHz	3.33 MHz	2.38 MHz
	8	6.25 MHz	4.17 MHz	2.50 MHz	1.79 MHz
	16	3.12 MHz	2.08 MHz	1.25 MHz	893 kHz
	32	1.56 MHz	1.04 MHz	625 kHz	446 kHz
	64	781 kHz	521 kHz	312 kHz	223 kHz
	128	391 kHz	260 kHz	156 kHz	112 kHz
	256	195 kHz	130 kHz	78.1 kHz	55.8 kHz
	512	97.7 kHz	65.1 kHz	39.1 kHz	27.9 kHz
	1024	48.8 kHz	32.6 kHz	19.5 kHz	14.0 kHz
	2048	24.4 kHz	16.3 kHz	9.77 kHz	6.98 kHz
	4096	12.2 kHz	8.14 kHz	4.88 kHz	3.49 kHz
	8192	6.10 kHz	4.07 kHz	2.44 kHz	1.74 kHz
	16384	3.05 kHz	2.04 kHz	1.22 kHz	872 Hz
	32768	1.53 kHz	1.02 kHz	610 Hz	436 Hz

43.5.5 Delay settings

Table 723 shows the values for the Delay after Transfer (t_{DT}) and CS to SCK Delay (t_{CSC}) that can be generated based on the prescaler values and the scaler values set in the CTARs. The values calculated assume a 100 MHz protocol frequency.

Note: The protocol clock frequency mentioned above is given as an example in this chapter. Refer to the Chapter 24: Clocking for the frequency used to drive this module in the device.

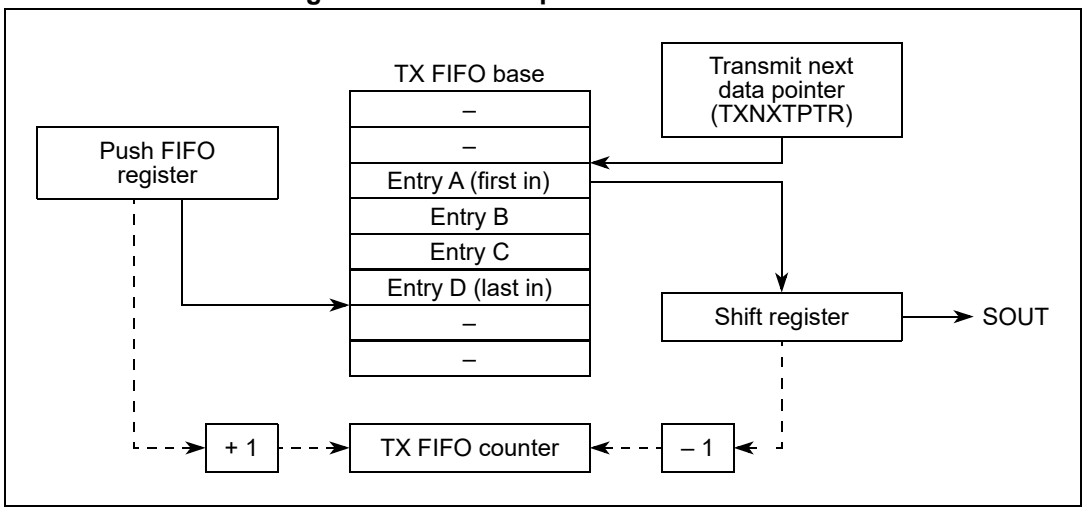
Table 723. Delay values

		Delay prescaler values (DSPI*n*_CTAR[PBR])			
		1	3	5	7
Delay scaler values	2	20.0 ns	60.0 ns	100.0 ns	140.0 ns
	4	40.0 ns	120.0 ns	200.0 ns	280.0 ns
	8	80.0 ns	240.0 ns	400.0 ns	560.0 ns
	16	160.0 ns	480.0 ns	800.0 ns	1.1 µs
	32	320.0 ns	960.0 ns	1.6 µs	2.2 µs
	64	640.0 ns	1.9 µs	3.2 µs	4.5 µs
	128	1.3 µs	3.8 µs	6.4 µs	9.0 µs
	256	2.6 µs	7.7 µs	12.8 µs	17.9 µs
	512	5.1 µs	15.4 µs	25.6 µs	35.8 µs
	1024	10.2 µs	30.7 µs	51.2 µs	71.7 µs
	2048	20.5 µs	61.4 µs	102.4 µs	143.4 µs
	4096	41.0 µs	122.9 µs	204.8 µs	286.7 µs
	8192	81.9 µs	245.8 µs	409.6 µs	573.4 µs
	16384	163.8 µs	491.5 µs	819.2 µs	1.1 ms
	32768	327.7 µs	983.0 µs	1.6 ms	2.3 ms
	65536	655.4 µs	2.0 ms	3.3 ms	4.6 ms

43.5.6 Calculation of FIFO pointer addresses

Complete visibility of the TX and CMD and RX FIFO contents is available through the FIFO registers, and valid entries can be identified through a memory-mapped pointer and a memory-mapped counter for each FIFO. The pointer to the first-in entry in each FIFO is memory-mapped. For the TX FIFO the first-in pointer is the Transmit Next Pointer (TXNXTPTR). For the CMD FIFO the first-in pointer is the Command Next Pointer (CMDNXTPTR). For the RX FIFO the first-in pointer is the Pop Next Pointer (POPNXTPTR). *Figure 677* illustrates the concept of first-in and last-in FIFO entries along with the FIFO Counter. The TX FIFO is chosen for the illustration, but the concepts carry over to the CMD FIFO and RX FIFO. See *Section 43.4.2.4: Transmit First In First Out (TX FIFO) buffering mechanism*, *Section 43.4.2.5: Command First In First Out (CMD FIFO) buffering mechanism* and *Section 43.4.2.6: Receive First In First Out (RX FIFO) Buffering Mechanism* for details on the FIFO operation.

Figure 677. TX FIFO pointers and counter



43.5.6.1 Address calculation for the first-in entry and last-in entry in the TX FIFO

The memory address of the first-in entry in the TX FIFO is computed by the following equation:

Equation 28
First-in EntryAddress = TXFIFOBase + (4 × TXNXTPTR)

The memory address of the last-in entry in the TX FIFO is computed by the following equation:

Equation 29
Last-in EntryAddress = TXFIFOBase + 4 × (TXCTR + TXNXTPTR – 1)mod(TXFIFOdepth)

TX FIFO Base – Base address of TX FIFO

TXCTR – TX FIFO Counter

TXNXTPTR – Transmit Next Pointer

TX FIFO Depth – Transmit FIFO depth, implementation specific

43.5.6.2 Address calculation for the first-in entry and last-in entry in the CMD FIFO

The memory address of the first-in entry in the CMD FIFO is computed by the following equation:

Equation 30
First-in EntryAddress = CMDFIFOBase + (4 × CMDNXTPTR)

The memory address of the last-in entry in the CMD FIFO is computed by the following equation:

Equation 31
Last-in EntryAddress = CMDFIFOBase + 4 × (CMDCTR + CMDNXTPTR – 1)mod(CMDFIFOdepth)

CMD FIFO Base – Base address of CMD FIFO

CMDCTR – CMD FIFO Counter

CMDNXTPTR – Command Next Pointer

CMD FIFO Depth – Command FIFO depth, implementation specific

43.5.6.3 Address calculation for the first-in entry and last-in entry in the RX FIFO

The memory address of the first-in entry in the RX FIFO is computed by the following equation:

Equation 32
First-in EntryAddress = RX FIFOBase + (4 × POPNXTPTR)

The memory address of the last-in entry in the RX FIFO is computed by the following equation:

Equation 33
Last-in EntryAddress = RX FIFOBase + 4 × (RXCTR + POPNXTPTR – 1)mod(RXFIFOdepth)

RX FIFO Base – Base address of RX FIFO

RXCTR – RX FIFO counter

POPNXTPTR – Pop Next Pointer

RX FIFO Depth – Receive FIFO depth, implementation specific

44 LINFlexD

44.1 Introduction

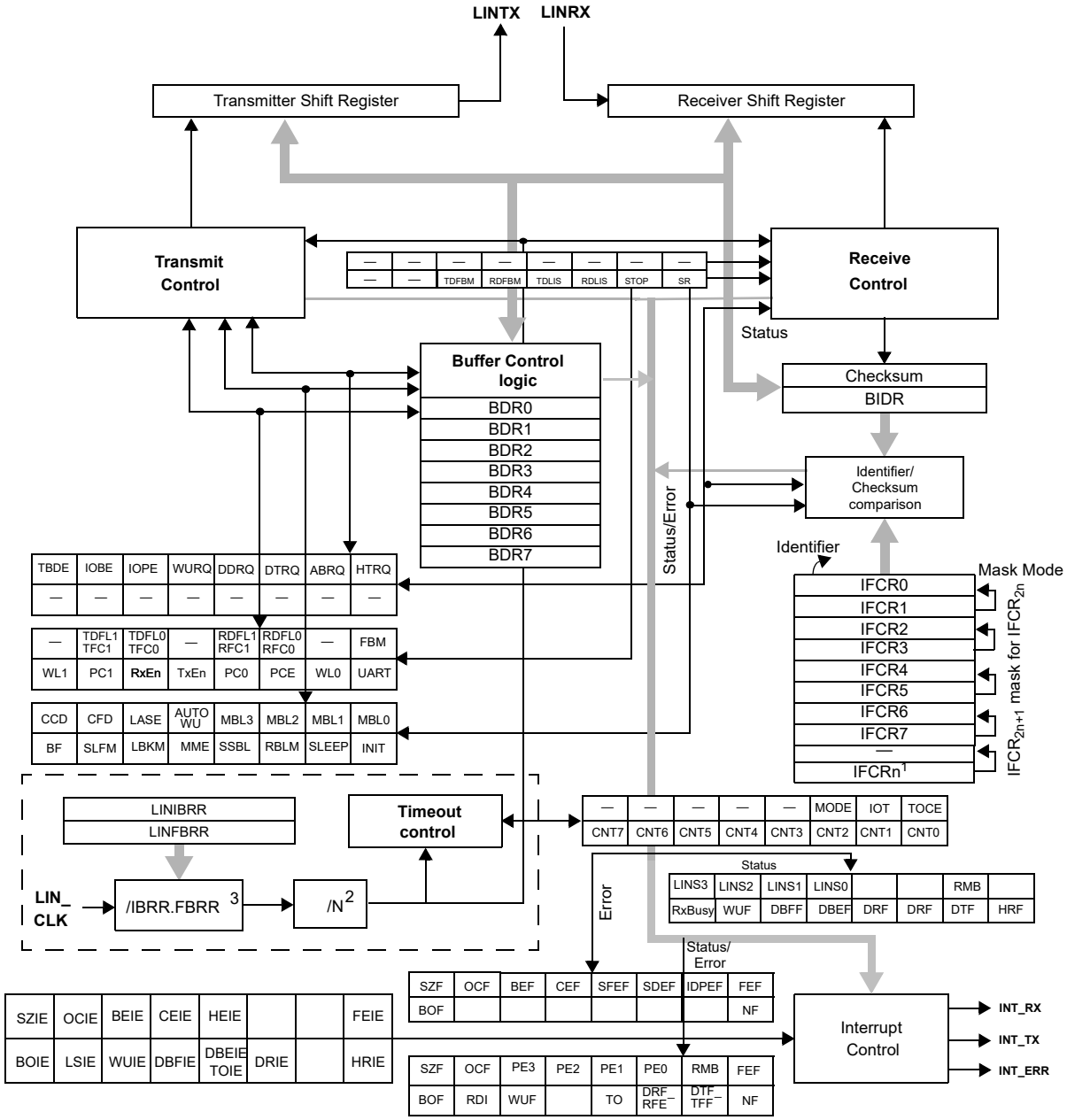
The LINFlexD controller is designed to manage a high number of LIN messages efficiently with a minimum of CPU load. To reduce the CPU loading in Master mode, LINFlexD autonomously handles the LIN messages once software has triggered the header transmission, until the next header transmission request in transmitter mode or until checksum reception in the receiver mode.

The LINFlexD supports LIN protocol version 1.3, 2.0, 2.1 and 2.2. It also consists of an 8-byte buffer for transmission/reception data.

The LINFlexD also provides support for some of the basic UART transfers of 8-bit, 9-bit, 16-bit, and 17-bit frames and also 12-bit data frame + parity reception in UART mode for MSC support.

The LINFlexD supports multi-channels and parametric DMA Tx/Rx interface in LIN/UART operating mode.

Figure 678. Block diagram



Notes:

1. The value of n depends on the no_of_filters. Refer to the *Chapter 7: Device configuration* to see the number of filters used in this device.
2. For LIN mode N = 16 and N is programmable by user in UART mode.
3. FBRR is not usable when reduced over-sampling is enabled in UART mode.

44.2 Main features

The LINFlexD controller can operate in several modes, each of which has a distinct set of features which are described in the following sections. In addition, the LINFlexD controller has several features common to all modes:

- Fractional baud rate generator
- 3 operating modes for power saving and configuration registers lock
  - Initialization
  - Normal
  - Sleep
- Test Mode: Loop Back
- Maskable interrupts
- A maximum of 16 possible identifiers can be programmed into the identifier list

44.2.1 LIN mode features

- Supports LIN protocol version 1.3, 2.0, and 2.1
- Bit rates up to 20 Kbit/s (LIN protocol)
- Master/Slave mode
- Classic and Enhanced Checksum calculation and check
- Single 8-byte buffer or FIFO for Transmission/Reception
- Timeout management
- Identifier filters
- DMA interface
- Supports a maximum of 16 possible identifiers
- Master mode with autonomous message handling
- Extended frame mode for in-application programming purposes
- Wakeup event on dominant bit detection
- True LIN field state machine
- Advanced LIN error detection
- Header, response, and frame timeout
- Slave mode
  - Autonomous header handling
  - Autonomous transmit/receive data handling
- Identifier filters for autonomous message handling in Slave mode
- Separate clock for baud rate calculation (The relationship “(2/3)* LIN_CLK $\geq$ PBRIDGEx_CLK > 1/3*LIN_CLK” should be maintained)

44.2.2 UART mode features

- Full-duplex communication
- Baud rate is a function of baud clock, LINIBRR and LINFBRR registers (see *Section 44.3.2.8: Baud rate generation*)
- Separate clock for baud rate calculation (The relationship “(2/3)* LIN_CLK ≥ PBRIDGEx_CLK > 1/3*LIN_CLK” should be maintained)
- 15/16/7/8 bits data, parity
- 1/2/3 stop bits
- 12-bit + parity reception
- 4-byte buffer for reception, 4-byte buffer for transmission
- 12-bit counter for timeout management
- The maximum baud rate achievable is 25 Mbit/s.
- For bit rate ≤ 6.25 Mbit/s
  - Sixteen times oversampling
  - 3:1 majority voting
- For 6.25 Mbit/s < bit rate ≤ 12.5 Mbit/s
  - Reduced over sampling programmable by the user
  - 3:1 majority voting for reduced over sampling of 8
- For 12.5 Mbit/s < bit rate ≤ 25 Mbit/s
  - Reduced over sampling programmable by the user
  - 1:1 voting for all reduced over sampling of 4, 5 and 6

44.2.3 Standard features

DMA2x bus interface

44.3 Functional description

44.3.1 LIN protocol

The LIN (Local Interconnect Network) is a serial communication protocol. A LIN cluster consists of one master task and several slave tasks. A master node contains the master task as well as a slave task - all other nodes contain a slave task only. The master node decides when and which frame is transferred on the bus. The slave task provides the data to be transported by the frame.

44.3.1.1 Frames

A frame consists of a header provided by the master task and a response provided by the slave task. The header consists of a break, a sync pattern, and an identifier. The break is followed by the sync pattern and the sync pattern is followed by the identifier. The slave task associated with the identifier provides the response. The response consists of a data field and checksum. The slave task designated to receive the data associated with the identifier receives the response and verifies the checksum.

Figure 679. Frames

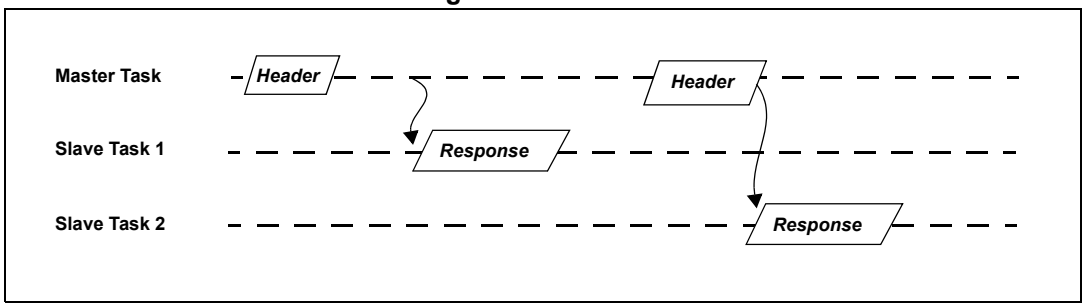
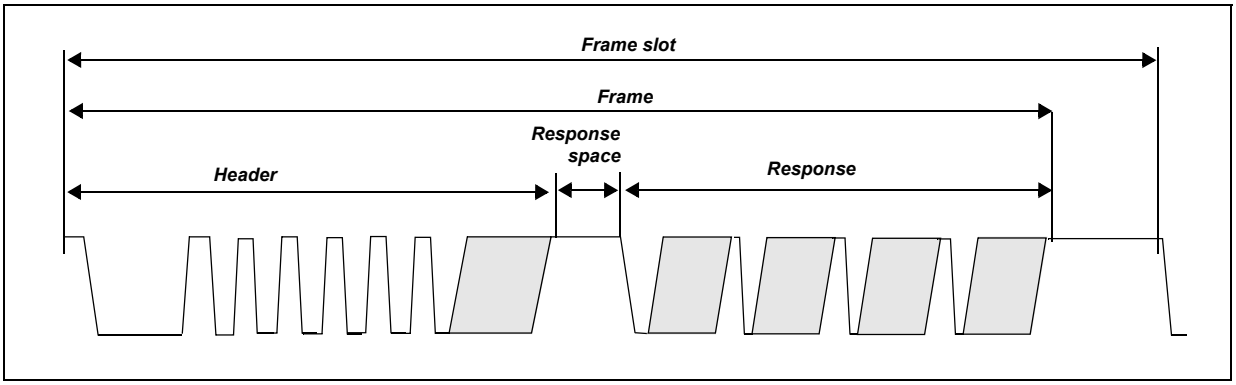


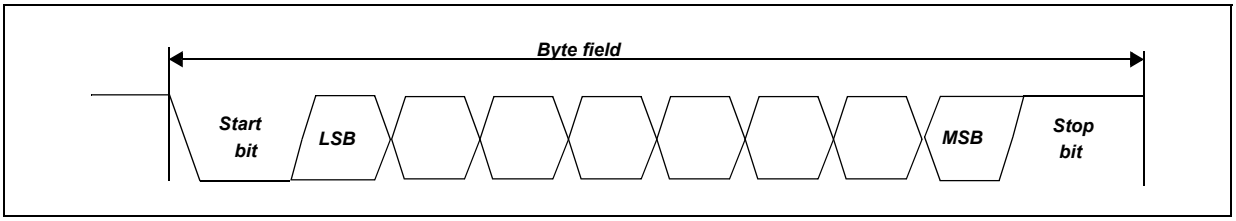
Figure 680. Structure of LIN frame



44.3.1.2 Data field

Each byte is transmitted as shown in *Figure 681*. The LSB of the data is sent first and the MSB is sent last. The start bit is encoded as a bit with value zero (dominant) and stop bit is encoded as bit value one (recessive).

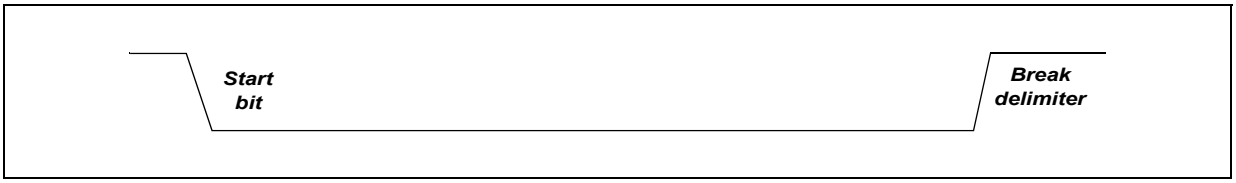
Figure 681. Structure of byte field



44.3.1.3 Break

The break symbol is used to signal the beginning of a new frame. It is the only field that does not comply with the above figure. The break is always generated by the master and shall be at least 13 bits of dominant value including the start bit, followed by a break delimiter as shown in *Figure 682*. The break delimiter must be of two-bit duration (to be compliant with LIN protocol 2.1).

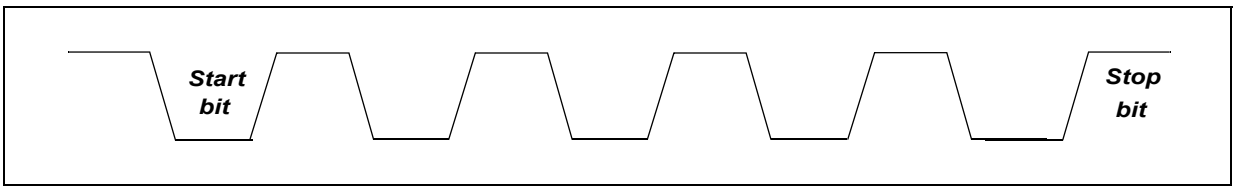
Figure 682. Break field



44.3.1.4 Sync byte

Sync is a byte field with the data value of 0x55.

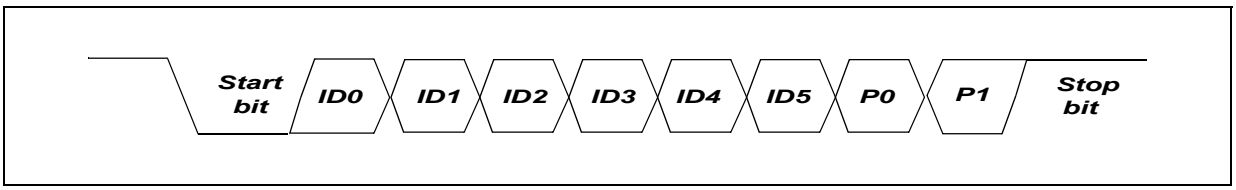
Figure 683. Sync byte field



44.3.1.5 Identifier

The Identifier field consists of two sub fields, the identifier and the identifier parity. Bits 0 to 5 are the identifier and bits 6 and 7 indicate the parity.

Figure 684. Identifier field



P0 = ID0 XOR ID1 XOR ID2 XOR ID4

P1 = NOT (ID1 XOR ID3 XOR ID4 XOR ID5)

44.3.1.6 Checksum

The last field of a frame is the checksum. The checksum contains the inverted 8-bit sum with carryover of all data bytes or all data bytes and the identifier. Checksum calculation over the data bytes only is called classic checksum and is used for communication with LIN 1.3 slaves. Checksum calculation over the data bytes and the protected identifier byte is called enhanced checksum, and it used for communication with LIN 2.0 slaves.

44.3.2 LINFlexD features

44.3.2.1 Operating modes

The LINFlexD has three operating modes: **Initialization, Normal,** and **Sleep** modes. After hardware reset, the LINFlexD enters sleep mode to reduce power consumption.

44.3.2.1.1 Initialization mode (INIT)

To enter this mode, software sets the INIT bit in the LINCR1. To exit the initialization mode, software should reset the INIT bit.

When in initialization mode, all message transfers to and from the LIN bus are stopped and the status of LIN bus output LINTX is recessive (high). If software invokes the initialization mode when a bus transfer is in progress, the transfer is aborted. The software should therefore check the LIN state before setting this bit.

To initialize the LINFlexD controller, software must:

1. Set up the baud rate registers
2. Reset the UART bit
3. Select the mode (Master or Slave)
4. Configure checksum control bits
5. Initialize the identifier list (Slave mode)

44.3.2.1.2 Normal mode (NM)

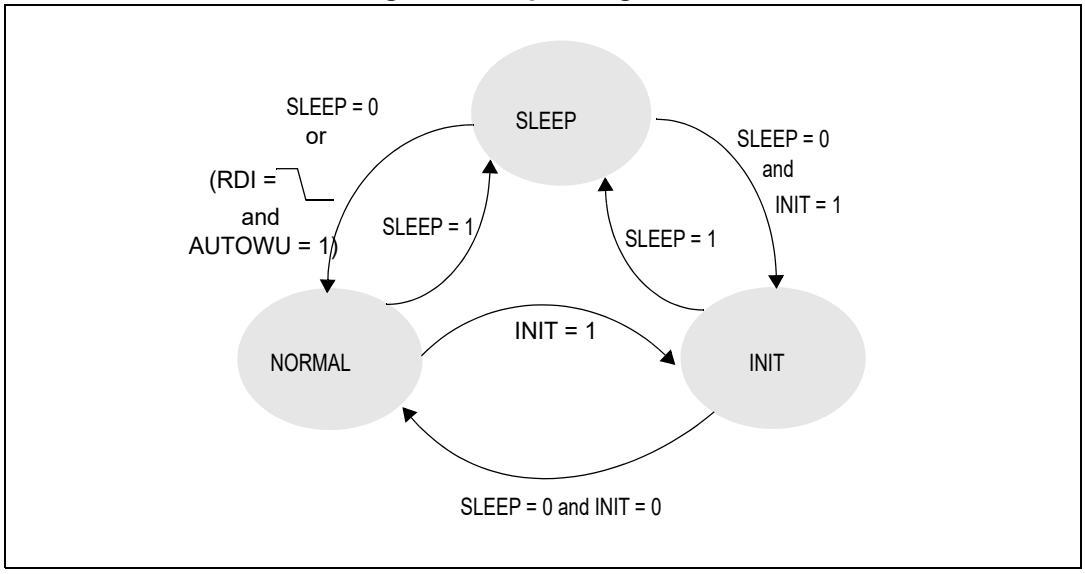
Once software has completed initialization of the LINFlexD controller, it can enter Normal mode by clearing the INIT bit.

44.3.2.1.3 Sleep mode (SM)

Sleep mode in LINFlexD reduces power consumption. This mode is entered by setting the SLEEP bit in LINCR1. In this mode the LINFlexD clock is stopped. LINFlexD can be woken from Sleep mode by clearing the SLEEP bit.

If software detects a wakeup pulse of 150 µs on the LIN Bus, it can request LINFlexD to wake up from Sleep mode. Refer to *Section 44.3.2.10: Wakeup management*.

Figure 685. Operating modes



44.3.2.2 Test mode

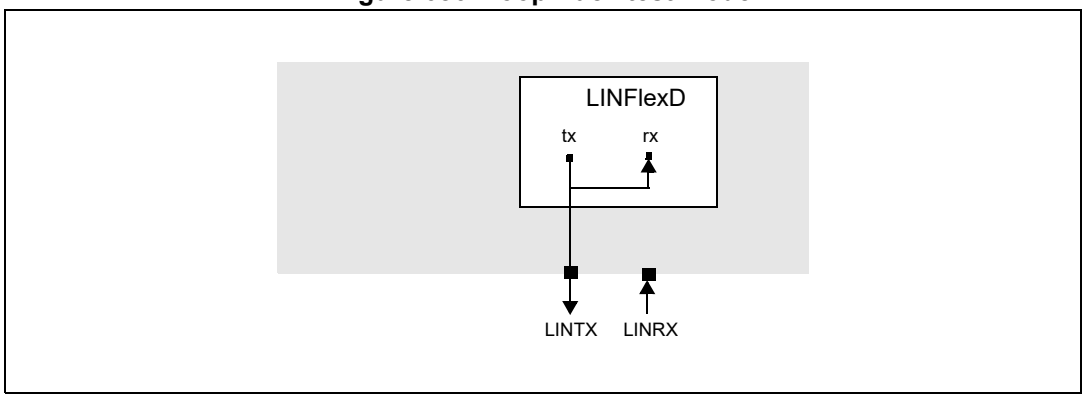
44.3.2.2.1 Loop Back mode

This mode is entered by setting the LBKM bit in LINCR1. In this mode, the LINFlexD receives the Identifier and Data transmitted by itself and writes the same in the BIDR and

Data buffers respectively. This mode is provided for selftest functions. Bit error is checked in this mode.

The LINFlexD ignores the LINRX signal. There is internal feedback from its TX output to its RX input. The TX pin can be disconnected from LINTX pin by SIUL2 setting.

Figure 686. Loop Back test mode



44.3.2.3 Master mode

Master mode is selected by means of the MME bit in LINCR1 register.

44.3.2.3.1 Header transmission

According to the LIN protocol any communication on the LIN bus is triggered by the master sending a header. The header is transmitted by the master task of a node while the response is transmitted by the slave task of a node.

To transmit the header, first program the Identifier, data field length, message direction and checksum enable in the BIDR register; then set the HTRQ bit in LINCR2. Once header transmission starts, the user should not modify BIDR register bits until the current frame is complete. The transmitted ID is also received by the master node and copied to BIDR.

44.3.2.3.2 Data transmission

When the master node is the publisher of the data corresponding to the Identifier sent by the master, then the slave task of the node should send the data in the response part of the frame. Hence software must provide the data to the LINFlexD before the header transmission is requested. The data to be transmitted is stored in the message buffer BDR[0:7]. The number of bytes to be transmitted depends on the Data Field length in BIDR. The software uses the BIDR[CCS] bit to configure the checksum type (classic or enhanced) for each message.

If the response has been sent successfully, LINSR[DTF] is set. In case of error, the DTF flag is not set and the corresponding error flag is set in the LINESR (refer to error handling). It is possible to handle frames with a response size larger than eight bytes of data (extended frames). If the data field length in the BIDR is configured with a value higher than eight data bytes, LINSR[DBEF] is set once the first eight bytes have been transmitted. The application has to update the buffer BDR before resetting the DBEF bit. The transmission of the next bytes starts when the DBEF bit is reset. Once the last data byte (or the checksum byte) has been sent, the DTF flag is set.

The direction of the message buffer is decided by the DIR bit in the BIDR. The transmitted data is also received by the same node and copied to the buffer.

44.3.2.3.3 Data reception

To receive data from a slave node, the master sends a header with the corresponding Identifier. The data received from the slave is stored in the message buffer and the status of the message is stored in the LINSR.

If the response has been received successfully, the LINSR (DRF) bit is set. In case of error, the DRF flag is not set and the corresponding error flag is set in the LINESR (refer to error handling). It is possible to handle frames with a Response size larger than eight bytes of data (extended frames). If the data field length in the BIDR is configured with a value higher than eight data bytes, the LINSR (DBFF) bit is set once the first eight bytes have been received. The application has to read the buffer BDR before resetting the DBFF bit. Once the last data byte (or the checksum byte) has been received, the DRF flag is set.

44.3.2.3.4 Data Discard

If the user wants to discard the data after header transmission, then the DDRQ bit in LINCR2 should be set.

44.3.2.4 Slave mode

This mode is selected when the MME bit of the LINCR1 register is cleared.

44.3.2.4.1 Data transmission

On header reception, the HRF bit is set and an RX interrupt is generated. The software must then:

1. Read the received ID in the BIDR register
2. Fill the BDR[0:x] register
3. Program the CCS and DIR bits in the BIDR register
4. Specify the data field length DFL[5:0] bits in the BIDR register
5. Trigger the data transmission by setting the DTRQ bit

Note that the HRF bit should be reset only after the DTRQ bit is set. For the DTRQ to be effective, the HRF bit must be set. This is to ensure that DTRQ is not set randomly, but only after a header reception. It must be noted that you cannot set the DIR and DTRQ bits once RXbusy is asserted in LINSR.

Alternately, one or more Identifier filters are configured for transmission by setting the DIR bit and activated by setting the enable bits in IFER. When at least one Identifier filter is active and configured for transmission and the received ID matches the filter, a TX interrupt is generated. The software can use the index in the IFMI register to point directly to the corresponding data array in the RAM and copy this data to the BDR[0:7]. The use of a filter saves software the processing time required to read the ID value in the BIDR, match it, and configure the data field length and checksum type.

If the number of filters provided by LINFlexD are not sufficient for the application, mask mode can be used for the filters.

The transmitted data is also received by the same node and copied to the buffer.

44.3.2.4.2 Data reception

When LINFlexD is the subscriber of the data of the received identifier, then on header reception the HRF flag is set and an RX interrupt is generated. The software must read the received ID from the BIDR register and specify the data field length before the reception of the stop bit of the first data byte. When the checksum is received, an RX interrupt is generated for software to read the received data from BDR[0:7] and the RMB bit is set. Software must then release the data buffer by resetting the RMB bit in the LINSR.

When at least one identifier filter is active and configured for reception, an RX interrupt is generated only after the checksum reception. No interrupt request is generated on reception of the ID.

If the Identifier is filtered by software, you can discard the data by setting the DDRQ bit in the LINCR2, while HRF is set.

Note that for software filtering, software must decide the type of checksum (configure the CCS bit of the BIDR register) before reception of data. Otherwise the previous value of the CCS bit is considered while calculating checksum.

44.3.2.5 Errors

44.3.2.5.1 Header error

A header error is an error during header reception. The types are:

- Sync Del error (SDEF)
- Sync field error (SFEF)
- Identifier Parity error (IDPEF)

1. Sync Delimiter too short

The delimiter should be 1 for at least one bit time, otherwise it is considered short and consequently the receiver discards synchronization on the current header. Hence the frame is discarded. An interrupt is generated if the HEIE bit of LINIER is set.

2. Inconsistent Sync field

This error condition is monitored differently depending on whether autosynchronization is ON or OFF.

Case 1: Autosynchronization ON (LASE bit of LINCR1 = 1):

When Autosynchronization is enabled, the inconsistent Sync Field informs:

- That the deviation error on the Sync Field is outside the LIN specification, which allows up to 14% of period deviation between the slave and master oscillators,

  or

- An overflow has occurred during the Sync Field Measurement, which leads to an overflow of the divider registers.

Deviation error on the Sync Field

The deviation error is checked by comparing the current baud rate (relative to the slave oscillator) with the received LIN Sync Field (relative to the master oscillator). Two checks are performed in parallel:

1. The first check is based on a measurement between the first falling edge and the last falling edge of the Sync Field. Let's refer to this period deviation as D.

   If there is an inconsistent Sync Field error, it means that:

   D > 14.0625%

   If there is no error, it means that:

   D < 14.84375%

   If 14.0625% < D < 14.84375%, then the Sync Field could be either consistent or inconsistent depending on dephasing between the signal on the RDI line and the LIN_CLK.

2. The second check is based on the measurement of each bit time between both edges of the Sync Field. This checks that each of these bit times is large enough (more than 12 samples) compared to the bit time of the current baud rate.

Case 2: Autosynchronization OFF (LASE bit of LINCR1 = 0)

In this case the Sync character is received as a normal character. If the received character is 0x55, then the Sync Field is OK.

On any occurrence of an inconsistent Sync Field, the receiver immediately exits from Sync_Field state and the frame is consequently discarded. The SFEF bit of LINESR is set.

1. Identifier Parity error

There are two parity bits in the identifier field. These bits are checked against the hardware-calculated parity over the other six bits of identifier, according to the formula:

P0 = ID0 XOR ID1 XOR ID2 XOR ID4

P1 = NOT (ID1 XOR ID3 XOR ID4 XOR ID5)

This parity is checked after the ID is transferred to the BIDR register (after stop of ID has been detected properly) and upon parity mismatch, the receiver state machine exits from Identifier state immediately if the IOPE bit of LINCR2 is set.

44.3.2.5.2 Bit error

This error is flagged in transmission mode when the value read back from the bus is different from the value transmitted. Bit error checking on each bit is guaranteed if transceiver delay is less than one bit time minus 6 LIN_CLK cycles. Bit error is not checked during break field transmission.

Example 1

1-bit time at 20 Kbit/s = 50 µs

6 LIN_CLK cycles at 80 MHz = 75 ns

Thus, (1-bit time) – (6 LIN_CLK cycles) = 49.925 µs

Transmission of the frame is stopped after the corrupted bit if the IOBE bit in LINCR2 is set. If IOBE is reset, the transmitter continues to transmit in spite of the bit error. An interrupt is generated if the BEIER bit is set in LINIER.

Note: If the break delimiter is not detected by the master within one bit time after delimiter transmission due to transceiver delay or error on the bus, then a train of bit error interrupts are generated. In this case, the Identifier may not be replaced in the BIDR.

Similarly, if the start of a falling edge of data is not detected by the transmitter node within one bit time after start bit transmission, then a train of bit error interrupts are generated. In

this case also, data and checksum replacements in the BDR and CFR respectively are not guaranteed.

44.3.2.5.3 Framing error

This error is flagged when a dominant state is sampled on the stop bit of the current received character (sync field, identifier field, data field, checksum field). LINFlexD discards the current frame and returns to Idle state. An interrupt is generated if the FEIE bit is set in LINIER.

The byte that caused framing error is also shifted to the buffer but DRF or DBFF is never set in this case.

44.3.2.5.4 Checksum error

This error is flagged when the checksum computed by hardware does not match the received checksum.

LINFlexD discards the received frame and returns to Idle state. An interrupt is generated if the CEIE bit is set in LINIER.

44.3.2.5.5 Overrun error

Once the message buffer is full (RMB is set), the next valid message reception will lead to an overrun and the message will be lost. The hardware signals the overrun condition by setting the BOF bit. Which message is lost depends on the buffer lock function control bit RBLM.

If RBLM is cleared, the old message in the buffer is overwritten by the most recent message. If RBLM bit is set, the most recent message is discarded and the oldest message is available to the software. In the case of a slave, if the buffer is not released (RMB is not reset) before reception of the next Identifier and if RBLM is set, then the ID along with the data is discarded.

44.3.2.5.6 Timeout error

Figure 687. Incomplete response (for example, missing checksum)

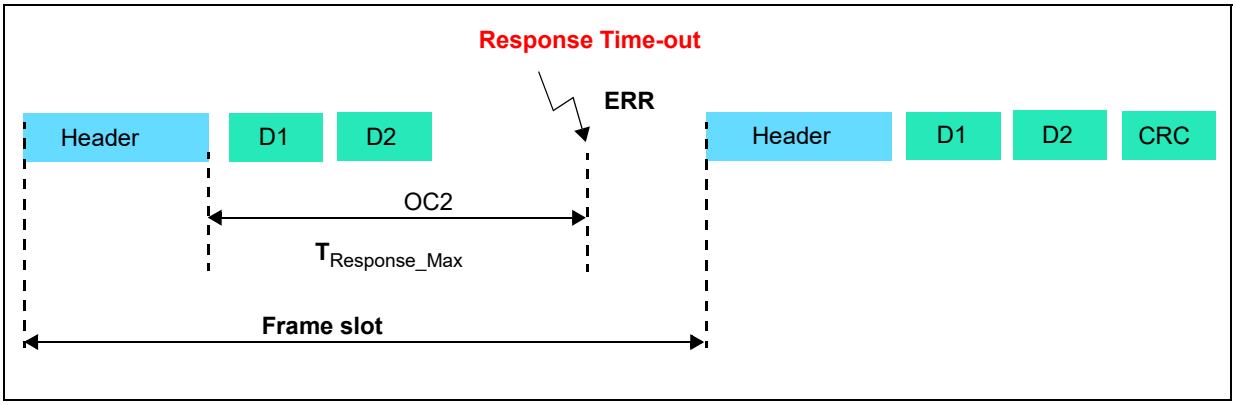
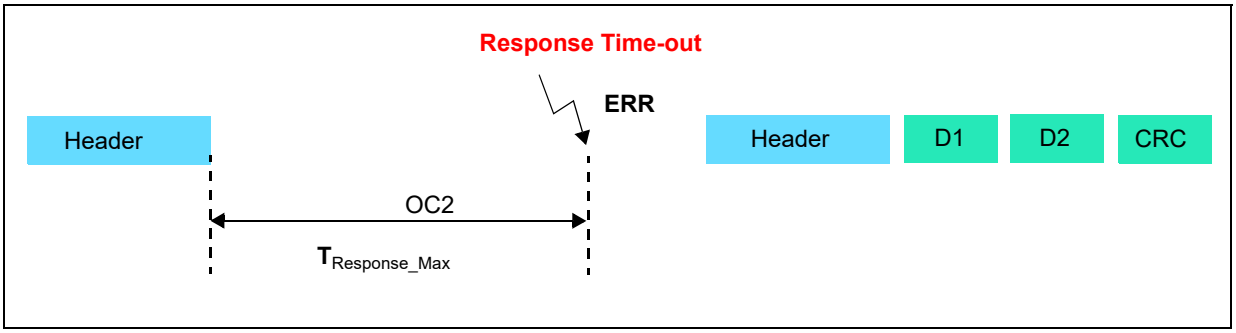


Figure 688. No response



Header timeout mechanism

- Master mode
  - ocr_2 is loaded with Nominal_time_out-Correction_factor at the end of the stop bit of the Identifier field (where nominal_time_out = 1.4 × ((DFL + 2) × 10 bit time)).
  - The change with respect to earlier implementation is that this loading takes place at the end of the ID field (and not at the beginning of the data field). Moreover, the correct value, which depends on DFL, is loaded immediately. No further ocr_2 update is required.
  - In case of no response at all within this time (in other words, start of data is not received), timeout takes place when the counter reaches the ocr_2 value (no change compared to earlier implementation).
  - In case of an incomplete response, timeout also occurs when the counter reaches the ocr_2 value (no change compared to earlier implementation).
- Slave mode
  - **Case 1**: The received identifier is managed by a filter. The implementation can be similar to the master mode, because the DFL value is loaded by hardware. Thus, ocr_2 can be loaded with Nominal_time_out-Correction_factor at the end of the stop bit of the Identifier field (where nominal_time_out = 1.4 × ((DFL + 2) × 10 bit time)).
  - **Case 2**: The received identifier is not managed by a filter. As the DFL value needs to be updated by software after the identifier field has been received, the previous strategy cannot be used as the DFL value is not yet available. The earlier implementation loaded the maximum value = 1.4 × (9 × 10 bit time) and then updated it according to DFL after the first data byte reception. This was not acceptable if no data byte was received at all, as timeout expired too late. The proposed implementation is the following:

    At the end of the ID, ocr_2 is loaded with 36 (maximum possible response space)

    At the end of the first_data_byte it is reloaded again according to DFL

    Before reloading, LINFlexD checks the count_val. If count value is higher than the value to be reloaded, timeout takes place immediately and no reloading occurs.

Header timeout mechanism

- Master mode: As the header is generated by the LINFlexD, there are only two cases:
  - No error on the bus and timing is correct (nominal header length).
  - An error occurs on the bus and LINFlexD flags it in LINESR register (typically a bit error).

    Therefore there is no meaning of header timeout in master mode, so it is disabled.
- Slave mode
  - Header_nominal = 13 + 2 + 10 + 10 = 35 Tbit
  - Header_max = 1.4 × Header_nominal = 49 Tbit
  - Taking into account a possible 14% clock deviation, header_max seen by LINFlexD is 49 × 1.14 = 56 Tbit.
  - If the counter starts after 11 Tbit, the HTO value is 56 – 11 = 45 Tbit

    The reset value of HTO is 45 and this register can only be programmed in slave mode.

    As response space is now included in the response, frame timeout is no longer needed and is removed completely. Indeed, header timeout and response timeout cover all cases.

    The timeout counter can be used to detect other timeouts. In this case, the MODE bit must be set and the output compare value can be updated in the LINTOCR register by software.

Stuck at zero timeout error

If the dominant pulse lasts for a time of at least 100 bits, the SZF bit in LINESR is set. If the same dominant pulse prolongs, the subsequent SZF setting will be 87 bit times apart (instead of 100 bit times).

44.3.2.5.7 Noise

During reception each bit is sampled 16 times and the value of the bit is obtained by taking the majority value of the 8th, 9th, and 10th samples. If any one of these three samples has a value different from the other two, this error is flagged.

When OSR = 8, majority of 2nd, 3rd, and 4th samples are taken into account to determine noise. Noise checking is disabled for all OSR less than eight.

This error is flagged when there is noise detected in the start bit (see *Section 44.4.2.4: LIN Error Status Register (LINESR)*).

44.3.2.6 Identifier filtering

In LIN protocol the identifier of a message is not associated with the address of a node but is related to the content of the message. A transmitter broadcasts its message to all the receivers. Based on the header received, the receiver decides whether to receive or transmit a response (depending on the identifier value). If the message does not target the node, it should be discarded.

To fulfill this requirement, the LINFlexD provides configurable filters in order to eliminate software intervention. This hardware filtering saves CPU resources that would otherwise be required to perform filtering by software.

There are a maximum of 16 filters (depending on the generic no_of_filters) in the LINFlexD, which can be programmed by the user only during Initialization mode. In order to activate a filter the corresponding FACT bit in IFER needs to be set. There are two modes possible for each identifier depending on the corresponding IFM bit of IFMR.

44.3.2.6.1 Identifier list mode

If the n^{th} bit of IFMR is cleared, then filter number 2*n* and 2*n*+1 is in identifier list mode. In this mode, the maximum number of filters that can be configured for transmission/reception equals no_of_filters, depending on the FACT bit of IFER. In this mode the identifier received should match bit by bit to the ID field of IFCR2*n* or IFCR2*n*+1 (if the corresponding FACT bit is set).

44.3.2.6.2 Identifier mask mode

If the number of filters required is more than no_of_filters, then filters should be configured in Mask mode. If the n^{th} bit of IFMR is set, then filter number 2*n* is the filter and 2*n*+1 acts as a mask for it. The FACT bit for filter 2*n*+1 has no effect in this case. In this mode, if the x^{th} bit of the mask is set, then the x^{th} bit of the received identifier must match the x^{th} bit of filter.

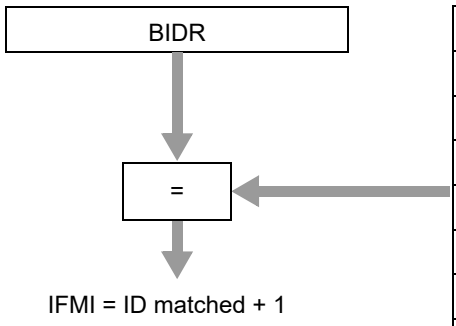
If there is a match of identifier with the *m*th filter in any mode, then *m*+1 is loaded in the IFMI register by hardware. No match condition is denoted by IFMI = 0.

Upon matching DFL (2:0), the CCS and DIR bits of the BIDR register are copied from the filter by hardware and from then on, the BIDR register is read-only until the end of the frame. Now if the DIR bit of BIDR is set, then a TXI interrupt is generated if the HRIE bit of LINIER is set. In this case, software uses the IFMI register to transfer the relevant data from the RAM area to BDR, and after a complete transfer the DTRQ bit of LINCR2 is set to start the transmission. If the DIR bit is cleared then an RXI interrupt is generated (provided DRIE bit of LINIER is set) when the checksum has been received and there is no checksum error.

In case of no filter match condition (IFMI = 0) and if the BF bit of LINCR1 is set, then an RXI is generated. Now it is the responsibility of software to configure BIDR and start transmission (by loading the BDR buffer and setting the DTRQ bit of LINCR2) or discard reception (by setting the DDRQ bit of LINCR2). If the BF bit is reset, then the receiver discards the received identifier and turns to Idle state in search of a new break.

Note: If one identifier matches with two filters (one is in the list and the other in Mask mode) then List mode prevails over Mask mode. In Mask mode, if two filters match with the identifier then the filter having the lower number prevails.

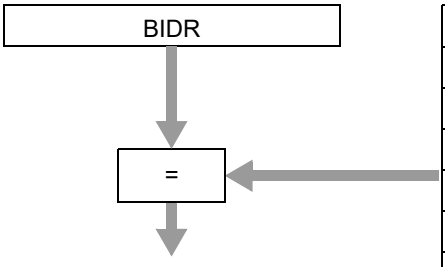
Figure 689. Identifier List mode



IFCR0	ID5	ID4	ID3	ID2	ID1	ID0
IFCR1	ID5	ID4	ID3	ID2	ID1	ID0
IFCR2	ID5	ID4	ID3	ID2	ID1	ID0
IFCR3	ID5	ID4	ID3	ID2	ID1	ID0
IFCR4	ID5	ID4	ID3	ID2	ID1	ID0
IFCR5	ID5	ID4	ID3	ID2	ID1	ID0
IFCR6	ID5	ID4	ID3	ID2	ID1	ID0
IFCR7	ID5	ID4	ID3	ID2	ID1	ID0
...	ID5	ID4	ID3	ID2	ID1	ID0
IFCRn[1]	ID5	ID4	ID3	ID2	ID1	ID0

1. The value of n depends on the no_of_filters. Refer to the *Chapter 7: Device configuration* to see the number of filters used in this device.

Figure 690. Identifier Mask mode



IFCR0	ID5	ID4	ID3	ID2	ID1	ID0
IFCR1	0	0	0	1	1	1
IFCR2	ID5	ID4	ID3	ID2	ID1	ID0
IFCR3	1	1	1	1	0	0
IFCR4	ID5	ID4	ID3	ID2	ID1	ID0
IFCR5	1	1	1	1	0	1
IFCR6	ID5	ID4	ID3	ID2	ID1	ID0
IFCR7	1	1	1	1	1	0
...	...	...	...	...	...	...
IFCRn[1]	...	...	...	...	...	...

$IFCR_{2n+1}$ = Mask for $IFCR_{2n}$
1 = must match bits
0 = don't care bits

1. The value of n depends on the no_of_filters. Refer to the *Chapter 7: Device configuration* to see the number of filters used in this device.

44.3.2.7 Start detection and delimiter detection in receiver

There is a 10-bit-shift-register sample register in the receiver that shifts signal data to the next less significant bit on each incoming sample.

Table 724. 10-bit-shift-register sample register

Sample register bit number										Counter
9	8	7	6	5	4	3	2	1	0	
—	—	—	—	—	—	—	—	—	RT1	counter = 0
—	—	—	—	—	—	—	—	RT1	RT2	counter = 1
—	—	—	—	—	—	—	RT1	RT2	RT3	counter = 2
—	—	—	—	—	—	RT1	RT2	RT3	RT4	counter = 3
—	—	—	—	—	RT1	RT2	RT3	RT4	RT5	counter = 4
—	—	—	—	RT1	RT2	RT3	RT4	RT5	RT6	counter = 5
—	—	—	RT1	RT2	RT3	RT4	RT5	RT6	RT7	counter = 6
—	—	RT1	RT2	RT3	RT4	RT5	RT6	RT7	RT8	counter = 7
—	RT1	RT2	RT3	RT4	RT5	RT6	RT7	RT8	RT9	counter = 8
RT1	RT2	RT3	RT4	RT5	RT6	RT7	RT8	RT9	RT10	counter = 9
RT2	RT3	RT4	RT5	RT6	RT7	RT8	RT9	RT10	RT11	counter = 10
RT3	RT4	RT5	RT6	RT7	RT8	RT9	RT10	RT11	RT12	counter = 11
RT4	RT5	RT6	RT7	RT8	RT9	RT10	RT11	RT12	RT13	counter = 12
RT5	RT6	RT7	RT8	RT9	RT10	RT11	RT12	RT13	RT14	counter = 13
RT6	RT7	RT8	RT9	RT10	RT11	RT12	RT13	RT14	RT15	counter = 14
RT7	RT8	RT9	RT10	RT11	RT12	RT13	RT14	RT15	RT16	counter = 15
1	1	1	0	x(1)	0	x	0	x	0	Start detection
1	1	1	0	x	0	x	0	x	1	
1	1	1	0	x	0	x	1	x	0	
1	1	1	0	x	1	x	0	x	0	
0	0	0	1	x	1	x	1	x	1	Delimiter detection
0	0	0	1	x	1	x	1	x	0	
0	0	0	1	x	1	x	0	x	1	
0	0	0	1	x	0	x	1	x	1	

1. x: due to sampling during transition of the bit, the bit value might be sampled as a ‘0’ or ‘1’ depending on the rise/fall time. So ‘x’ refers to an indeterministic value.

44.3.2.7.1 Start detection mechanism

The following steps are followed for detecting a start bit:

1. Refer to *Table 724* for the status of the sample register (shift register) when count = 6.
2. At this point, if RT1 (the first incoming sample) = 0 and the previous three samples already received are all ones, then it might be a start bit.
3. To make sure it is indeed a start bit, the incoming data samples in the sample register (4), sample register (2) and sample register (0) which correspond to RT3, RT5, and RT7 is verified using the steps mentioned below.
4. If the majority of these 3 samples are equal to ‘0’ (2 or more), then start bit is said to be detected.
5. These three samples RT3, RT5, and RT7 are called the “verification samples” which are checked at count = 6.
6. The sample register bits 9, 8, 7, and 6 are called “qualification samples” which are checked at count = 6.
7. At count = 9, if majority value of RT8, RT9, and RT10 is not equal to ‘0’ then Noise flag is set.

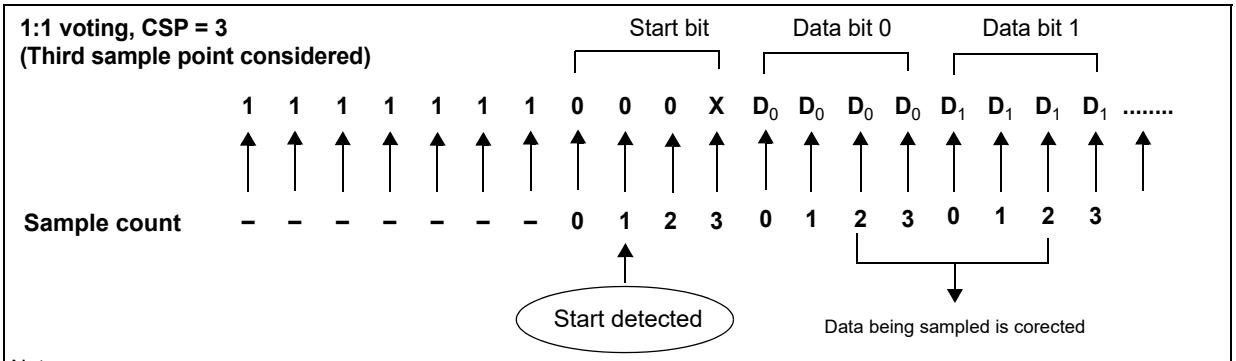
44.3.2.7.2 Delimiter detection mechanism

The following steps are followed for detecting a delimiter bit:

1. Refer to *Table 724* for the status of the sample register (shift register) when count = 6.
2. At this point, if RT1(the first incoming sample) = 1 and the previous three samples already received are all zeros, then it might be a delimiter bit.
3. To make sure it is indeed a delimiter bit, the incoming data samples in the sample register (4), sample register (2) and sample register (0) which correspond to RT3, RT5, and RT7 is verified using the steps mentioned below.
4. If the majority of these 3 samples are equal to ‘1’ (2 or more), then delimiter bit is said to be detected.
5. These three samples RT3, RT5, and RT7 are called the “verification samples” which are checked at count = 6.
6. The sample register bits 9, 8, 7, and 6 are called “qualification samples” which are checked at count = 6.

Hence, a start is detected as soon as it is qualified with 1110 in the sample register and verified with at least two out of three predefined verification samples being zero. Similarly for delimiter detection, qualification samples are 0001 and at least two out of the three verification samples are one. The Noise Flag is set if the start is verified with only two valid samples.

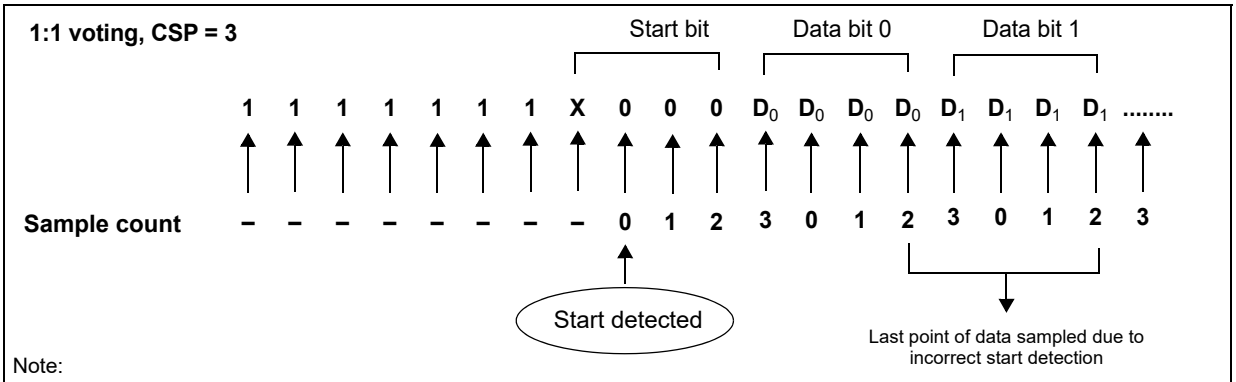
Figure 691. Start detection and sampling for over sampling rate = 4 (Case 1)



Note:

1. D_0, D_1: Databits
2. X: Due to sampling during transition of the bit, value might be sampled as a ‘0’ or ‘1’ depending on the rise/fall time.Therefore ‘X’ refers to a indeterministic value.

Figure 692. Start detection and sampling for over sampling rate = 4 (Case 2a)



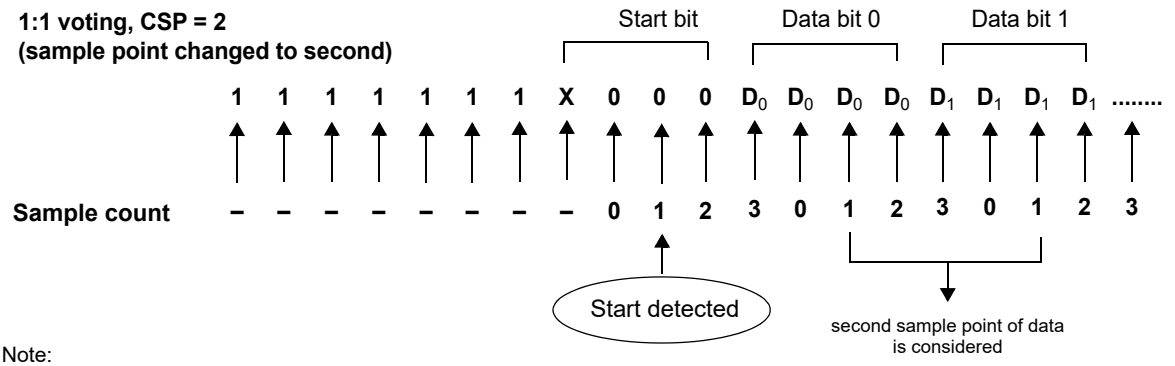
Note:

1. D_0, D_1: Databits
2. X: Due to sampling during transition of the bit, value might be sampled as a ‘0’ or ‘1’ depending on the rise/fall time. Therefore ‘X’ refers to a indeterministic value.

The choice of the correct sample points depends on the external Rx signal quality. During the application development process, the error information indicated by the parity error can help to find the best setting for an application-specific hardware signal.

To sample at the correct point therefore, CSP has to be changed to two; resulting in the following situation:

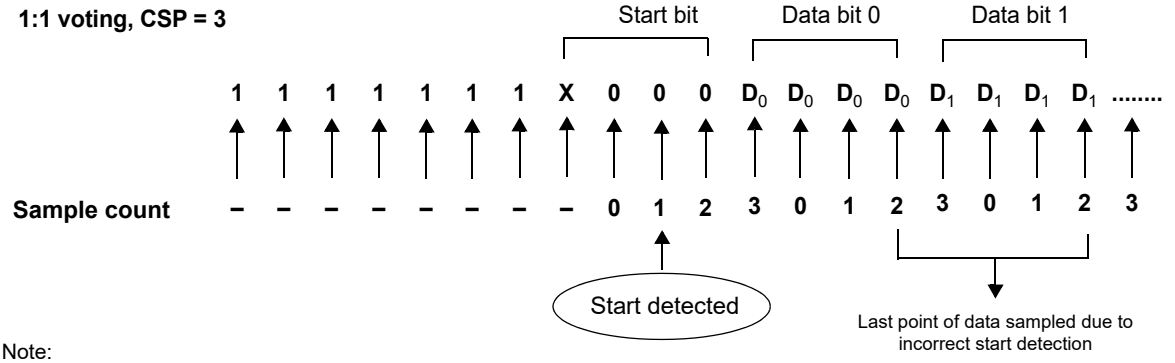
Figure 693. Start detection and sampling for over sampling rate = 4 (Case 2b)



Note:

1. D_0, D_1: Databits
2. X: Due to sampling during transition of the bit, value might be sampled as a ‘0’ or ‘1’ depending on the rise/fall time.

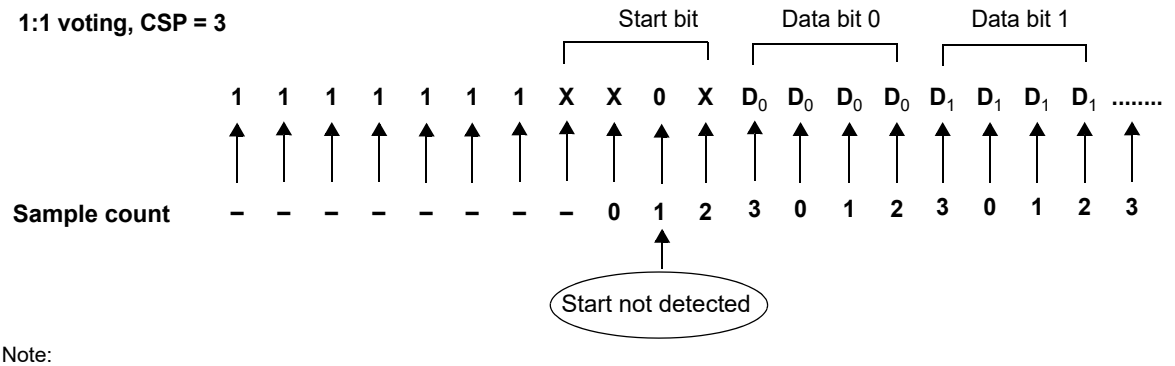
Figure 694. Start detection and sampling for over sampling rate = 4 (Case 3)



Note:

1. D_0, D_1: Databits
2. X: Due to sampling during transition of the bit, value might be sampled as a ‘0’ or ‘1’ depending on the rise/fall time.
3. Therefore ‘X’ refers to a non deterministic value.

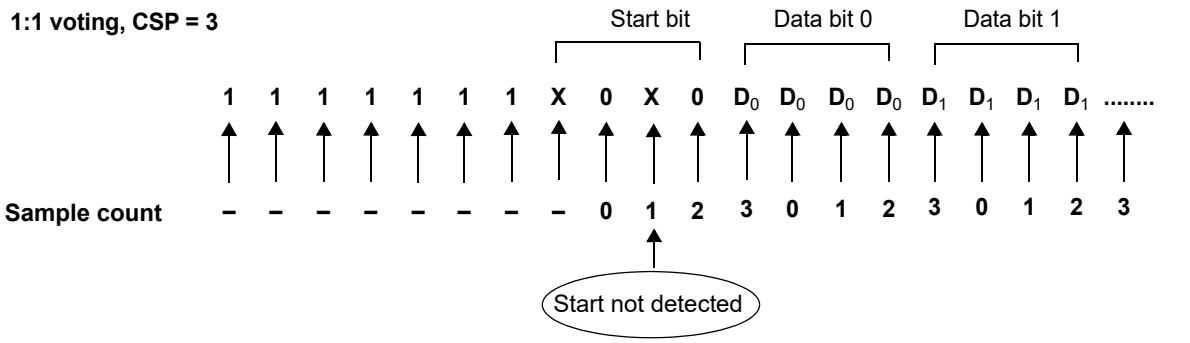
Figure 695. Start detection and sampling for over sampling rate = 4 (Case 4)



Note:

1. D_0, D_1: Databits
2. X: Due to sampling during transition of the bit, value might be sampled as a ‘0’ or ‘1’ depending on the rise/fall time.
3. Therefore ‘X’ refers to a non deterministic value.

Figure 696. Start detection and sampling for over sampling rate = 4 (Case 5)



Note:

1. D_0, D_1: Databits
2. X: Due to sampling during transition of the bit, value might be sampled as a '0' or '1' depending on the rise/fall time.
3. Therefore 'X' refers to a non deterministic value.

44.3.2.8 Baud rate generation

The LIN baud rate is programmed in two registers: the LIN Integer Baud Rate register and the LIN Fraction Baud Rate register. The baud rate registers can be programmed only during Initialization mode.

The baud rate is calculated with the following formula for both receiver and transmitter:

Equation 34 Tx = Rx = LIN_CLK / (16 × LDIV)

Where LIN_CLK is the frequency of the baud clock (ipg_baud_clk).

LDIV is an unsigned fixed point number. The mantissa is coded into 20 bits of LINIBRR and the fraction is coded on 4 bits of LINFBRR.

When reduced oversampling is enabled LINFBRR is not used and LDIV contains only the integer part of LINIBRR.

Example 2

When ROSE = 0 (For LIN and UART mode):

LDIV = 1041.67, LIN_CLK = 80 MHz

LINIBRR = 1041d = 411 h

LINFBRR = 11d = B h

Baud rate = 80 MHz/(16 x 1041,67) = 4.8 Kbit/s

Example 3

When ROSE = 1 (Only for UART mode):

LDIV = 10 d, LIN_CLK = 80 MHz

LINIBRR = 10 d, OSR = 4

Baud rate = LIN_CLK / (OSR × LDIV) = 80 MHz / (4 × 10) = 2 Mbit/s

Note: Due to design constraints the maximum baud rate achievable is 25 Mbit/s.

The relationship "(2/3)* LIN_CLK ≥ PBRIDGEx_CLK > 1/3*LIN_CLK" should be maintained)

44.3.2.9 Automatic resynchronization

To automatically adjust the baud rate based on measurement of the LIN sync field, write the nominal Prescaler value (nominal baud rate) in LINIBRR and LINFBRR, then set the LASE bit in LINCR1 to enable automatic synchronization.

When automatic synchronization is enabled, after each LIN Sync Del, the time duration between five falling edges on RDI is sampled on LIN_CLK.

44.3.2.10 Wakeup management

Any node in a sleeping LIN cluster may request a wakeup. The wakeup request is issued by forcing the bus to the dominant state for 250 µs to 5 ms. Every slave node should detect the wakeup request (a dominant pulse longer than 150 µs) and be ready to listen to bus commands within 100 ms, measured from the ending edge of the dominant pulse. The master also wakes on detecting a wakeup request and when the slaves are ready, starts sending frame headers to find out the cause of the wakeup. If the master does not issue a frame header within 150 ms from the wakeup request, then the node issuing a request may try issuing a new wakeup request.

In LINFlexD, a wakeup request can be generated by writing the wakeup character in BDR0 and setting the WURQ bit in LINCR2. On setting the WURQ bit, the character in BDR0 is transmitted. For LIN 2.0, character 0xF0 is sent as the wakeup character.

In LINFlexD, wakeup can be detected in two ways.

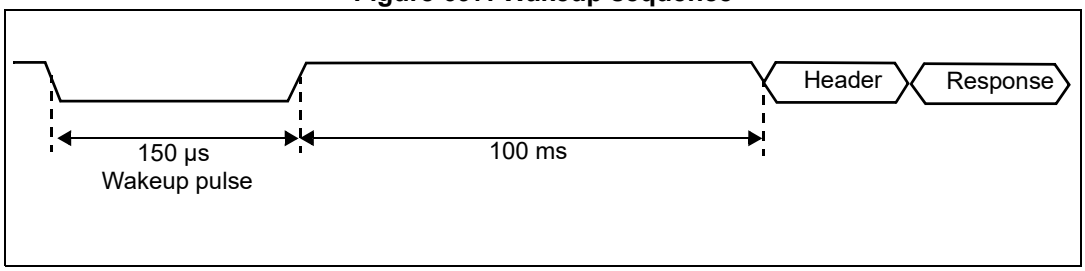
1. AUTOWU = 1

   On detecting a falling edge in sleep mode, the SLEEP bit is cleared by hardware, the WUF flag is set, and an interrupt is generated (if WUPIE is set). LINFlexD is now in normal mode and ready to receive frames.

2. AUTOWU = 0

   On detecting a falling edge the WUF flag is set and an interrupt is generated (if WUPIE is set). It is then up to the software to clear the SLEEP bit.

Figure 697. Wakeup sequence



44.3.2.11 Use case and limitations

In LIN mode, when IP is requested to enter STOP mode, the SW has to first set the INIT bit to send the receiver to IDLE. This ensures that the IP acknowledges STOP request.

44.3.3 Timer

There is an 8 bit counter which has different behavior in different modes.

In Output Compare Mode (LINTCSR[MODE]= 1) : This counter is running even in Sleep and Initialization mode. The counter value which when matches the two software configurable

output compare registers (LINOCR[OC1] or LINOCR[OC2]), generate output compare interrupt (LINESR[OCF]) provided TOCE bit in LINTCSR is set. Once an interrupt occurs subsequent interrupt is generated only when the application has soft reset the LIN IP through GCR [SR] bit to get reset the output compare flag (LINESR[OCF]).

In LIN Mode: The software has no control over the TOCE bit and output compare registers are utilized for generation of LIN timeout interrupts (header, frame, response times out). In this case, if LIN moves to Sleep or Init state then this counter remains in Reset state. LIN mode has no meaning if UART is enabled, hence the counter will remain in Reset state.

44.3.4 UART mode

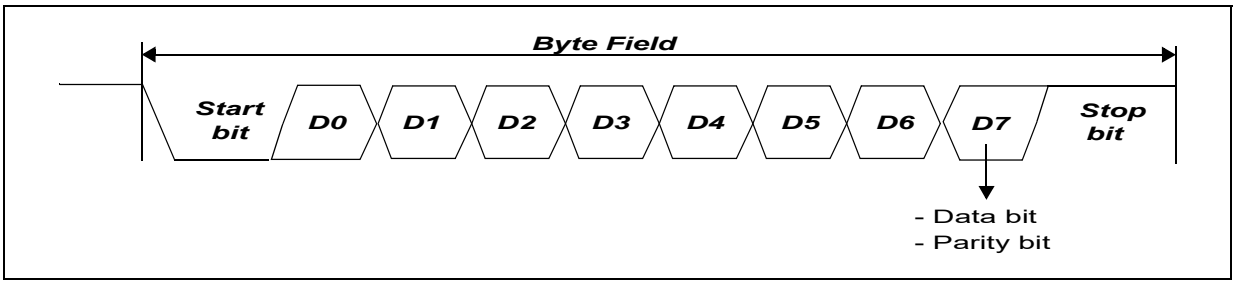
The main features in UART mode are:

- Full duplex communication
- 8-bit frames, 9-bit frames,16-bit frames,17-bit frames
- Even/Odd/0/1 Parity
- User programmable over sampling rate to obtain the baud rate of up to 25 Mbit/s

44.3.4.1 8-bit data frames

The eighth bit can be a data bit or a parity bit. Even/Odd/0/1 parity can be selected by the PC[1:0] bit in the same register. An even parity will be set if the modulo-2 sum of the seven data bits is one. An odd parity will be cleared in this case.

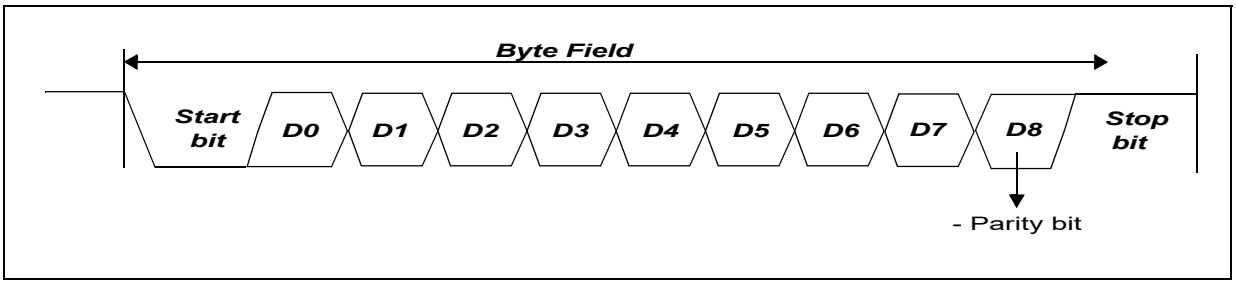
Figure 698. UART mode 8-bit data frame



44.3.4.2 9-bit frames

The ninth bit should be a parity bit. Even/Odd/0/1 Parity can be selected by the PC[1:0] field in the same register. An even parity will be set if the modulo-2 sum of the seven data bits is one. An odd parity will be cleared in this case. Parity 0 forces a zero logical value. Parity 1 forces a high logical value.

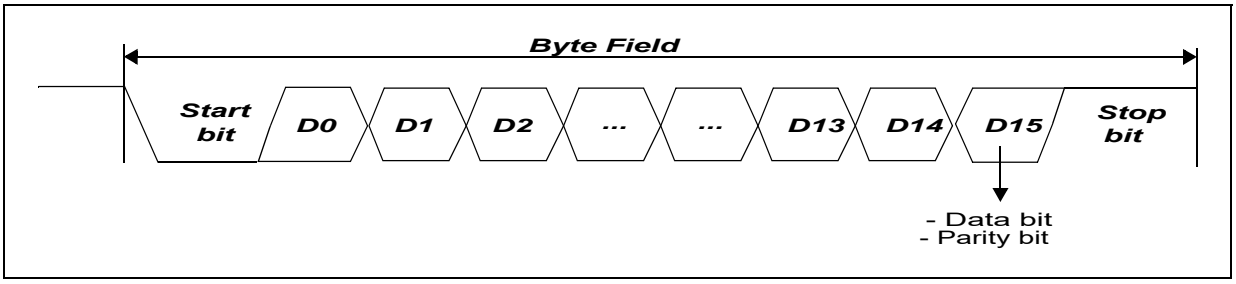
Figure 699. UART mode 9-bit data frame



44.3.4.3 16-bit data frames

The 16th bit can be a data or parity bit. Even/Odd/0/1 Parity bit can be selected by the PC[1:0] bit in the same register. Parity 0 forces a zero logical value. Parity 1 forces a high logical value.

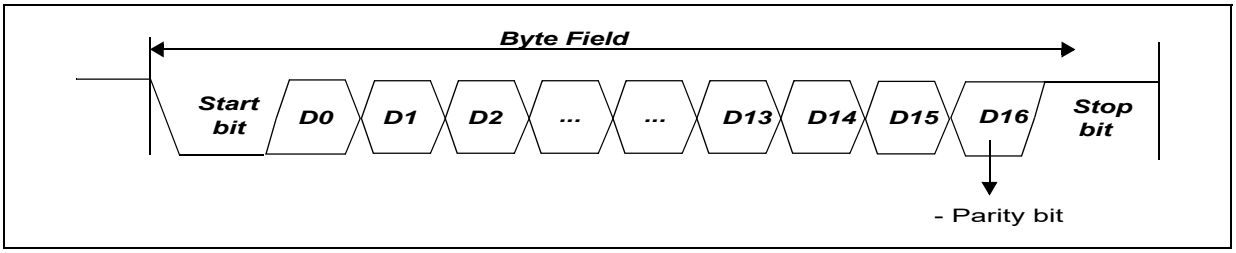
Figure 700. UART mode 16-bit data frame



44.3.4.4 17-bit frames

The 17th bit is the parity bit. Even/Odd/0/1 Parity bit can be selected by the PC[1:0] bit in the same register. Parity 0 forces a zero logical value. Parity 1 forces a high logical value.

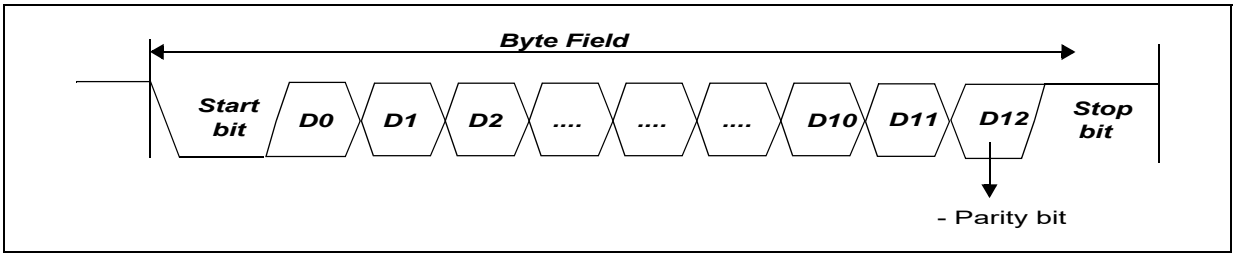
Figure 701. UART mode 17-bit data frame



44.3.4.5 13-bit frames

Whenever WLS is one, special word length is selected in UART mode. This bit enables 12-bit + parity bit reception only in FIFO mode.

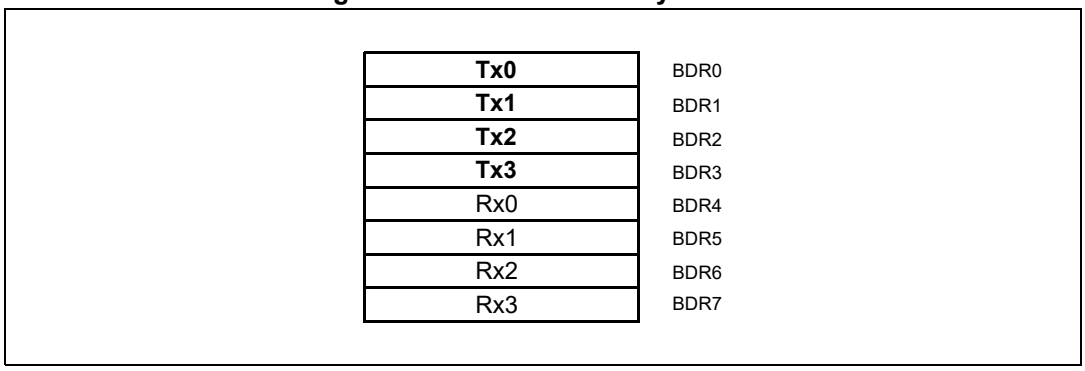
Figure 702. UART mode 13-bit data frame



44.3.4.6 Buffer in UART mode

The 8-byte buffer is divided into two parts: one for receiver and one for transmitter, as shown in *Figure 703*.

Figure 703. Structure of 8-byte buffer



For 16-bit frames, the lower eight bits are written in BDR0 and the upper eight bits in BDR1.

The same applies for reception.

44.3.4.7 UART transmitter

In order to start transmission in UART mode, the UART bit should be set and the transmitter enable bit should be set. Transmission starts when the BDR0 (least significant data byte) is programmed and continues until the number of bytes/halfwords transmitted is equal to the value in the TDFL bits in UARTCR.

The transmit buffer is four bytes (when UARTCR[WL1] = 0) or two halfwords (when UARTCR[WL1] = 1), hence a maximum of four bytes (two halfwords) transmission can be triggered. Once the programmed number of bytes (halfwords) has been transmitted, the DTF flag is set in UARTSR. If the Txen bit of UART is reset in the middle of a transmission, then the current transmission is completed and no further transmissions can be invoked.

The buffer can be configured in FIFO mode (mandatory when the DMA Tx is enabled) by setting the control bit UARTCR[TFBM].

Note: If TFF bit is set and a write is performed to the FIFO, the data transmitted may be erroneous.

Invalid writes to BDRL which generates IPS transfer error, also triggers data transmission at the same time.

When the FIFO is full, a read to DATA0 of BDRM reflects the content of DATA3 in BDRL. Proper functionality of the IP is not affected by this.

Table 725. BDRL access in UART mode (1)

IPS operation	Register	Mode (UARTCR[TFBM])	Word length (UARTCR:WL)	IPS operation result
Write byte0	BDRL	FIFO	Byte	OK
Write byte1-2-3	BDRL	FIFO	Byte	IPS transfer error
Write halfword0-1	BDRL	FIFO	Byte	IPS transfer error
Write word	BDRL	FIFO	Byte	IPS transfer error
Write byte0-1-2-3	BDRL	FIFO	Halfword	IPS transfer error
Write halfword0	BDRL	FIFO	Halfword	OK
Write halfword1	BDRL	FIFO	Halfword	IPS transfer error

Table 725. BDRL access in UART mode (continued)(1)

IPS operation	Register	Mode (UARTCR[TFBM])	Word length (UARTCR:WL)	IPS operation result
Write word	BDRL	FIFO	Halfword	IPS transfer error
Read byte0-1-2-3	BDRL	FIFO	Byte/halfword	IPS transfer error
Read halfword0-1	BDRL	FIFO	Byte/halfword	IPS transfer error
Read word	BDRL	FIFO	Byte/halfword	IPS transfer error
Write byte0-1-2-3	BDRL	Buffer	Byte/halfword	OK
Write halfword0-1	BDRL	Buffer	Byte/halfword	OK
Write word	BDRL	Buffer	Byte/halfword	OK
Read byte0-1-2-3	BDRL	Buffer	Byte/halfword	OK
Read halfword0-1	BDRL	Buffer	Byte/halfword	OK
Read word	BDRL	Buffer	Byte/halfword	OK

1. Halfword0 refers to DATA1 and DATA0 of BDRL register. Halfword1 refers to DATA3 and DATA2 of BDRL register. Halfword2 refers to DATA5 and DATA4 of BDRM register. Halfword3 refers to DATA7 and DATA6 of BDRM register.

In UART FIFO mode (UARTCR[TFBM] = 1), any read operation causes an IPS transfer error.

44.3.4.8 UART receiver

The reception of a data byte is started as soon as the user exits Initialization mode, sets the Rxen bit, and detects the start bit. There is a dedicated 4-byte (if UARTCR[WL1] = 0) or 2-halfword (if UARTCR[WL1] = 1) data buffer for received data. Once the programmed number (RDFL bits) of bytes have been received, the DRF flag is set in UARTSR and the current reception gets completed. Rxen bit needs to be set only to start the reception.The reception is automatically completed as soon as the programmed number (RDFL bits) of bytes have been received.

The buffer can be configured in FIFO mode (mandatory when the DMA Rx is enabled) by setting the control bit UARTCR[RFBM].

Table 726. BDRM access in UART mode(1),(2)

IPS operation	Register	Mode (UARTCR[RFBM])	Word length (UARTCR:WL)	IPS operation result
Read byte4	BDRM	FIFO	Byte	OK
Read byte5-6-7	BDRM	FIFO	Byte	IPS transfer error
Read halfword2-3	BDRM	FIFO	Byte	IPS transfer error
Read word	BDRM	FIFO	Byte	IPS transfer error
Read byte4-5-6-7	BDRM	FIFO	Halfword	IPS transfer error
Read halfword2	BDRM	FIFO	Halfword	OK
Read halfword3	BDRM	FIFO	Halfword	IPS transfer error
Read word	BDRM	FIFO	Halfword	IPS transfer error

Table 726. BDRM access in UART mode(1),(2) (continued)

IPS operation	Register	Mode (UARTCR[RFBM])	Word length (UARTCR:WL)	IPS operation result
Write byte4-5-6-7	BDRM	FIFO	Byte/halfword	IPS transfer error
Write halfword2-3	BDRM	FIFO	Byte/halfword	IPS transfer error
Write word	BDRM	FIFO	Byte/halfword	IPS transfer error
Read byte4-5-6-7	BDRM	Buffer	Byte/halfword	OK
Read halfword2-3	BDRM	Buffer	Byte/halfword	OK
Read word	BDRM	Buffer	Byte/halfword	OK
Write byte4-5-6-7	BDRM	Buffer	Byte/halfword	IPS transfer error
Write halfword2-3	BDRM	Buffer	Byte/halfword	IPS transfer error
Write word	BDRM	Buffer	Byte/halfword	IPS transfer error

1. Refer to the layout of the registers BDRL and BDRM to identify the mapping between byte x and the data bits of the BDRL and BDRM registers.
2. Halfword0 refers to DATA1 and DATA0 of BDRL register. Halfword1 refers to DATA3 and DATA2 of BDRL register.
   Halfword2 refers to DATA5 and DATA4 of BDRM register. Halfword3 refers to DATA7 and DATA6 of BDRM register.

Note: If the user does not know in advance how many bytes are to be received, RDFL should not be programmed in advance. The reset value of RDFL is zero. This will ensure that the reception will happen byte by byte. The state machine will move to the Idle state after each byte reception.

If RDFL is programmed for a certain value but that number bytes are not received, then reception will hang. In that case, software needs to take care of timeout by seeing the flag. Software has to set the sleep bit to move to the Idle state.

If a STOP request arrives in the middle of a reception, it is only acknowledged after all the programmed number of data bytes have been received; it is not served immediately. If the programmed number of data bytes are not received, then software has to take care of timeout; when the state machine moves to Idle state, then only a stop request is served.

If during reception of any byte a parity error occurs, then the corresponding PEx bit in UARTSR is set. No interrupt is generated in this case. If a framing error occurs in any byte (FE bit in UARTSR is set) then an interrupt is generated if the FEIE bit in LINIER is set. As there is only one register bit for a framing error, this interrupt will be helpful in identifying which byte has the framing error.

If the last received frame has not been read from the buffer (in other words, RMB is not reset by the user), then upon reception of the next byte, an overrun error occurs (BOF bit in UARTSR will be set) and one message is lost depending on the RBLM bit of LINCR1. An interrupt is generated if the BOIE bit in LINIER is set.

When WLS bit = 1, BDRM access depends on WL bit setting.

When WLS bit = 1, WL = 0 or 1 should not be used, since this will lead to incorrect reception of data.

In UART FIFO mode, when IP is requested to enter STOP mode, the SW has to first set the INIT bit to send the receiver to IDLE. This ensures that the IP acknowledges STOP request.

44.3.5 DMA interface

44.3.5.1 Main features

The LINFlexD DMA offers a parametric and programmable solution with the following distinctive features:

- LIN Master node, TX mode: single DMA channel
- LIN Master node, RX mode: single DMA channel
- LIN Slave node, TX mode: 1 to N DMA channel where N = max number of ID filters
- LIN Slave node, RX mode: 1 to N DMA channel where N = max number of ID filters
- UART node, TX mode: single DMA channel
- UART node, RX mode: single DMA channel + time-out

This release is backwards-compatible with the existing LINFlexD release if the new features are not enabled.

44.3.5.2 Definitions

Control/status register fields of the LINFlexD macrocell are described in *Table 727* and *Table 728*.

Table 727. LINFlexD control/status fields description

Register	Field	Level	Description
LINCR2	DDRQ	High	Data discard request in reception mode
	DTRQ	High	Data transmission request (slave mode) of the LIN data field stored in BDRL/BDRM
	HTRQ	High	Header transmission request (master mode)
BIDR	DFL[5:0]	—	Data field length: 1 8 bytes: normal frames 1 64 bytes: extended frames
	DIR	—	Direction of the data field: 0 Reception 1 Transmission
	CCS	—	Checksum type: 0 Enhanced 1 Classic
	ID[5:0]	—	Identifier

Table 727. LINFlexD control/status fields description (continued)

Register	Field	Level	Description
LINSR	RMB	—	Buffer data ready to be read via CPU/DMA 0 Buffer data is free 1 Buffer data is ready
	DBFF	High	Data buffer full (8 bytes received) and DFL > 7 Used for extended frames.
	DBEF	High	Data buffer empty (8 bytes have been transmitted) and DFL > 7 Used for extended frames.
	DRF	High	Data reception completed
	DTF	High	Data transmission completed
	HRF	High	Header received Used in slave mode in case of Tx filter match.
IFMI	IFMIx	!= zero	Filter match index (slave mode)
UARTCR	FBM	—	FIFO/Buffer mode 0 Buffer mode 1 FIFO mode (mandatory in DMA mode)
UARTSR	RFE	—	Rx FIFO empty 0 Rx FIFO not empty 1 Rx FIFO empty
	TFF	—	Tx FIFO full 0 Tx FIFO not full 1 Tx FIFO full

DMATXE[2**TX_CH_NUM](1)		High	DMA Tx channel enable (read/write)
			In UART or LIN master mode only channel 0 can be programmed.
			In LIN slave mode: # DMA TX channel = IFMI[3:0] – 1.
			DMATXE[x] = 0b → TX channel x disabled
			DMATXE[x] = 1b → TX channel x enabled

DMARXE[2**RX_CH_NUM](1)		High	DMA Rx channel enable (read/write)
			In UART or LIN master mode only the channel 0 can be programmed.
			In LIN slave mode: # DMA RX channel = IFMI[3:0] – 1.
			DMARXE[x] = 0b → RX channel x disabled
			DMARXE[x] = 1b → RX channel x enabled

1. ** stands for exponentiation

Table 728. LINFlexD DMA control fields description

Field	Mode	Value	Level	Description
DMA_TEN	LIN master Tx or UART Tx	DMATXE[0] & dma_enb_req_tx[0]	High	Logical AND between the two DMA enable bits (LINFlexD and eDMA)
DMA_TEN	LIN slave Tx	DMATXE[x] & dma_enb_req_tx[x] x = IFMI – 1	High	
DMA_REN	LIN master Rx or UART Rx	DMARXE[0] & dma_enb_req_rx[0]	High	
DMA_REN	LIN slave Rx	DMARXE[x] & dma_enb_req_rx[x] x = IFMI – 1	High	

Control/status fields of the Tranfer Control Descriptors (TCDs) referred to in this document are described in *Table 729*.

Table 729. TCD control fields description

TCD field	Level	Description
CITER[14:0]	—	Current major iteration count
BITER[14:0]	—	Beginning major iteration count
NBYTES[31:0]	—	Inner minor byte transfer count Number of bytes to be transferred in each service request of the channel.
SADDR[31:0]	—	Source address
SOFF[15:0]	—	Source address signed offset Applied to the current source address as each source read is completed.
SSIZE[2:0]	—	Source data transfer size 000 8-bit 001 16-bit 010 32-bit 011 64-bit
SLAST[31:0]	—	Last source address adjustment
DADDR[31:0]	—	Destination address
DOFF[15:0]	—	Destination address signed offset Applied to the current destination address as each destination write is completed.
DSIZE[2:0]	—	Destination data transfer size 000 8-bit 001 16-bit 010 32-bit 011 64-bit
DLAST_SGA[31:0]	—	Last destination address adjustment or the memory address for the next TCD to be loaded into this channel (scatter/gather)

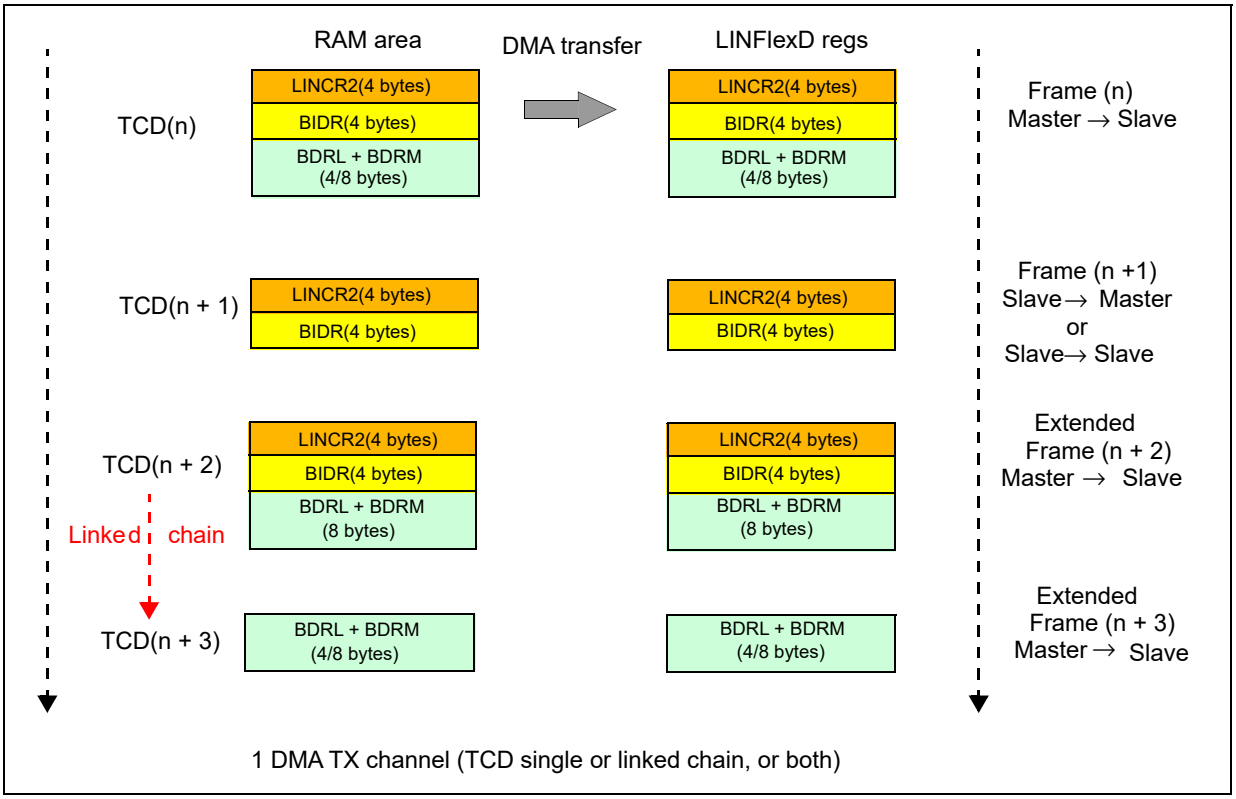
Table 729. TCD control fields description (continued)

TCD field	Level	Description
INT_MAJ	High	Enable an interrupt when major iteration count completes
START	High	The channel is explicitly started via a software initiated service request
DONE	High	Channel done (the DMA has completed the outer major loop)
D_REQ	High	Disable request If this flag is set the DMA hardware automatically clears the corresponding DMAERQ bit when the current major iteration count reaches zero.

44.3.5.3 Master node: TX mode

On a master node, in TX mode the DMA interface requires a single TX channel. Each TCD controls a single frame, except for the extended frames (multiple TCDs). The memory map associated with the TCD chain (RAM area and LINFlexD registers) is given in *Figure 704*.

Figure 704. Master node: TX memory map



The TCD chain of the DMA Tx channel on a master node supports:

- Master to Slave: transmission of the entire frame (header + data)
- Slave to Master: transmission of the header; the data reception is controlled by the Rx channel on the master node
- Slave to Slave: transmission of the header

Table 730 provides the register settings of the LINCR2 and BIDR for each class of LIN frame.

Table 730. Master node: Tx mode - Register setting

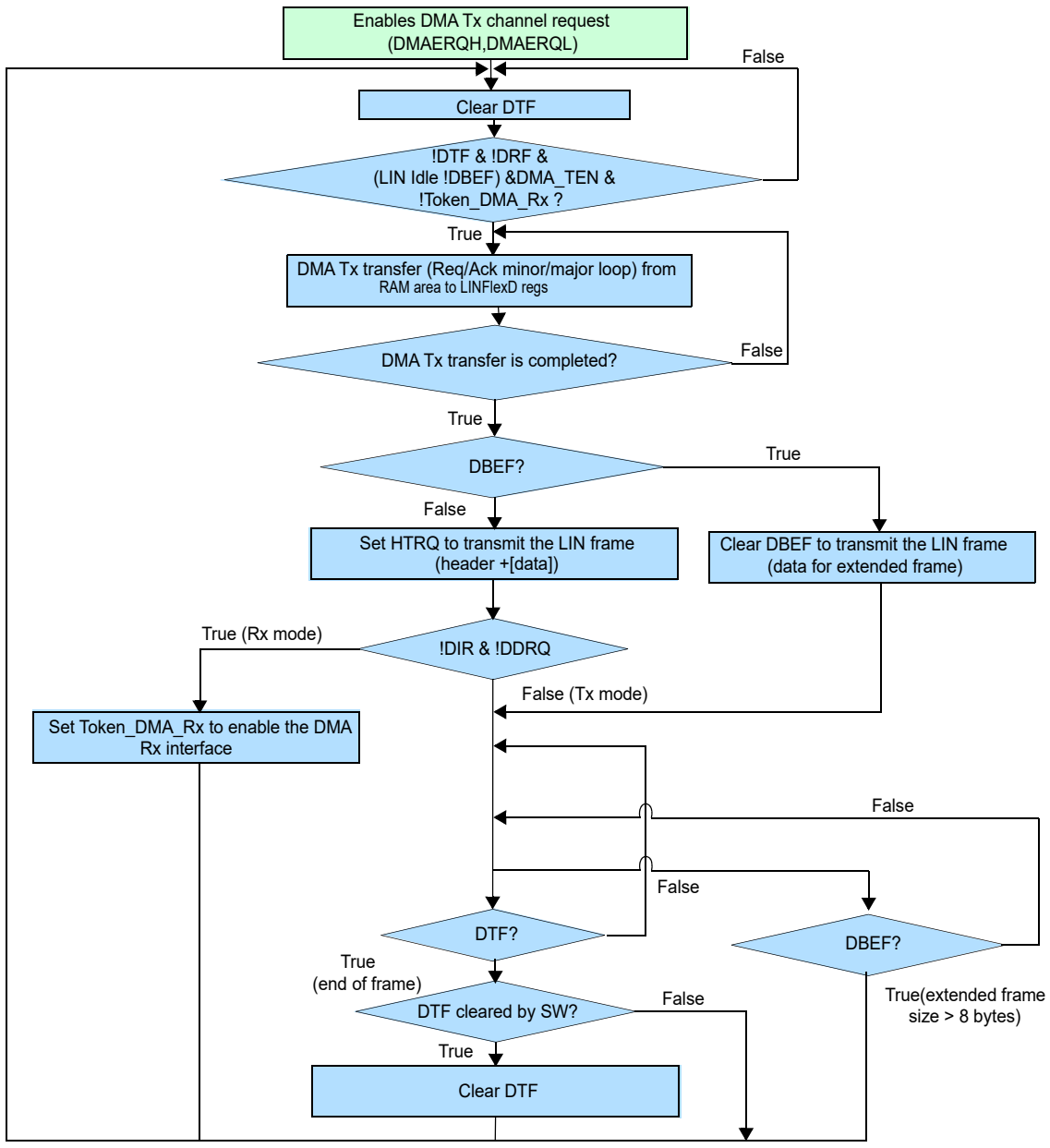
LIN frame	LINCR2	BIDR
Master to Slave	DDRQ = 1 DTRQ = 0 HTRQ = 0	DFL = payload size ID = address CCS = checksum DIR = 1 (TX)
Slave to Master	DDRQ = 0 DTRQ = 0 HTRQ = 0	DFL = payload size ID = address CCS = checksum DIR = 0 (RX)
Slave to Slave	DDRQ = 1 DTRQ = 0 HTRQ = 0	DFL = payload size ID = address CCS = checksum DIR = 0 (RX)

The concept FSM to control the DMA Tx interface is given in *Figure 705*. DMA TX FSM moves to Idle state immediately at next clock edge if DMATXE[0] = 0. The TCD setting (word transfer) is given in *Table 731*. All other TCD fields = 0. TCD settings based on halfword or byte transfer are allowed.

Table 731. TCD setting: Master node -Tx mode

TCD field	Value	Description
CITER[14:0]	1	Single iteration for the major loop
BITER[14:0]	1	Single iteration for the major loop
NBYTES[31:0]	[4 + 4] + 0/4/8 = N	Data buffer is filled with dummy bytes if length is not word-aligned LINCR2 + BIDR + BDRL + BDRM
SADDR[31:0]	—	RAM address
SOFF[15:0]	4	Word increment
SSIZE[2:0]	2	Word transfer
SLAST[31:0]	–N	—
DADDR[31:0]	—	LINCR2 address
DOFF[15:0]	4	Word increment
DSIZE[2:0]	2	Word transfer
DLAST_SGA[31:0]	–N	No scatter/gather processing
INT_MAJ	0/1	Interrupt disabled/enabled
D_REQ	1	Only on the last TCD of the chain
START	0	No software request

Figure 705. Master node: DMA Tx FSM (concept scheme)

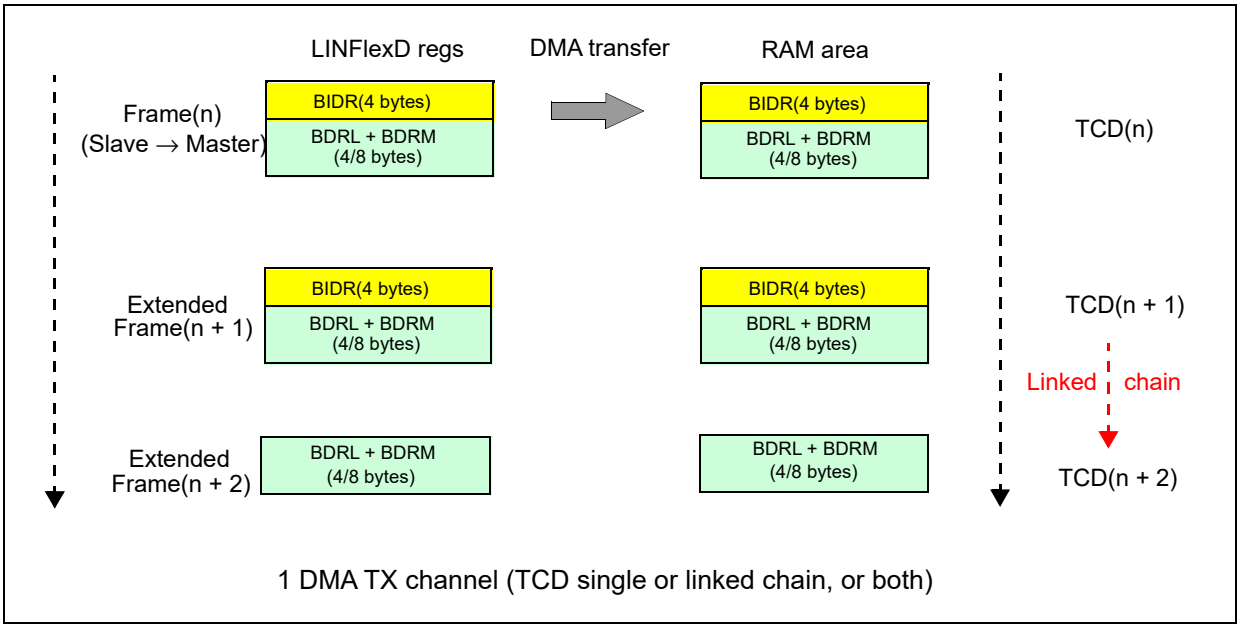


Note: DTF Flag gets auto cleared by Hardware if Software does not clear it within 3 ipg_clk cycles.

44.3.5.4 Master node: RX mode

On a master node in RX mode, the DMA interface requires a single RX channel. Each TCD controls a single frame, except for the extended frames (multiple TCDs). The memory map associated with the TCD chain (RAM area and LINFlexD registers) is given in *Figure 706*.

Figure 706. Master node: RX memory map



The TCD chain of the DMA Rx channel on a master node supports Slave to Master: reception of the data field of the header.

The BIDR register is optionally copied into the RAM area. This BIDR field (part of FIFO data) contains the ID of each message, which only allows the CPU to figure out which ID the LINFlexD DMA received if the single DMA channel setup is used.

The concept FSM to control the DMA Rx interface is given in *Figure 707*. DMA RX FSM moves to Idle state immediately at the next clock edge if DMARXE[0] = 0. The TCD setting (word transfer) is given in *Table 732*. All other TCD fields = 0. TCD settings based on halfword or byte transfer are allowed.

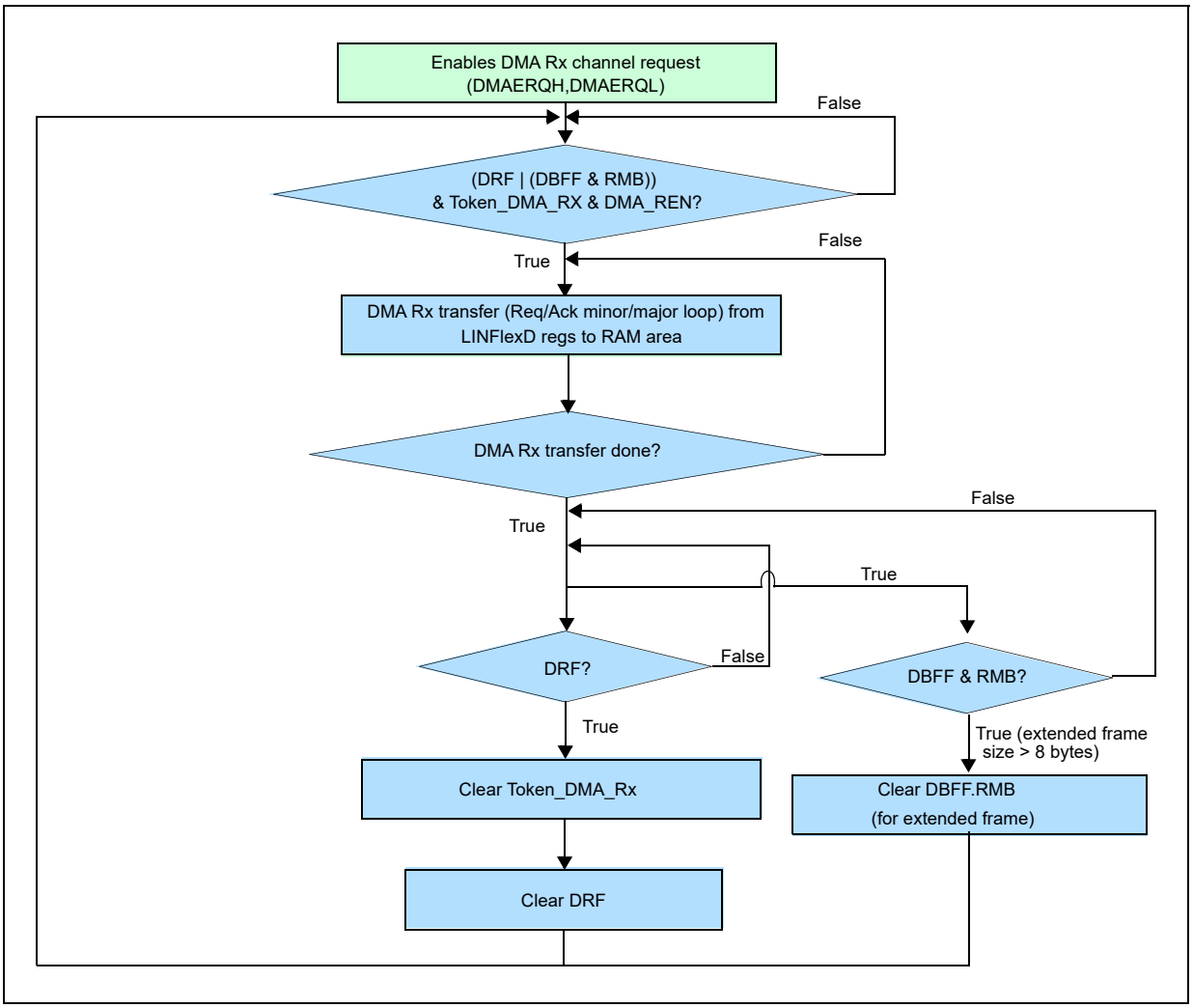
Table 732. TCD setting: Master node - Rx mode

TCD field	Value	Description
CITER[14:0]	1	Single iteration for the major loop
BITER[14:0]	1	Single iteration for the major loop
NBYTES[31:0]	[4] +4/8 = N	Data buffer is filled with dummy bytes if length is not word-aligned BIDR + BDRL + BDRM
SADDR[31:0]	—	BIDR address
SOFF[15:0]	4	Word increment
SSIZE[2:0]	2	Word transfer
SLAST[31:0]	–N	—
DADDR[31:0]	—	RAM address
DOFF[15:0]	4	Word increment
DSIZE[2:0]	2	Word transfer
DLAST_SGA[31:0]	–N	No scatter/gather processing

Table 732. TCD setting: Master node - Rx mode (continued)

TCD field	Value	Description
INT_MAJ	0/1	Interrupt disabled/enabled
D_REQ	1	Only on the last TCD of the chain
START	0	No software request

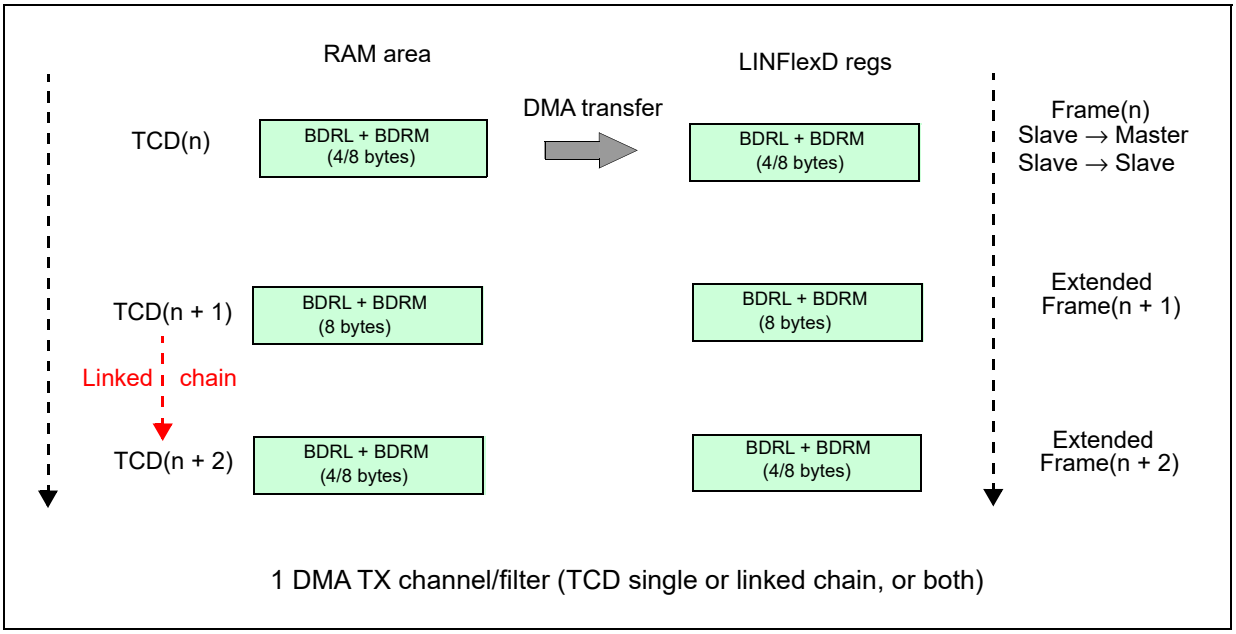
Figure 707. Master node — DMA Rx FSM (concept scheme)



44.3.5.5 Slave node: TX mode

On a slave node in TX mode, the DMA interface requires a DMA TX channel for each ID filter programmed in TX mode. In case a single DMA TX channel is available, a single ID field filter must be programmed in TX mode. Each TCD controls a single frame, except for the extended frames (multiple TCDs). The memory map associated with the TCD chain (RAM area and LINFlexD registers) is given in *Figure 708*.

Figure 708. Slave node — TX memory map



The TCD chain of the DMA Tx channel on a slave node supports:

- Slave to Master: transmission of the data field
- Slave to Slave: transmission of the data field

The register setting of the LINCR2, IFER, IFMR, and IFCR registers is given in *Table 733*.

Table 733. Slave node: Tx mode - Register setting

LIN frame	LINCR2	IFER	IFMR	IFCR
Slave to Master or Slave to Slave	DDRQ = 0 DTRQ = 0 HTRQ = 0	To enable an ID filter (Tx mode) for each DMA TX channel	– Identifier list mode Identifier mask mode	DFL = payload size ID = address CCS = checksum DIR = 1 (TX)

The concept FSM to control the DMA Tx interface is given in *Figure 709*. DMA TX FSM moves to Idle state if DMATXE[x] = 0 where x = IFMI – 1. The TCD setting (word transfer) is given in *Table 734*. All other TCD fields = 0. TCD settings based on halfword or byte transfer are allowed.

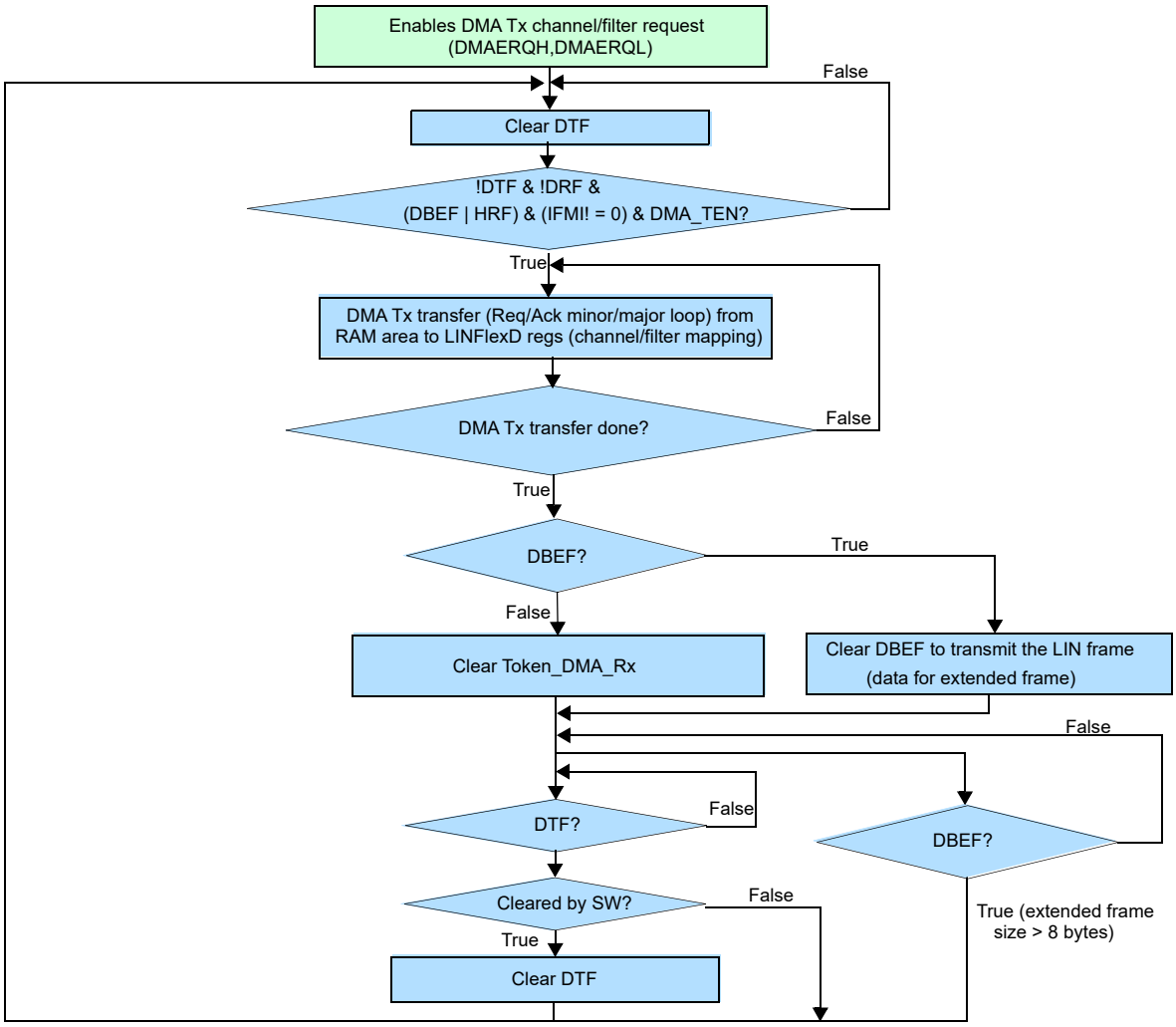
Table 734. TCD setting: Slave node - Tx mode

TCD field	Value	Description
CITER[14:0]	1	Single iteration for the major loop
BITER[14:0]	1	Single iteration for the major loop
NBYTES[31:0]	4/8 = N	Data buffer is filled with dummy bytes if length is not word-aligned BDRL + BDRM
SADDR[31:0]	—	RAM address
SOFF[15:0]	4	Word increment

Table 734. TCD setting: Slave node - Tx mode (continued)

TCD field	Value	Description
SSIZE[2:0]	2	Word transfer
SLAST[31:0]	-N	—
DADDR[31:0]	—	BDRL address
DOFF[15:0]	4	Word increment
DSIZE[2:0]	2	Word transfer
DLAST_SGA[31:0]	-N	No scatter/gather processing
INT_MAJ	0/1	Interrupt disabled/enabled
D_REQ	1	Only on the last TCD of the chain
START	0	No software request

Figure 709. Slave node: DMA Tx FSM (concept scheme)

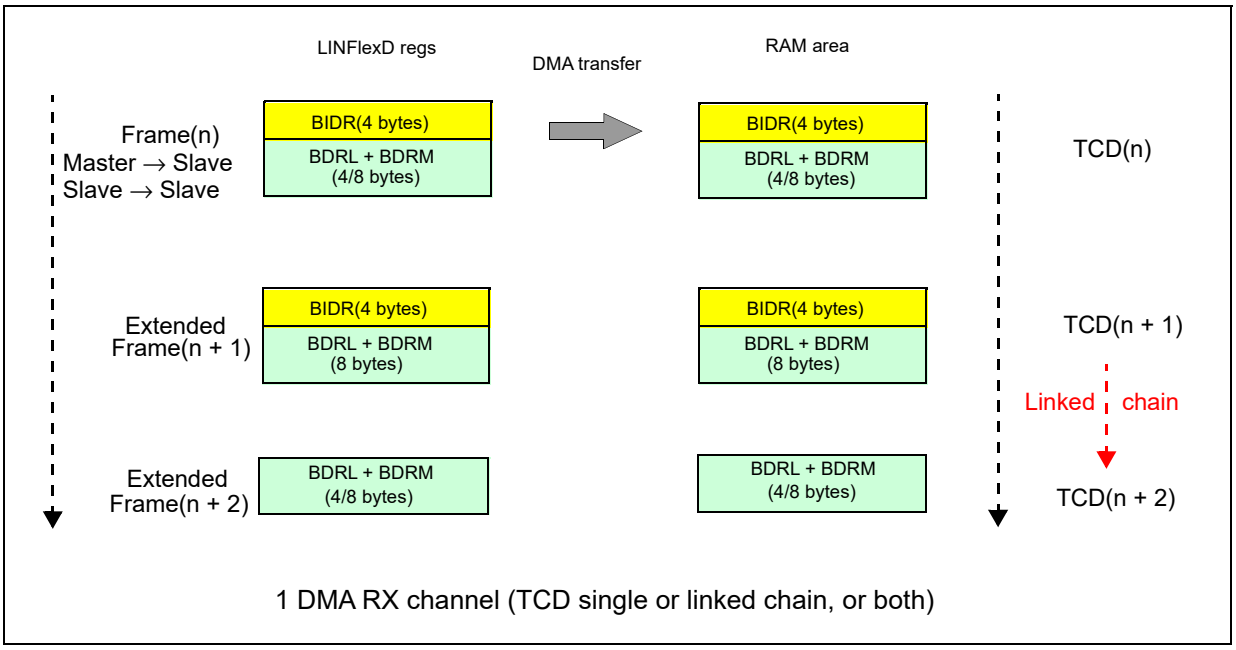


Note: DTF Flag gets auto cleared by Hardware if Software does not clear it within 3 ipg_clk cycles.

44.3.5.6 Slave node: RX mode

On a slave node in RX mode, the DMA interface requires a DMA RX channel for each ID filter programmed in RX mode. In case a single DMA RX channel is available, a single ID field filter must be programmed in RX mode. Each TCD controls a single frame, except for the extended frames (multiple TCDs). The memory map associated with the TCD chain (RAM area and LINFlexD registers) is given in *Figure 710*.

Figure 710. Slave node: RX memory map



The TCD chain of the DMA Rx channel on a slave node supports:

- Master to Slave: reception of the data field
- Slave to Slave: reception of the data field

The register setting of the LINCR2, IFER, IFMR, and IFCR registers is given in *Table 735*.

Table 735. Slave node: Rx mode - Register setting

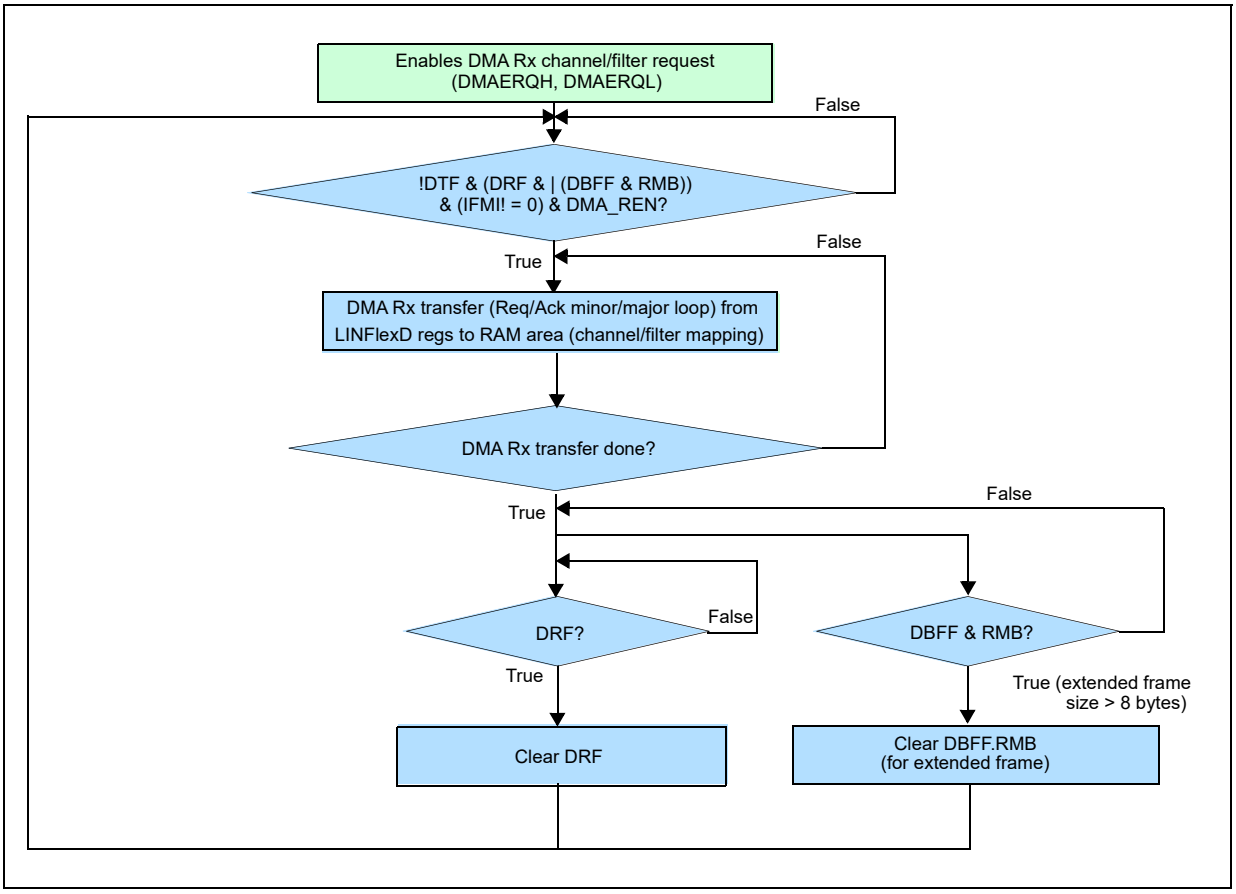
LIN frame	LINCR2	IFER	IFMR	IFCR
Master to Slave or Slave to Slave	DDRQ = 0 DTRQ = 0 HTRQ = 0	To enable an ID filter (Rx mode) for each DMA RX channel	– Identifier list mode Identifier mask mode	DFL = payload size ID = address CCS = checksum DIR = 0 (RX)

The concept FSM to control the DMA Rx interface is given in *Figure 711*. DMA Rx FSM moves to Idle state if DMARXE[x] = 0 where x = IFMI – 1. The TCD setting (word transfer) is given in *Table 736*. All other TCD fields = 0. TCD settings based on halfword or byte transfer are allowed.

Table 736. TCD setting: Slave node - Rx mode

TCD field	Value	Description
CITER[14:0]	1	Single iteration for the major loop
BITER[14:0]	1	Single iteration for the major loop
NBYTES[31:0]	[4] + 4/8 = N	Data buffer is filled with dummy bytes if length is not word-aligned BIDR + BDRL + BDRM
SADDR[31:0]	—	BIDR address
SOFF[15:0]	4	Word increment
SSIZE[2:0]	2	Word transfer
SLAST[31:0]	-N	—
DADDR[31:0]	—	RAM address
DOFF[15:0]	4	Word increment
DSIZE[2:0]	2	Word transfer
DLAST_SGA[31:0]	-N	No scatter/gather processing
INT_MAJ	0/1	Interrupt disabled/enabled
D_REQ	1	Only on the last TCD of the chain
START	0	No software request

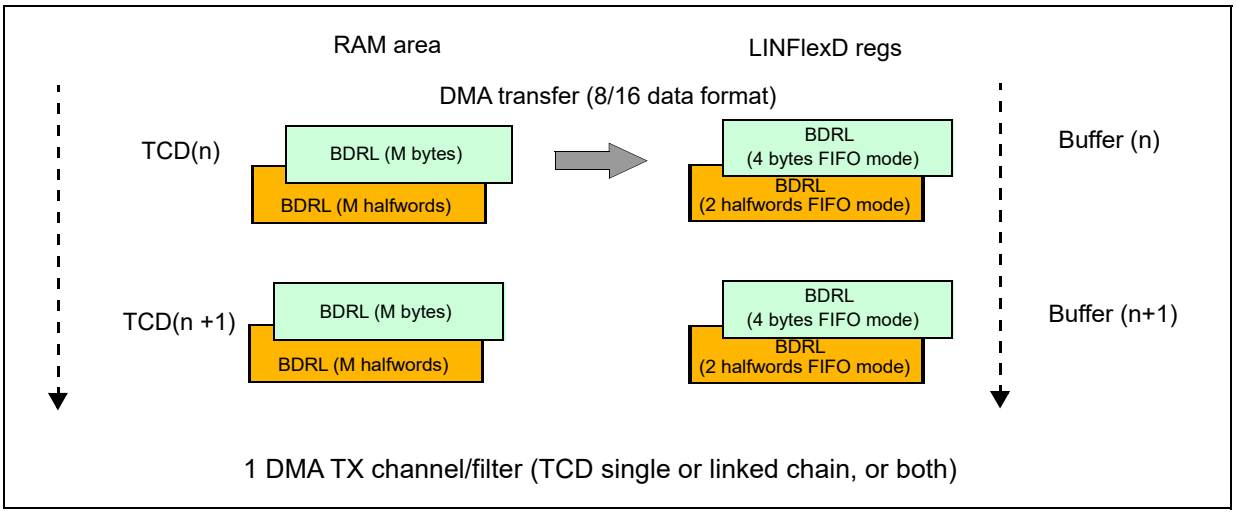
Figure 711. Slave node — DMA Rx FSM (concept scheme)



44.3.5.7 UART: TX mode

In UART TX mode, the DMA interface requires a DMA TX channel. A single TCD can control the transmission of an entire Tx buffer. The memory map associated with the TCD chain (RAM area and LINFlexD registers) is given in *Figure 712*.

Figure 712. UART: TX memory map



The UART TX buffer must be configured in FIFO mode in order to:

- Allow the transfer of large data buffer by a single TCD
- Adsorb the latency, following a DMA request (due to the DMA arbitration), to move data from the RAM to the FIFO
- Use low priority DMA channels

The Tx FIFO size is:

- 4 bytes in 8-bit data format
- 2 halfwords in 16-bit data format

A DMA request is triggered by FIFO-not-full (TX) status signals.

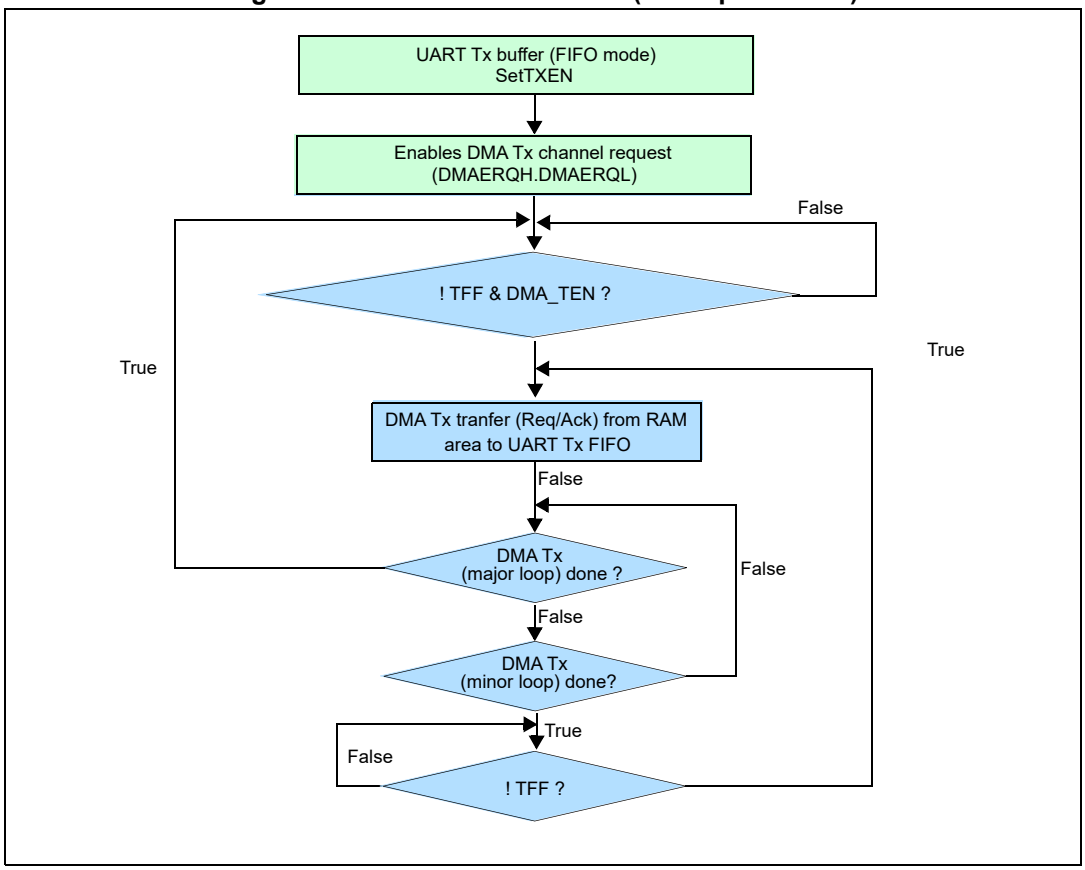
The concept FSM to control the DMA Tx interface is given in *Figure 713*. DMA Tx FSM will move to Idle state if DMATXE[0] = 0. The TCD setting (typical case) is given in *Table 737*. All other TCD fields = 0. The minor loop transfers a single byte/halfword as soon as a free entry is available in the Tx FIFO.

Table 737. TCD setting: UART - Tx mode

TCD field	Value		Description
	8 bits data	16 bits data	
CITER[14:0]	M		Multiple iterations for the major loop
BITER[14:0]	M		Multiple iterations for the major loop
NBYTES[31:0]	1	2	Minor loop transfer = 1 or 2 bytes
SADDR[31:0]	—		RAM address
SOFF[15:0]	1	2	Byte/halfword increment
SSIZE[2:0]	0	1	Byte/halfword transfer
SLAST[31:0]	-M	-M × 2	—
DADDR[31:0](1)	—		BDRL address
DOFF[15:0]	0		No increment (FIFO)
DSIZE[2:0]	0	1	Byte/halfword transfer
DLAST_SGA[31:0]	0		No scatter/gather processing
INT_MAJ	0/1		Interrupt disabled/enabled
D_REQ	1		Only on the last TCD of the chain
START	0		No software request

1. DADDR[31:0] = BDRL + 0x3 for byte transfer — DADDR [31:0] = BDRL + 0x2 for halfword transfer.

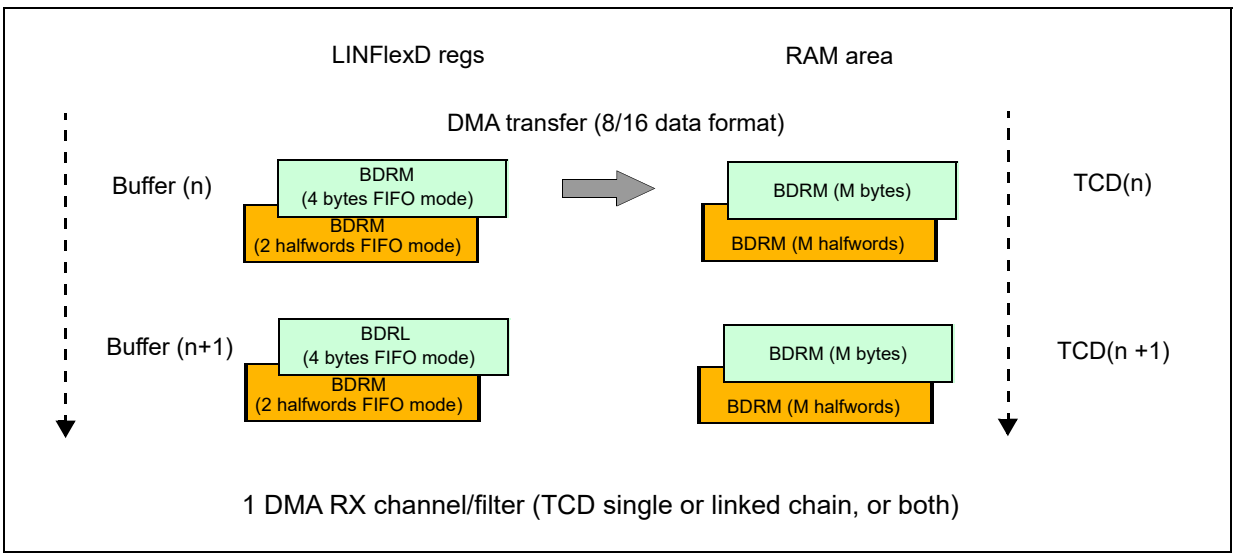
Figure 713. UART: DMA Tx FSM (concept scheme)



44.3.5.8 UART: RX mode

In UART RX mode, the DMA interface requires a DMA RX channel. A single TCD can control the reception of an entire Rx buffer. The memory map associated with the TCD chain (RAM area and LINFlexD registers) is given in *Table 738*.

Figure 714. UART: RX memory map



The UART RX buffer must be configured in FIFO mode in order to:

- Allow the transfer of large data buffer by a single TCD
- Adsorb the latency, following a DMA request (due to the DMA arbitration), to move data from the FIFO to the RAM
- Use low priority DMA channels

The Rx FIFO size is:

- 4 bytes in 8-bit data format
- 2 halfwords in 16-bit data format

This will be sufficient because just one byte allows a reaction time of about 3.8 µs (at 2 Mbit/s) (~450 clock cycles at 120 MHz) before the transmission is affected. A DMA request is triggered by FIFO and not by empty (Rx) status signals.

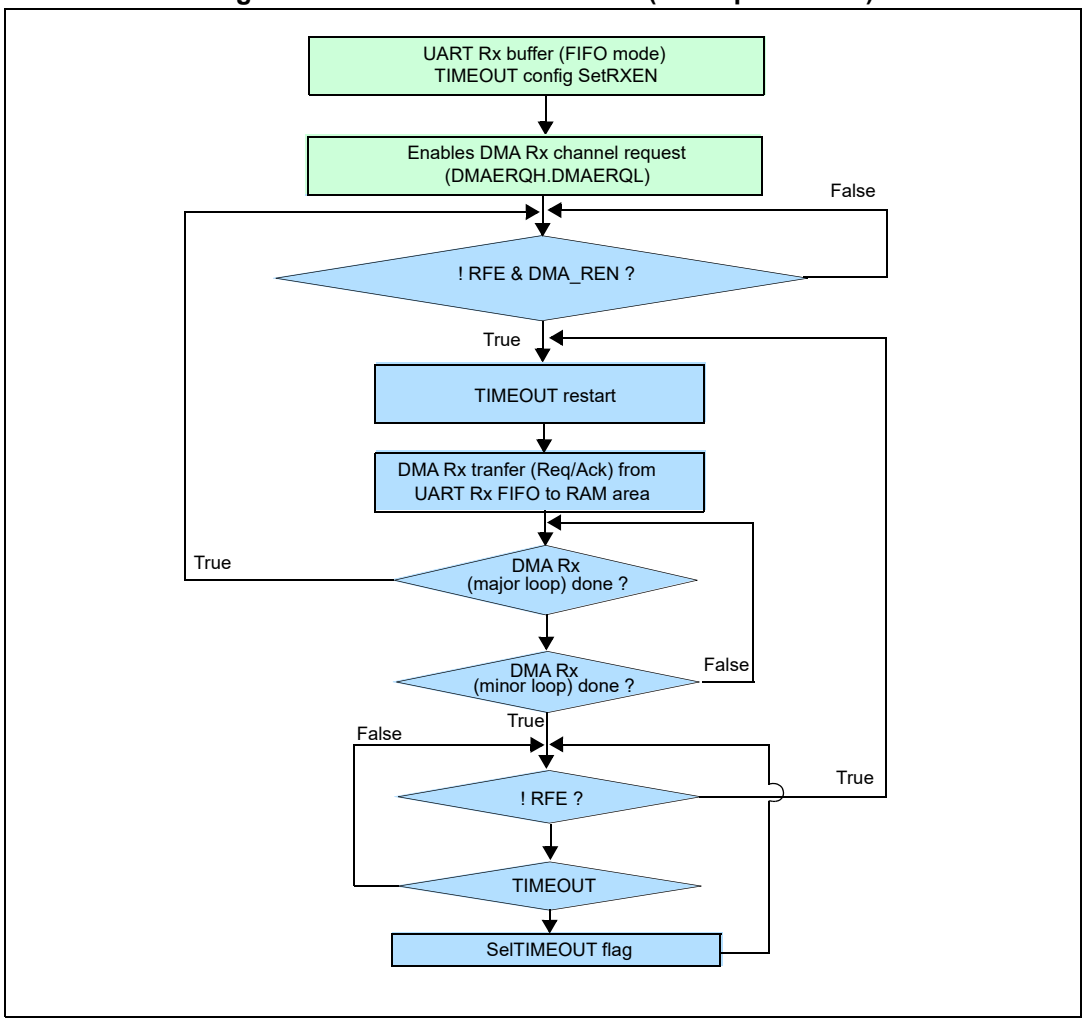
The concept FSM to control the DMA Rx interface is given in *Figure 715*. DMA Rx FSM will move to Idle state if DMARXE[0] = 0. The TCD setting (typical case) is given in the next table. All other TCD fields equal zero. The minor loop transfers a single byte/halfword as soon an entry is available in the Rx FIFO. A new software reset bit is required that allows the LINFlexD FSMs to be reset in case this timeout state is reached or in any other case. The timeout counter can be re-written by software at any time to extend the timeout period.

Table 738. TCD setting: UART - Rx mode

TCD field	Value		Description
	8 bits data	16 bits data	
CITER[14:0]	M		Multiple iterations for the major loop
BITER[14:0]	M		Multiple iterations for the major loop
NBYTES[31:0]	1	2	Minor loop transfer = 1 or 2 bytes
SADDR[31:0](1)	—		BDRM address
SOFF[15:0]	0		No increment (FIFO)
SSIZE[2:0]	0	1	Byte/halfword transfer
SLAST[31:0]	0		—
DADDR[31:0]	—		RAM address
DOFF[15:0]	1	2	Byte/halfword increment
DSIZE[2:0]	0	1	Byte/halfword transfer
DLAST_SGA[31:0]	-M	-M × 2	No scatter/gather processing
INT_MAJ	0/1		Interrupt disabled/enabled
D_REQ	1		Only on the last TCD of the chain
START	0		No software request

1. SADDR = BDRM + 0x3 for byte transfer; SADDR[31:0] = BDRM + 0x2 for halfword transfer.

Figure 715. UART — DMA Rx FSM (concept scheme)



44.3.5.9 DMA interface timing

The LINFlexD DMA interface includes the following signals:

- ipd_req
  - DMA peripheral request, asserted until the DMA transfer has been completed
- dma_done_lw
  - DMA minor loop done. It indicates that the channel's data transfer of nbytes (the minor loop) has been completed. It is asserted for one cycle after the data phase of the channel's last AMBA AHB write access, or when an error condition is detected
- dma_ipd_complete
  - DMA major loop finished. It is asserted for one cycle after the data phase of the last write
- dma_ipd_ack, dma_ipd_done
  - Unused in this implementation

All signals of the DMA interface must be registered in order to meet timing requirements of the DMA engine.

Figure 716. DMA interface (major loop — single iteration)

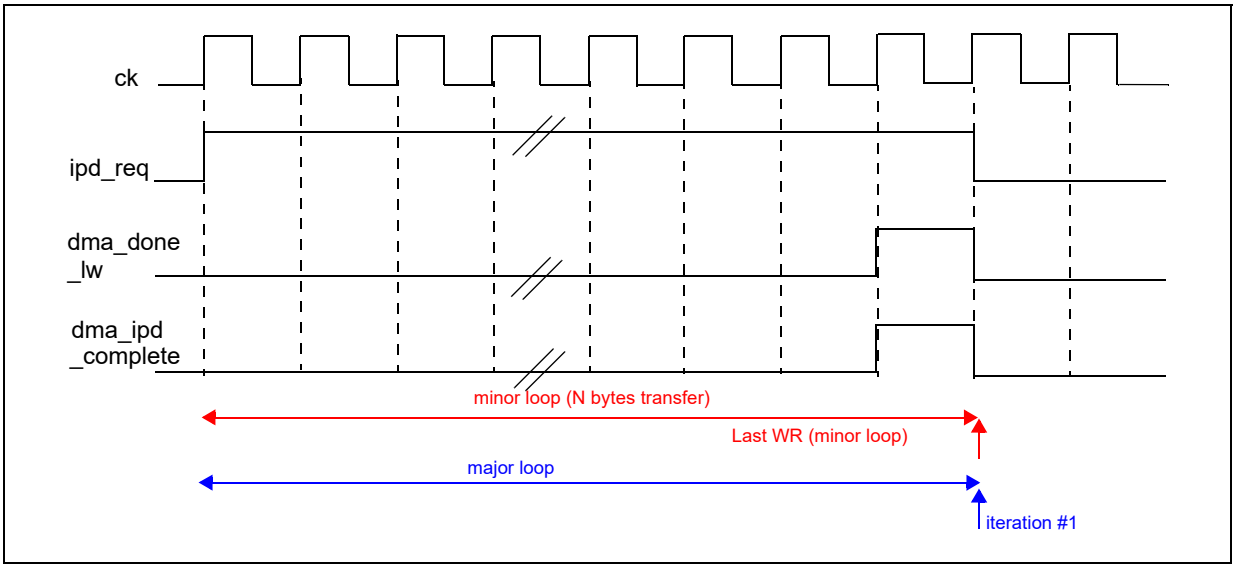
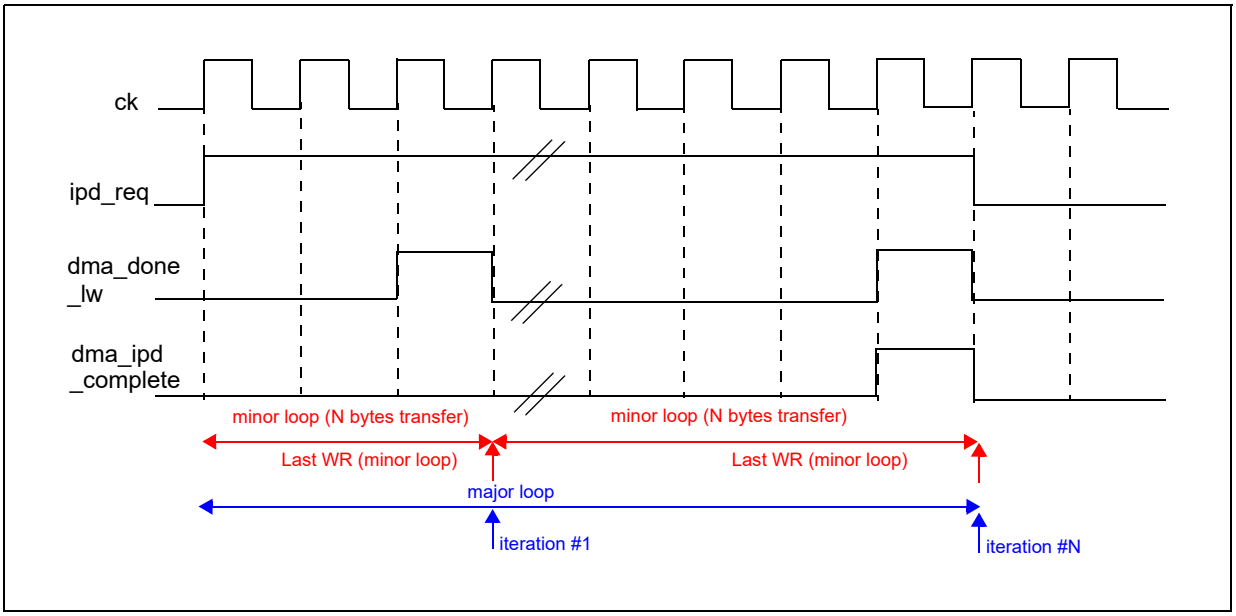


Figure 717. DMA interface (major loop — N iteration)



44.3.5.10 Use cases and limitations

- In LIN slave mode, the DMA capability can be used only if the ID filtering mode is activated. The number of ID filters enabled must be equal to the number of DMA

channels enabled. The correspondence between channel number and ID filter is based on IFMI (identifier filter match index).

- In LIN master mode both the DMA channels (TX and RX) must be enabled in case the DMA capability is required.
- In UART mode the DMA capability can be used only if the UART Tx/Rx buffers are configured as FIFOs.
- DMA and CPU operating modes are mutually exclusive for the data/frame transfer on a UART or LIN node. Once a DMA transfer is finished the CPU can manage subsequent accesses.
- Error management must always be executed via the CPU enabling the related error interrupt sources. DMA capability does not provide support for error management. Error management means checking status bits, handling IRQs, and potentially canceling DMA transfers.
- The DMA programming model must be coherent with the TCD setting defined in this document
- When IPG_STOP is requested, SW has to first disable DMATXE/DMARXE channel registers, after the current minor loop finishes (indicated by the clearing of ACTIVE bit in DMA TCD register). This ensures that IPG_STOP_ACK is generated and IP enters STOP mode.

44.4 Memory map and register description

44.4.1 Memory map

Table 739 shows the LINFlexD memory/register map.

Note: In Master mode, the registers IFCR0 – IFCR15 are not present.

In Master Mode, read access to IFMI register and read/write access to IFER and IFMR registers would result in bus transfer error.

Table 739. LINFlexD memory map

Address offset (hex)	Register description	Register classification(1)	Location
0x00	LIN Control register 1 (LINCR1)	Saf-Relv	*Section 44.4.2.1*
0x04	LIN Interrupt Enable register (LINIER)	Saf-Relv	*Section 44.4.2.2*
0x08	LIN Status register (LINSR)	No-Saf	*Section 44.4.2.3*
0x0C	LIN Error Status register (LINESR)	No-Saf	*Section 44.4.2.4*
0x10	UART Mode Control register (UARTCR)	Saf-Relv	*Section 44.4.2.5*
0x14	UART Mode Status register (UARTSR)	No-Saf	*Section 44.4.2.6*
0x18	LIN Timeout Control Status register (LINTCSR)	Saf-Relv	*Section 44.4.2.7*
0x1C	LIN Output Compare register (LINOCR)	Saf-Relv	*Section 44.4.2.8*
0x20	LIN Timeout Control register (LINTOCR)	Saf-Relv	*Section 44.4.2.9*
0x24	LIN Fractional Baud Rate register (LINFBRR)	Saf-Relv	*Section 44.4.2.10*

Table 739. LINFlexD memory map (continued)

Address offset (hex)	Register description	Register classification[1]	Location
0x28	LIN Integer Baud Rate register (LINIBRR)	Saf-Relv	Section 44.4.2.11
0x2C	LIN Checksum Field register (LINCFR)	Saf-Relv	Section 44.4.2.12
0x30	LIN Control register 2 (LINCR2)	Saf-Relv	Section 44.4.2.13
0x34	Buffer Identifier register (BIDR)	Saf-Relv	Section 44.4.2.14
0x38	Buffer Data Register Least Significant (BDRL)	No-Saf	Section 44.4.2.15
0x3C	Buffer Data Register Most Significant (BDRM)	No-Saf	Section 44.4.2.16
0x40	Identifier Filter Enable register (IFER)	Saf-Relv	Section 44.4.2.17
0x44	Identifier Filter Match index (IFMI)	Saf-Relv	Section 44.4.2.18
0x48	Identifier Filter Mode register (IFMR)	Saf-Relv	Section 44.4.2.19
0x4C[2]	Identifier Filter Control registers (IFCR0)	Saf-Relv	Section 44.4.2.20
0x50[2]	Identifier Filter Control registers (IFCR1)	Saf-Relv	Section 44.4.2.21
0x54[2]	Identifier Filter Control registers (IFCR2)	Saf-Relv	Section 44.4.2.20
0x58[2]	Identifier Filter Control registers (IFCR3)	Saf-Relv	Section 44.4.2.21
0x5C[2]	Identifier Filter Control registers (IFCR4)	Saf-Relv	Section 44.4.2.20
0x60[2]	Identifier Filter Control registers (IFCR5)	Saf-Relv	Section 44.4.2.21
0x64[2]	Identifier Filter Control registers (IFCR6)	Saf-Relv	Section 44.4.2.20
0x68[2]	Identifier Filter Control registers (IFCR7)	Saf-Relv	Section 44.4.2.21
0x6C[2]	Identifier Filter Control registers (IFCR8)	Saf-Relv	Section 44.4.2.20
0x70[2]	Identifier Filter Control registers (IFCR9)	Saf-Relv	Section 44.4.2.21
0x74[2]	Identifier Filter Control registers (IFCR10)	Saf-Relv	Section 44.4.2.20
0x78[2]	Identifier Filter Control registers (IFCR11)	Saf-Relv	Section 44.4.2.21
0x7C[2]	Identifier Filter Control registers (IFCR12)	Saf-Relv	Section 44.4.2.20
0x80[2]	Identifier Filter Control registers (IFCR13)	Saf-Relv	Section 44.4.2.21
0x84[2]	Identifier Filter Control registers (IFCR14)	Saf-Relv	Section 44.4.2.20
0x88[2]	Identifier Filter Control registers (IFCR15)	Saf-Relv	Section 44.4.2.21
0x8C	Global Control register (GCR)	Saf-Relv	Section 44.4.2.22
0x90	UART Preset Timeout register (UARTPTO)	Saf-Relv	Section 44.4.2.23
0x94	UART Current Timeout register (UARTCTO)	No-Saf	Section 44.4.2.24
0x98	DMA Tx Enable register (DMATXE)	Saf-Relv	Section 44.4.2.25
0x9C	DMA Rx Enable register (DMARXE)	Saf-Relv	Section 44.4.2.26

1. No-Saf = Not safety relevant, Saf-Lat = Safety Latent, Saf-Relv = Safety Relevant, Saf-Crit = Safety Critical.
2. Refer to the *Chapter 7: Device configuration* for the number of filters used in each instance of LINFlexD in this device.

44.4.2 Register description

The following points should be considered for the below mentioned LINFlexD registers:

- Reset values are with-slave/without-slave (master only) format.
- If not specified, then the bit can be configured for both master only and master/slave format; in other words, for both values of generic slave = 0 and slave = 1.

44.4.2.1 LIN Control Register 1 (LINCR1)

LINCR1 consists of control bits used to configure features of the LINFlexD.

Address: 0x0000 Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	CCD(1)	CFD(1)	LASE(1)	AUTOWU(1)	MBL(1)				BF(1)	0	LBKM(1)	MME(1)	SSBL(1)	RBLM(1)	SLEEP	INIT
W																
Reset	0	0	0	0	0	0	0	0	1	0	0	0/1(2)	0	0	1	0

1. This field can be read in any modes and written only in *Initialization* mode.
2. When slave = 0, this field reads '1' and cannot be programmed.

Figure 718. LIN Control Register 1 (LINCR1)

Table 740. LINCR1 field descriptions

Field	Description
16 CCD	Calculation disable 0 Checksum calculation is done by hardware. When this bit is reset the LINCFR register is read-only. 1 Checksum calculation is disabled. When this bit is set the LINCFR register is read/write. User can program this register to send a software calculated checksum/CRC (provided CFD is reset). This bit can be written during Initialization mode only. It is read-only in Normal mode.
17 CFD	Checksum field disable 0 Checksum field is sent after the required number of data bytes are sent 1 No checksum field is sent in the frame This bit can be configured during Initialization mode only. It is read-only in Normal mode.
18 LASE	LIN Auto Synchronization Enable 0 Auto synchronization disabled 1 Auto synchronization enabled This bit can be programmed in Initialization mode only. It is read-only in Normal mode. (Note: If generic auto_sync = 0, then this bit will always read a 0).

Table 740. LINCR1 field descriptions (continued)

Field	Description
19 AUTOWU	Auto Wakeup 0 Sleep bit is cleared by software only 1 Sleep bit gets cleared by hardware whenever WUF bit of LINSR is set This bit can be configured during Initialization mode only. This bit is utilized in UART mode also.
20:23 MBL[3:0]	Master Break Length These bits choose the length of the Sync break to be generated by the master. 0x0 10-bit break length 0x1 11-bit break length 0x2 12-bit break length 0x3 13-bit break length 0x4 14-bit break length 0x5 15-bit break length 0x6 16-bit break length 0x7 17-bit break length 0x8 18-bit break length 0x9 19-bit break length 0xA 20-bit break length 0xB 21-bit break length 0xC 22-bit break length 0xD 23-bit break length 0xE 36-bit (cooling) break length 0xF 50-bit break length These bits can be programmed in Initialization mode only. They are read-only in Normal mode.
24 BF	By-pass filter 0 No IRQ if ID does not match any filter 1 A RX IRQ is generated on ID not matching any filter This bit can be programmed during Initialization mode only. Note: If generic slave = 0 or no_of_filters = 0, then this bit will always read a 1, and cannot be programmed.
25	Reserved
26 LBKM	Loop Back mode 0 Loop Back Mode disabled 1 Loop Back mode enabled Refer to *Section 44.3.2.2*. Note that this bit can be programmed only in Initialization mode. This bit is utilized in UART mode also.
27 MME	Master mode enable 0 Slave Mode 1 Master Mode This bit can be programmed in Initialization mode only. It is read-only in Normal mode. If generic *slave* = 0, then this bit will read a ‘1’ always, and cannot be programmed.

Table 740. LINCR1 field descriptions (continued)

Field	Description
28 SSBL	Slave Mode Sync Break Length 0 11-bit break length 1 10-bit break length This bit can be programmed in Initialization mode only. It is read-only in Normal mode.
29 RBLM	Receiver Buffer Locked mode 0 Receiver Buffer not locked, next incoming message will overwrite the old one 1 Receiver buffer locked against overrun. Once the buffer is full the next incoming message will be discarded if buffer is not released — in other words, RMB is not reset by software. This bit can be programmed in Initialization mode only. It is read-only in Normal mode. This bit is utilized in UART mode also.
30 SLEEP	Sleep Mode Request This bit is set by software to request LINFlexD to enter Sleep mode. This bit is cleared by software or hardware (if AUTOWU bit in LINCR1 and WUF bit in LINSR are set) to exit sleep mode. This bit is utilized in UART mode also.
31 INIT	Initialization Mode Request The software sets this bit to switch the hardware into Initialization mode. On clearing this bit (and if SLEEP bit is also zero) LINFlexD enters normal mode. This bit is utilized in UART mode also.

44.4.2.2 LIN Interrupt enable register (LINIER)

Address: 0x0004 Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	SZIE	OCIE	BEIE	CEIE	HEIE	0	0	FEIE	BOIE	LSIE	WUIE	DBFIE	DBEIE_TOIE	DRIE	DTIE	HRIE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 719. LIN Interrupt enable register (LINIER)

Table 741. LINIER field descriptions

Field	Description
16 SZIE	Stuck at Zero Interrupt Enable 0 No interrupt 1 Interrupt enabled An interrupt is generated if this bit is set and the Stuck at Zero Flag (SZF) in LINESR or UARTSR is set.
17 OCIE	Output Compare Interrupt Enable 0 No interrupt 1 Interrupt generated when OCF bit in LINESR or UARTSR is set
18 BEIE	Bit Error Interrupt Enable 0 No interrupt 1 Interrupt generated when BEF bit in LINESR is set
19 CEIE	Checksum Error Interrupt Enable 0 No interrupt 1 Interrupt enabled An interrupt is generated if this bit is set and the Checksum Error Flag (CEF) is set in LINESR.
20 HEIE	Header Error Interrupt Enable 0 No interrupt 1 Interrupt enabled An interrupt is generated when this bit is set and either of the following flags are set SFEF, SDEF, IDPEF in LINESR are set. **Note:** If generic slave = 0, then this bit will always read a 0 and cannot be programmed.
23 FEIE	Frame Error Interrupt Enable 0 No interrupt 1 Interrupt generated if Frame Error Flag (FEF) bit is set in LINESR or UARTSR
24 BOIE	Buffer Overrun Error Interrupt Enable 0 No interrupt 1 Interrupt enabled An interrupt is generated if this bit is set and the Buffer Overrun Flag (BOF) is set in LINESR or UARTSR.
25 LSIE	LIN state Interrupt enable 0 No interrupt 1 Interrupt generated on entering the following states: Sync Del, Sync Field, Identifier field, Checksum Interrupt is generated only when entering the above fields. This interrupt can mainly be used for debugging purposes. The interrupt has no status flag.
26 WUIE	Wakeup interrupt enable 0 No interrupt 1 Interrupt enabled If this bit is set and the WUF in LINSR or UARTSR is set then an interrupt is generated.

Table 741. LINIER field descriptions (continued)

Field	Description
27 DBFIE	Data Buffer Full Interrupt enable 0 No interrupt 1 Interrupt enabled An interrupt is generated if this bit is set and the DBFF bit in LINSR is also set.
28 DBEIE_TOIE	Data Buffer Empty Interrupt enable/Timeout Interrupt Enable 0 No interrupt 1 Interrupt enabled An interrupt is generated if this bit is set and LINSR[DBEF] status bit is set (in LIN mode) or if this bit is set and UARTSR[TO] status bit is set (in UART mode).
29 DRIE	Data Reception complete Interrupt enable 0 No interrupt 1 Interrupt enabled An interrupt is generated when this bit is set and Data Received flag (DRF) in LINSR or UARTSR is set.
30 DTIE	Data Transmitted Interrupt enable 0 No interrupt 1 Interrupt enabled An interrupt is generated when this bit is set and Data Transmitted flag (DTF) in LINSR or UARTSR is set.
31 HRIE	Header Received Interrupt 0 No interrupt 1 Interrupt enabled An interrupt is generated when this bit is set and the Header Received flag (HRF) in LINSR is set. **Note:** If generic slave = 0, then this bit will always read a 0, and cannot be programmed.

44.4.2.3 LIN Status Register (LINSR)

This register consists of status bits indicating the state of the LINFlexD hardware.

Address: 0x0008 Access: User read/write

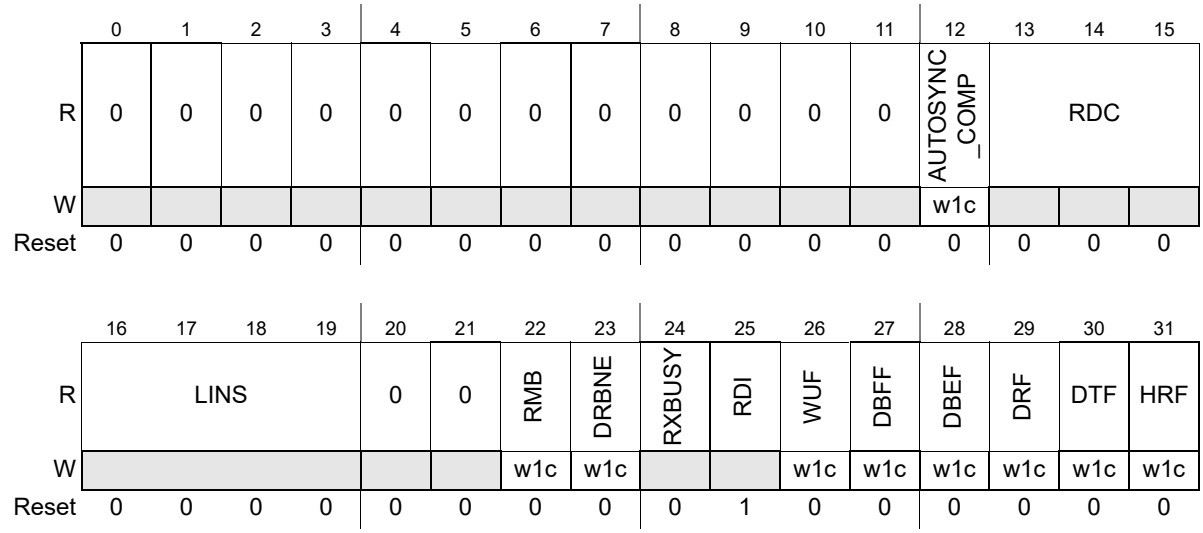


Figure 720. LIN Status Register (LINSR)

Table 742. LINSR field descriptions

Field	Description
12 AUTOSYNC_COMP	Autosynchronization Completed This bit is set after autosynchronization is complete and when the LASE bit in LINCR1 register is enabled. Only after this bit is set, the contents of LINIBRR and LINFBRR registers should be read in autosynchronization mode.
13:15 RDC	Receive Data Byte Count RDC contains the number of entries (bytes) in the Receive data buffer in LIN mode. RDCx is a read-only field available for debug purposes. 000 1 byte 001 2 bytes 010 3 bytes 011 4 bytes 100 5 bytes 101 6 bytes 110 7 bytes 111 8 bytes

Table 742. LINSR field descriptions (continued)

Field	Description
16:19 LINS	LIN state 0x0 Sleep mode LINFlexD is in Sleep mode to save power consumption. 0x1 Init 0x2 Idle This mode is entered when: SLEEP bit and INIT bit are reset by software, Wakeup pulse has been received on RX pin (AUTOWU set), Previous frame transmission/reception has been completed/aborted. 0x3 Sync Break In slave mode, a falling edge followed by a dominant state has been detected. Receiving sync break. In master mode, sync break transmission ongoing. In slave mode upon any error LIN state could be either Idle or Rec Break, depending on last bit detected on LIN_RX. If last bit detected is dominant then Rec_Break, otherwise Idle. 0x4 Sync Del In Slave mode, valid Sync break has been detected (10 bit or 11 bit). Waiting for a rising edge. In Master mode, Sync break transmission has been completed, sync delimiter transmission is ongoing. 0x5 Sync Field In Slave mode, a valid sync Del has been detected (recessive state for at least one bit time). Receiving sync field. In Master mode, sync field transmission ongoing. 0x6 Identifier Field In Slave mode, a valid sync field has been received. Receiving ID field. In Master mode, identifier transmission is ongoing. 0x7 Header Reception/Transmission In Slave mode, a valid header has been received and Identifier field is available in the BIDR. In Master mode, header transmitted. 0x8 Data Reception/Data Transmission In Receiver mode, reception ongoing. In Transmitter mode, response transmission ongoing. 0x9 Checksum Data transmission/reception completed, checksum transmission/reception ongoing. In UART mode Idle, Init, Sleep, and Data Transmission/Reception states are flagged by the LIN status bits.
22 RMB	Release Message Buffer 0 Buffer data is free. Reset by hardware in when in Initialization mode 1 Buffer data ready to be read by software. This bit should be cleared by software after reading the data received in the buffer.
23 DRBNE	Data Reception Buffer Not Empty Flag This bit is set by hardware as soon as the first byte of response has been received and stored in BDRL (when there is at least one data byte in reception buffer). Software should clear it after reading all the buffers. This bit is also reset by hardware in Initialization mode. This flag could be checked by software in case of a response timeout event.

Table 742. LINSR field descriptions (continued)

Field	Description
24 RXBUSY	Receiver Busy flag 0 Receiver is idle 1 Reception ongoing In Slave mode after header reception, if DIR bit is reset and reception starts, then this bit is set. In this case user cannot set the DTRQ bit.
25 RDI	LIN Receive signal This bit reflects the current status of the Rx pin. **Note:** After reset is released, RDI reflects the actual value of Rx pin.
26 WUF	Wakeup flag This bit is set by hardware when a falling edge is detected on the Rx pin and when: The slave is in Sleep mode The master is in Sleep mode or Idle mode It can be cleared by software. It gets reset by hardware in Initialization mode.
27 DBFF	Data Buffer full flag This bit is set by hardware when the buffer is full (8 bytes received) and DFL > 7. It should be cleared by software. This bit is also reset by hardware in Initialization mode.
28 DBEF	Data Buffer empty flag This bit is set by hardware when the buffer is empty (8 bytes have been transmitted) and DFL > 7. It should be cleared by software after data buffer is refilled by software. Transmission of next eight bytes will not start unless this bit is reset by software. This bit is also reset by hardware in Initialization mode.
29 DRF	Data Reception Completed flag This bit is set by hardware and indicates that data reception has been completed. This flag should be cleared by software. This bit is also reset by hardware in Initialization mode. **Note:** In case framing error or checksum error occurs then this flag is not set.
30 DTF	Data Transmission Completed flag This bit is set by hardware and indicates that data transmission is completed. This flag should be cleared by software. This bit is also reset by hardware in Initialization mode. **Note:** For LIN mode, in case a bit error occurs (and IOBE is reset) then this flag is not set.
31 HRF	Header Received flag This bit is set when the header reception is completed. It should be cleared by software. This bit is set only when: All filters are inactive and Bypass filter (BF) bit is set No match in any filter and Bypass filter (BF) bit is set TX filter match At end of frame or frame aborted, if HRF is still set, it gets reset by hardware. This bit is also reset by hardware in Initialization mode. **Note:** If generic slave = 0, then this bit will always read a 0, and cannot be programmed.

44.4.2.4 LIN Error Status Register (LINESR)

Address: 0x000C Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	SZF	OCF	BEF	CEF	SFEF	SDEF	IDPEF	FEF	BOF	0	0	0	0	0	0	NF
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c							w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 721. LIN error status register (LINESR)

Refer to *Section 44.3.2.5* for detailed description of all errors.

Table 743. LINESR field descriptions

Field	Description
16 SZF	Stuck at Zero flag This bit is set when there is a stuck-at-zero timeout error. It should be reset by software.
17 OCF	Output Compare Flag 0 No output compare event occurred 1 The content of the counter has matched the content of OCR1 or OCR2 This bit should be cleared by software. If this bit is set, MODE bit in LINTCSR is cleared, and the IOT bit of LINTCSR is set, then LINFlexD moves to Idle state. If the MODE bit in LINTCSR is clear, then OCF gets reset by hardware in Initialization mode; if the MODE bit is set, then OCF maintains its status irrespective of the LIN state.
18 BEF	Bit Error flag This bit is set by hardware when there is a bit error. It should be cleared by software. Reset by hardware in Initialization mode.
19 CEF	Checksum Error flag This bit is set by hardware if the received checksum does not match the hardware-calculated checksum. It should be cleared by software. This error will never occur if CCD or CFD bit of LINCR1 is set. Reset by hardware in Initialization mode.
20 SFEF	Sync Field Error flag This bit is set by hardware when the received Sync Field is inconsistent. It should be cleared by software. Reset by hardware in Initialization mode. **Note:** If generic slave = 0, then this bit will always read a 0, and cannot be programmed.

Table 743. LINESR field descriptions (continued)

Field	Description
21 SDEF	Sync Delimiter Error flag This bit is set by hardware when the delimiter is too short (in other words, less than one bit time). It should be cleared by software. Reset by hardware in Initialization mode. **Note:** If generic slave = 0, then this bit will always read a 0, and cannot be programmed.
22 IDPEF	ID Parity Error flag This bit is set by hardware when there is an error in the ID parity. It should be cleared by software. **Note:** Header Error Interrupt is triggered for SFEF or SDEF or IDPEF flag is set and HEIE is set. If generic slave = 0, then this bit will always read a 0, and cannot be programmed. For generic slave = 1,this bit is reset by hardware in Initialization mode.
23 FEF	Framing Error flag This bit is set by hardware when there is a framing error (invalid stop bit). It should be cleared by software. Reset by hardware in initialization mode.
24 BOF	Buffer overrun flag This bit is set by hardware when there is a new byte received and RMB bit is not cleared. It can be cleared by software. Reset by hardware in initialization mode.
31 NF	Noise flag This bit is set by hardware when noise is detected in the received character. It should be cleared by software. Reset by hardware in Initialization mode.

44.4.2.5 UART Mode Control Register (UARTCR)

Address: 0x0010 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	MIS(1)	CSP(1)			OSR(1)				ROSE(1)	NEF(1)(2)			DTU_PCETX(1)	SBUR(1)		WLS(1)
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	TDFL_TFC(2)			RDFL_RFC(3)			RFBM(1)	TFBM(1)	WL1(1)	PC1(1)	RXEN	TXEN	PC0(1)	PCE(1)	WL0(1)	UART(1)
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. This field can be read in any modes and written only in *Initialization* mode.
2. When TFBM = 0, the field is read/write; when TFBM = 1, the field is read-only.
3. When RFBM = 0, the field is read/write; when RFBM = 1, the field is read-only.

Figure 722. UART Mode Control Register (UARTCR)

Table 744. UARTCR field descriptions

Field	Description
0 MIS	Monitor Idle State 1 UARTCTO monitors the idle state of the reception line. 0 UARTCTO monitors the number of bits to be received. The default value of this bit is 0.
1:3 CSP	Configurable Sample Point (i) These bits will decide the sample point during reduced over sampling (ROSE=1). CSP can take the following range of values for a certain over sampling rate: – OSR 4: CSP 2, 3 – OSR 5: CSP 2, 3, 4 – OSR 6: CSP 3, 4, 5 – OSR 8: CSP NA For any values of CSP other than the ones mentioned in the above table, data sampled is erroneous.
4:7 OSR	Over Sampling Rate These bits are programmable by the user to configure the number of samples taken for a bit when reduced over sampling is enabled (ROSE=1). For proper functionality, OSR can only take values 4, 5, 6, and 8. For any other values start bit of data is not detected by receiver and thus the reception will never start.
8 ROSE	Reduced Over Sampling Enable 0 Each bit is over sampled sixteen times. 1 OSR bits decide the over sampling rate. The default value of this bit is 0.
9:11 NEF	Number of expected frame These bits are used to configure the number of expected frames in UART reception mode. If the DTU bit is set, then the UART timeout counter will be reset after the configured number of frames have been received.
12 DTU_PCETX	Disable Timeout in UART mode 0 Timeout has to be handled by software 1 Timeout in UART mode is disabled after the configured number of data frames are received This bit can be programmed in initialization mode only when the UART bit is set. There is an internal counter that gives the number of frames received (num_of_frames). When num_of_frames equals NEF, Disable Timeout is set and num_of_frames gets reset to zero, restarting the timer and enabling the timeout again.
13:14 SBUR	Stop bits in UART reception mode When the UART is used for transmission and reception we have to set the same number of stop bits in GCR and SBUR. When the UART is used only as receiver, it is enough to set SBUR bits only. 00 1 stop bit 01 2 stop bits 10 3 stop bits 11 Reserved This bit can be programmed in the initialization mode only, when the UART bit is set.

Table 744. UARTCR field descriptions (continued)

Field	Description
15 WLS	Special Word Length in UART mode 0 This bit is disabled. 1 This bit enables 12-bit + parity bit in reception (UART mode) for MSC (Micro Second Channel upstream frame) support. This bit can be programmed in initialization mode only when the UART bit is set. Parity check is enabled by default in this mode and PC0/1 bits can be used to select the parity control. This bit is given the highest priority in UART reception mode.
16:18 TDFL_TFC	Transmitter Data Field Length/TX FIFO Counter TDFL_TFC defines the number of bytes to be transmitted in UART buffer mode (TFBM = 0). TDFL_TFC is a read/write field. TDFL_TFC[2] is reserved and not implemented. When the UART data length is configured as a halfword (WL = 10 or WL = 11), only the configurations TDFL_TFC = x01 or TDFL_TFC = x11 are allowed. x00 1-byte x01 2-bytes x10 3-bytes x11 4-bytes TDFL_TFC contains the number of entries (bytes) of the Tx FIFO in UART FIFO mode (TFBM = 1). TDFL_TFCx is a read-only field available for debug purposes. 000 empty 001 1-byte 010 2-bytes 011 3-bytes 100 4-bytes Others: Reserved **Note:** TDFL_TFC can be programmed and are significative only when the UART bit is set.
19:21 RDFL_RFC	Reception Data Field Length/RX FIFO Counter RDFL_RFC defines the number of bytes to be received in UART buffer mode (RFBM = 0). RDFL_RFC is a read/write field. Bit 19 is reserved and not implemented. When the UART data length is configured as a halfword (WL = 10 or WL = 11), only the configurations RDFL_RFC = x01 or RDFL_RFC = x11 are allowed. x00 1-byte x01 2-bytes x10 3-bytes x11 4-bytes RDFL_RFC contains the number of entries (bytes) of the Rx FIFO in UART FIFO mode (RFBM = 1). RDFL_RFCx is a read-only field available for debug purposes. 000 Empty 001 1-byte 010 2-bytes/1.5 byte in case of WLS bit setting 011 3-bytes 100 4-bytes/2×1.5 byte in case of WLS bit setting Others: Reserved RDFL_RFC can be programmed and are significative only when the UART bit is set. **Note:** In buffer mode, RDFL_RFC should be programmed to be greater than or equal to NEF (number of expected frames). In FIFO mode, there is no such constraint.

Table 744. UARTCR field descriptions (continued)

Field	Description
22 RFBM	RFBM Rx FIFO/Buffer mode 0 Rx Buffer mode enabled 1 Rx FIFO mode enabled (mandatory in DMA Rx mode) This bit can be programmed in Initialization mode, only when the UART bit is set.
23 TFBM	Tx FIFO/Buffer mode 0 Tx Buffer mode enabled 1 Tx FIFO mode enabled (mandatory in DMA Tx mode) This bit can be programmed in initialization mode, only when the UART bit is set.
24 WL1	Word Length in UART mode 00 7 bits data + parity 01 8 bits data when PCE = 0 or 8 bits data + parity when PCE = 1 10 15 bits data + parity 11 16 bits data when PCE = 0 or 16 bits data + parity when PCE = 1 This bit can be programmed in Initialization mode only, when the UART bit is set. **Note:** The two-digit binary values are a combination of the fields “WL1” and “WL0” to describe the functionality of word length in UART mode.
25 PC1	Parity Control 00 Parity sent is Even 01 Parity sent is Odd 10 A logical 0 is always transmitted/checked as parity bit 11 A logical 1 is always transmitted/checked as parity bit This bit can be programmed in Initialization mode, only when UART bit is set. **Note:** The two-digit binary values are a combination of the fields “PC1” and “PC0” to describe the functionality of word length in UART mode.
26 RXEN	Receiver Enable 0 Receiver disabled 1 Receiver enabled This bit can be programmed only when the UART bit is set.
27 TXEN	Transmitter Enable 0 Transmitter disabled 1 Transmitter enabled, transmission starts only when this bit is set and Data byte 0 (DATA0) is programmed This bit can be programmed only when UART bit is set.
28 PC0	Parity Control 00 Parity sent is Even 01 Parity sent is Odd 10 A logical 0 is always transmitted/checked as parity bit 11 A logical 1 is always transmitted/checked as parity bit This bit can be programmed in Initialization mode, only when UART bit is set. **Note:** The two-digit binary values are a combination of the fields “PC1” and “PC0” to describe the functionality of word length in UART mode.
29 PCE	Parity Control Enable 0 Parity transmission, reception, and checking disabled 1 Parity transmission, reception, and checking enabled This bit can be programmed in Initialization mode only, when the UART bit is set.

Table 744. UARTCR field descriptions (continued)

Field	Description
30 WL0	Word Length in UART mode 00 7 bits data + parity 01 8 bits data when PCE = 0 or 8 bits data + parity when PCE = 1 10 15 bits data + parity 11 16 bits data when PCE = 0 or 16 bits data + parity when PCE = 1 This bit can be programmed in Initialization mode only, when UART bit is set. **Note:** The two-digit binary values are a combination of the fields “WL1” and “WL0” to describe the functionality of word length in UART mode.
31 UART	UART Mode 0 LIN mode 1 UART mode This bit can be programmed in Initialization mode only.

44.4.2.6 UART Mode Status Register (UARTSR)

Address: 0x0014 Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	SZF	OCF	PE				RMB	FEF	BOF	RDI	WUF	RFNE	TO	DRF_RFE(1)	DTF_TFF(2)	NF
W	w1c	w1c	w1c				w1c	w1c	w1c	w1c	w1c		w1c			w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. When RFBM = 0, the field can be read/cleared by software. Writing 1 clears contents; when RFBM = 1, the field is read-only.
2. When TFBM = 0, the field can be read/cleared by software. Writing 1 clears contents; when TFBM = 1, the field is read-only.

Figure 723. UART Mode Status Register (UARTSR)

Table 745. UARTSR field descriptions

Field	Description
16 SZF	Stuck at Zero flag This bit is set by hardware when 100 dominant bits are detected. It should be cleared by software. An interrupt will be generated if the SZIE bit in LINIER is set. This bit will reflect the same value as in LINESR, when in Initialization mode and the UART bit is set.
17 OCF	Output Compare Flag 0 No output compare event occurred 1 The content of the counter has matched the content of LINOCR An interrupt will be generated if the OCIE bit in LINIER is set. This bit will reflect the same value as in LINESR, when in Initialization mode and the UART bit is set. For Baud rate above 1 Mbit/s, this flag is not usable.
18:21 PE	Parity Error flag These bits indicate if there is a Parity Error in the corresponding byte. No interrupt is generated if this error occurs. PE[i] (where i = 0,1,2 or 3) can be equal to: 0 No parity error 1 Parity error in the corresponding received byte This bit will reflect the same value as in LINESR, when in initialization mode and UART bit set. **Note:** Parity is checked only after a complete frame is received. When WL bits = 10 or 11, either PE[0] or PE[2] is only set. – When PE[0] is set, it indicates Parity error in either first or second byte received. – When PE[2] is set, it indicates Parity error in either third or fourth byte received.
22 RMB	Release Message Buffer 0 Buffer data is free 1 Buffer data ready to be read by software This bit should be cleared by software. This bit will reflect the same value as in LINESR, when in Initialization mode and the UART bit is set.
23 FEF	Framing Error flag This bit is set by hardware when there is a framing error (invalid stop bit). It should be cleared by software. It will generate an interrupt if the FEIE bit of LINIER is set. This bit will reflect the same value as in LINESR, when in Initialization mode and the UART bit is set.
24 BOF	FIFO/Buffer overrun flag This bit is set by hardware when there is a new byte received and the RMB bit is not cleared in UART buffer mode. In UART FIFO mode this bit is set when there is a new byte and the Rx FIFO is full. In UART FIFO mode, once Rx FIFO is full, the new received message will be discarded irrespective of the value of the RBLM bit. In UART Rx Buffer mode, if RBLM is set then the new message received will be discarded; if RBLM is reset then the new message will overwrite the buffer. It can be cleared by software writing a 1. An interrupt is generated if the BOIE bit of LINIER is set. This bit will reflect the same value as in LINESR, when in Initialization mode and the UART bit is set.

Table 745. UARTSR field descriptions (continued)

Field	Description
25 RDI	Receiver Data Input signal This bit reflects the current status of the RX pin.
26 WUF	Wakeup flag This bit is set by hardware when a falling edge is detected on the RX pin in sleep mode. It should be cleared by software. An interrupt will be generated if the WUIE bit in LINIER is set. This bit will reflect the same value as in LINESR when in Initialization mode and the UART bit is set.
27 RFNE	Receive FIFO Not Empty RFNE bit is set by hardware in UART FIFO mode (RFBM = 1), when there is at least one data byte present in the receive FIFO. RFNE is a read-only bit for debugging purposes. This flag can be used by software in case of a timeout event.
28 TO	Timeout This bit is set by hardware when a UART timeout occurs — in other words, the value of UARTCTO becomes equal to the preset value of the timeout (UARTPTO register setting). TO should be cleared by software.The SR bit should be used to reset the receiver fsm to Idle state in case of UART TIMEOUT for UART reception, depending on the application, in both buffer and FIFO mode. An interrupt will be generated when LINIER.DBEIE/TOIE bit is set on the Error interrupt line in UART mode.
29 DRF_RFE	Data Reception Completed Flag/Rx FIFO Empty Flag DRF_RFE is set by hardware in UART buffer mode (RFBM = 0) and indicates that the number of bytes programmed in RDFL have been received. DRF_RFE should be cleared by software. An interrupt will be generated if the DRIE bit in LINIER is set. DRF_RFE bit is set only when the configured number of stop bits are received for the last frame (number of frames is configurable by RDFL bits). DRF_RFE is set irrespective of framing error, parity error or overrun error. This bit will reflect the same value as in LINESR when in Initialization mode and the UART bit is set. DRF_RFE is set by hardware in UART FIFO mode (RFBM = 1) when the RX FIFO is empty. DRF_RFE is a read-only bit for debugging purposes. It is internally used by the DMA RX interface.
30 DTF_TFF	Data Transmission Completed Flag/TX FIFO Full Flag DTF_TFF is set by hardware in UART buffer mode (TFBM = 0) and indicates that data transmission is completed. DTF_TFF should be cleared by software. An interrupt will be generated if the DTIE bit in LINIER is set. This bit will reflect the same value as in LINESR when in Initialization mode and the UART bit is set. DTF_TFF is set by hardware in UART FIFO mode (TFBM = 1) when TX FIFO is full. DTF_TFF is a read-only bit for debugging purposes. It is internally used by the DMA TX interface.
31 NF	Noise flag This bit is set by hardware when noise is detected in the received character. It should be cleared by software. This bit will reflect the same value as in LINESR when in Initialization mode and the UART bit is set. During reduced oversampling (ROSE bit = 1), it is enabled only when OSR = 8.

44.4.2.7 LIN Time-Out Control Status Register (LINTCSR)

This register contains control and status bits for the timeout feature.

Address: 0x0018 Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	MODE(1)	IOT(1)	TOCE	CNT							
W																
Reset	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0

1. This field can be read in any modes and written only in *Initialization* mode.

Figure 724. LIN Time-Out Control Status Register (LINTCSR)

Table 746. LINTCSR field descriptions

Field	Description
21 MODE	Time-out counter mode 0 LIN mode 1 Output compare mode This bit can be configured only during initialization.
22 IOT	Idle on timeout 0 LIN state machine does not reset to Idle on timeout 1 LIN state machine resets to Idle on timeout event This feature is applicable only when MODE bit in LINTCSR is cleared.
23 TOCE	Time-out counter enable 0 Time-out counter disable. OCF flag is not set on an output compare event. 1 Time-out counter enable. OCF flag is set if an output compare event occurs. TOCE is always configurable by software in Initialization mode. If LIN state is other than INIT and if timer is configured in LIN mode, then hardware takes control of TOCE.
24:31 CNT[7:0]	Counter Value These bits reflect the value of a free-running counter used for timeout. **Note:** For proper functionality of this counter, LINIBRR should be greater than or equal to 5. This counter increments once in every bit duration, as per the programmed bit rate. The counter starts as soon as the IP comes out of reset and increments based on the slowest baud rate possible (LINIBRR = FFFFFh), until the user programs LINIBRR and LINFBRR with the desired values.

44.4.2.8 LIN Output Compare Register (LINOCR)

This register contains the value to be compared to the count value in LINTOCR register. This register is writable by software only in Output Compare Mode.

Address: 0x001C Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	OC2								OC1							
W																
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Figure 725. LIN Output Compare Register (LINOCR)

Table 747. LINOCR field descriptions

Field	Description
16:23 OC2	Output compare value 2 When MODE bit = 0 in LINTCSR register: This field is loaded by HW, with response timeout calculated based on RTO value in LINTOCR register and CNT value in LINTCSR register. When MODE bit = 1 in LINTCSR register: This field can be used as output compare register, upto bit rates of 1 Mbit/s.
24:31 OC1	Output compare value 1 When MODE bit = 0 in LINTCSR register: This field is loaded by HW, with header timeout calculated based on HTO value in LINTOCR register and CNT value in LINTCSR register. When MODE bit = 1 in LINTCSR register: This field can be used as output compare register, upto bit rates of 1 Mbit/s. **Note:** OC1 field can be used as output compare register only in LIN slave mode.

44.4.2.9 LIN Time-Out Control Register (LINTOCR)

Address: 0x0020 Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	RTO				0	HTO						
W																
Reset	0	0	0	0	1	1	1	0	0	0	0(1)/1	1(1)/0	1	1	0	0

1. This reset value is valid only in Master mode.

Figure 726. LIN Time-Out Control Register (LINTOCR)

Table 748. LINTOCR field descriptions

Field	Description
20:23 RTO	Response timeout value This is the response timeout duration (in bit time) for 1 byte. The reset value is 0xE = 14, corresponding to $T_{Response_Maximum} = 1.4 * T_{Response_Nominal}$
25:31 HTO	Header timeout value This register contains the header timeout duration (in bit time). This register can be written only for Slave mode. The reset value is 0x2C = 45, corresponding to $T_{Header_Maximum}$ in the LIN specification. If Master mode is enabled then these bits will always reflect 0x1C = 28 (1.4 × 10 bits of sync + 1.4 × 10 bits of ID). For slave, these bits should be programmed without considering 11 bits of break.

44.4.2.10 LIN Fractional Baud Rate Register (LINFBRR)

This register consists of bits that decide the fractional part of the LIN Baud Rate. It can be programmed only in Initialization mode.

Note: When LASE bit is set, this register should be read only after AUTOSYNC_COMP bit in LINSR register is set to obtain the correct value.

This register cannot be used when reduced oversampling is enabled (ROSE bit = 1)

Address: 0x0024 Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	FBR(1)			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. This field can be read in any modes and written only in *Initialization* mode.

Figure 727. LIN Fractional Baud Rate Register (LINFBRR)

Table 749. LINFBRR field descriptions

Field	Description
28:31 FBR	Fractional Baud rates 0x0 Fraction (LDIV): 0 0x1 Fraction (LDIV): 1/16 0x2 Fraction (LDIV): 2/16 0x3 Fraction (LDIV): 3/16 0x4 Fraction (LDIV): 4/16 0x5 Fraction (LDIV): 5/16 0x6 Fraction (LDIV): 6/16 0x7 Fraction (LDIV): 7/16 0x8 Fraction (LDIV): 8/16 0x9 Fraction (LDIV): 9/16 0xA Fraction (LDIV): 10/16 0xB Fraction (LDIV): 11/16 0xC Fraction (LDIV): 12/16 0xD Fraction (LDIV): 13/16 0xE Fraction (LDIV): 14/16 0xF Fraction (LDIV): 15/16

44.4.2.11 LIN Integer Baud Rate Register (LINIBRR)

This register consists of control bits that decide the baud rate along with the LINFBRR. It can be programmed only in Initialization mode.

Note: When LASE bit is set, this register should be read only after AUTOSYNC_COMP bit in LINSR register is set to obtain the correct value.

Address: 0x0028 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	IBR(1)			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	IBR(1)															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. This field can be read in any modes and written only in *Initialization* mode.

Figure 728. LIN Integer Baud Rate Register (LINIBRR)

Table 750. LINIBRR field descriptions

Field	Description
12:31 IBR	Integer Baud rates These bits along with the fractional baud rate bits decide the LIN baud rate. 0x00000 Mantissa (LDIV): LIN clock disabled 0x00001 Mantissa (LDIV): 1 ... 0xFFFFE Mantissa (LDIV): 1048574 0xFFFFF Mantissa (LDIV): 1048575

44.4.2.12 LIN Checksum Field Register (LINCFR)

*Note: There is a delay between 4 to 6 clock cycles of PBRIDGEx_CLK for the internal checksum's value (which is clocked with LIN_CLK/16 * LDIV) to reflect on LINCFR.*

This register consists of checksum bits.

Address: 0x002C Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	CF							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 729. LIN Checksum Field Register (LINCFR)

Table 751. LINCFR field descriptions

Field	Description
24:31 CF	Checksum bits When the LINCR1[CCD] bit is reset these bits are read-only and are calculated by hardware. When the LINCR1[CCD] bit is set, these bits can be written by software.

LINCR1[CFD]	LINCR1[CCD]	LINCHKSUM Read/write	Checksum sent
1	1	read/write	None
1	0	read-only	None
0	1	read/write	Programmed checksum
0	0	read-only	Calculated checksum

44.4.2.13 LIN Control Register 2 (LINCR2)

This register includes control status bits related to buffer operations.

Address: 0x0030 Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TBDE(1)	IOBE(1)	IOPE(1)	WURQ(2)	DDRQ(2)	DTRQ(2)	ABRQ(2)	HTRQ(2)	0	0	0	0	0	0	0	0
W																
Reset	0	1	0(3)/1	0	0	0	0	0	0	0	0	0	0	0	0	0

1. This field can be read in any modes and written only in Initialization mode.
2. This field can be read/set by software.
3. When slave = 0, reset value is '0' and the this field cannot be programmed.

Figure 730. LIN Control Register 2 (LINCR2)

Table 752. LINCR2 field descriptions

Field	Description
16 TBDE	Two Bit delimiter bit 0 Delimiter length in break field is 1 bit 1 Delimiter length in break field is 2 bits This bit can be set in Initialization mode only.
17 IOBE	Idle on Bit Error 0 Bit Error does not reset LIN state machine 1 Bit Error resets LIN state machine This bit can be set in Initialization mode only.
18 IOPE	Idle on Identifier Parity Error 0 Parity Error does not reset LIN state machine 1 Parity Error resets LIN state machine This bit can be set in Initialization mode only. **Note:** If generic slave = 0, then this bit will always read a 0, and cannot be programmed.
19 WURQ	Wakeup Generate Request Setting this bit will generate a wakeup pulse. It is reset by hardware when the wakeup character has been transmitted. The character sent during wakeup is copied from BDRL (DATA0). Note that this bit cannot be set in Sleep mode — software has to exit Sleep mode before setting this bit. Bit Error is not checked when transmitting the wakeup request.
20 DDRQ	Data Discard request Set by software to stop data reception if the frame does not concern the node. This bit is reset by hardware once LINFlexD ignores the response and moves to Idle state. For LIN slave this bit can be set only when HRF bit is set and Identifier is software-filtered.
21 DTRQ	Data Transmission Request Set by software in slave mode to request the transmission of the LIN Data field stored in the Buffer data register. This bit can be set only when the HRF bit is set (to ensure that data transmission is requested only after a header reception). Cleared by hardware when the request has been completed, or on abort request or error condition. In Master mode, this bit is set by hardware when the DIR bit is set and header transmission is complete.
22 ABRQ	Abort Request Set by software to abort the current transmission. Cleared by hardware when the transmission has been aborted. LINFlexD aborts the transmission at the end of the current bit. This bit can abort a wakeup request also and can be used in UART mode also.
23 HTRQ	Header Transmission Request Set by software to request the transmission of the LIN Header. Cleared by hardware when the request has been completed or on abort request. This bit has no effect in UART mode. **Note:** In master mode, if both HTRQ and ABRQ are set at the same time then ABRQ has no effect. Similarly, in slave mode after header reception, if DTRQ and ABRQ are simultaneously set then ABRQ has no effect.

44.4.2.14 Buffer Identifier Register (BIDR)

This register contains the bits which provide information about the identifier of the transaction and other related information.

Note: All the fields (ID, CSS, DIR, DFL) of the BIDR register must be updated when an ID filter (enabled) in Slave mode (Tx or Rx) matches the ID received.

Address: 0x0034 Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	DFL						DIR	CCS	0	0	ID					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 731. Buffer Identifier Register (BIDR)

Table 753. BIDR field descriptions

Field	Description
16:21 DFL	Data Field Length Number of data bytes in the response part of the frame. DFL[5:0] = Number of data bytes – 1 **Note:** DFL[5:3] is provided for extended frames. Normally LIN mode will use DFL[2:0]. Identifier filters are now compatible with DFL[5:0] also.
22 DIR	Direction This bit controls the direction of the data field. 0 LINFlexD receives the data and copies them in the BDRs 1 LINFlexD transmits the data from the BDRs
23 CCS	Classic Checksum This bit controls the type of checksum applied on the current message. 0 Enhanced Checksum covering Identifier and Data fields. This is compatible with LIN specification rev. 2.0 and higher. 1 Classic Checksum covering Data field only. This is compatible with LIN specification 1.3 and lower.
26:31 ID	Identifier Identifier part of the identifier field without the identifier parity. This field can be written only in Master mode (MME = ‘1’).

44.4.2.15 Buffer Data Register Least Significant (BDRL)

This register is a part of an 8-byte data buffer.

Address: 0x0038 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	DATA3								DATA2							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	DATA1								DATA0							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 732. Buffer Data Register Least Significant (BDRL)

Table 754. BDRL field descriptions

Field	Description
0:7 DATA3	Data Byte 3 Data byte 3 of the data field
8:15 DATA2	Data Byte 2 Data byte 2 of the data field
16:23 DATA1	Data Byte 1 Data byte 1of the data field
24:31 DATA0	Data Byte 0 Data byte 0 of the data field

44.4.2.16 Buffer Data Register Most Significant (BDRM)

This register is a part of an 8-byte data buffer.

Address: 0x003C Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	DATA7								DATA6							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	DATA5								DATA4							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 733. Buffer Data Register Most Significant (BDRM)

Table 755. BDRM field descriptions

Field	Description
0:7 DATA7	Data Byte 7 Data byte 7 of the data field
8:15 DATA6	Data Byte 6 Data byte 6 of the data field
16:23 DATA5	Data Byte 5 Data byte 5 of the data field
24:31 DATA4	Data Byte 4 Data byte 4 of the data field

44.4.2.17 Identifier Filter Enable Register (IFER)

This register enables/disables a particular filter.

Address: 0x0040 Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	FACT(1)															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. This field can be read in any modes and written only in *Initialization* mode.

Figure 734. Identifier Filter Enable Register (IFER)

Table 756. IFER field descriptions

Field	Description
16:31 FACT	Filter active The software sets the bit FACT[x] to activate the filter x in Identifier list mode. 1. In Identifier mask mode, FACT (2*n*+1) have no effect on the corresponding filters as they act as mask for the Identifier 2*n*. 2. The length of this field depends on the value of no_of_filters. 3. The register diagram depicts the maximum no_of_filters, which can be up to 16. **Note:** Refer to *Chapter 7: Device configuration* to see the number of filters used in the device.

44.4.2.18 Identifier Filter Match Index (IFMI)

This register contains the index corresponding to the received ID. It can be used to read or write the data directly in RAM.

Address: 0x0044 Access: User read

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	IFMI				
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 735. Identifier Filter Match Index (IFMI)

Table 757. IFMI field descriptions

Field	Description
27:31 IFMI	Filter match index Upon a filter match with xth filter – IFMI[4:0] = x+1. On no match IFMI is equal to 00h. This register contains the index corresponding to the received identifier. It can be used to directly write or read the data in SRAM (see *Section 44.3.2.4* for more details).

44.4.2.19 Identifier Filter Mode Register (IFMR)

This register configures the modes of filters.

Address: 0x0048 Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	IFM(1)							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. This field can be read in any modes and written only in *Initialization* mode.

Figure 736. Identifier Filter Mode Register (IFMR)

Table 758. IFMR field descriptions

Field	Description
24:31 IFM[7:0]	Filter mode 0 Filters 2*n* and 2*n*+1 are in identifier list mode 1 Filters 2*n* and 2*n*+1 are in mask mode, where filter 2*n*+1 is the mask for filter 2*n*

44.4.2.20 Identifier Filter Control Register (IFCR2*n*)

This register is read-only in normal mode and can be programmed only in Initialization mode. This register acts as a filter in both Identifier list and Identifier mask modes.

Address: 0x4C+ *n**0x8 (*n* = 0 to (no_of_filters(2)/2–1)) Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	DFL(1)						DIR(1)	CCS(1)	0	0	ID(1)					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. This field can be read in any modes and written only in *Initialization* mode.
2. Refer to device configuration chapter to see the number of filters used in the device

Figure 737. Identifier Filter Control Register (IFCR2n)

Table 759. IFCR2*n* field descriptions

Field	Description
16:21 DFL	Data Field Length Number of data bytes in the response part of the frame. DFL[5:0] = Number of data bytes - 1
22 DIR	Direction This bit controls the direction of the data field. 0 LINFlexD receives data and copies to the BDR registers 1 LINFlexD transmits data from the BDR registers
23 CCS	Classic Checksum This bit controls the type of checksum applied on the current message. 0 Enhanced Checksum covering Identifier and Data fields. This is compatible with LIN specification rev. 2.0 and higher. 1 Classic Checksum covering Data field only. This is compatible with LIN specification 1.3 and lower.
26:31 ID	Identifier Identifier part of the identifier field without the identifier parity.

44.4.2.21 Identifier Filter Control Register (IFCR2*n*+1)

This register is read-only in normal mode and can be programmed only in Initialization mode. This register acts as a filter in Identifier list mode but acts as a mask for IFCR2*n* in Identifier mask mode.

Address: 0x50+ *n**0x8 (*n* = 0 to (no_of_filters(2)/2–1)) Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	DFL(1)						DIR(1)	CCS(1)	0	0	ID(1)					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. This field can be read in any modes and written only in *Initialization* mode.
2. Refer to device configuration chapter to see the number of filters used in the device

Figure 738. Identifier Filter Control Register (IFCR2n+1)

Table 760. IFCR2*n*+1 field descriptions

Field	Description
16:21 DFL	Data Field Length Number of data bytes in the response part of the frame. DFL[5:0] = Number of data bytes – 1
22 DIR	Direction This bit controls the direction of the data field. 0 LINFlexD receives data and copies to the BDR registers 1 LINFlexD transmits data from the BDR registers
23 CCS	Classic Checksum This bit controls the type of checksum applied on the current message. 0 Enhanced Checksum covering Identifier and Data fields. This is compatible with LIN specification rev. 2.0 and higher. 1 Classic Checksum covering Data field only. This is compatible with LIN specification 1.3 and lower.
26:31 ID	Identifier Identifier part of the identifier field without the identifier parity.

44.4.2.22 Global Control Register (GCR)

This register is read-only in Normal mode and can be programmed only in Initialization mode. This is a global control register — in other words, the register configuration will be applied in LIN mode as well as in UART mode.

Address: 0x8C Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	TDFBM(1)	RDFBM(1)	TDLIS(1)	RDLIS(1)	STOP(1)	SR(2)
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. This field can be read in any modes and written only in *Initialization* mode.
2. This field is read 0 by software.

Figure 739. Global Control Register (GCR)

Table 761. GCR field descriptions

Field	Description
26 TDFBM	Transmit data first bit MSB This bit controls the first bit of transmit data (payload only) as MSB/LSB in both UART and LIN modes. 0 The first bit of transmitted data is LSB — in other words, the first bit transmitted is mapped on LSB bit (BDR (0), BDR (8), BDR (16), BDR (24)) 1 The first bit of transmitted data is MSB — in other words, the first bit transmitted is mapped on MSB bit (BDR (7), BDR (15), BDR (23), BDR (31))
27 RDFBM	Received data first bit MSB This bit controls the first bit of received data (payload only) as MSB/LSB both in UART and LIN modes. 0 The first bit of received data is LSB – in other words, the first bit received is mapped on LSB bit (BDR (0), BDR (8), BDR (16), BDR (24)) 1 The first bit of received data is MSB – in other words, the first bit received is mapped on MSB bit (BDR (7), BDR (15), BDR (23), BDR (31))
28 TDLIS	Transmit data level inversion selection This bit controls the data inversion of transmitted data (payload only) in both UART and LIN modes. 0 Transmitted data is not inverted 1 Transmitted data is inverted
29 RDLIS	Received data level inversion selection This bit controls the data inversion of received data (payload only) in both UART and LIN modes. 0 Received data is not inverted 1 Received data is inverted

Table 761. GCR field descriptions (continued)

Field	Description
30 STOP	1/2 stop bit configuration This bit controls the number of stop bit transmitted data in both UART and LIN modes. The stop bit is configured for all the fields (Delimiter, Sync, ID, Checksum, Payload). 0 1 stop bit 1 2 stop bits
31 SR	Soft reset SR executes a soft reset of the LINFlexD controller (FSMs, FIFO pointers, counters, timers, status and error registers) without modifying the configuration registers when a 1 write operation is performed. This bit should be cleared by software to perform further operations (this bit is not cleared by hardware). This bit is always read as 0. The following register fields are reset by SR: LINSR: All fields except RXbusy and AUTOSYNC_COMP LINESR: All fields LINTCSR: Only CNT[0:7] UARTSR: All fields except RFE & TFFt UARTCR: Only TFC & RFC UARTCTO: All fields

44.4.2.23 UART Preset Timeout Register (UARTPTO)

This register contains the preset value of the timeout register in UART mode and is programmed according to the number of bits to be received or to monitor the idle state of the reception line. This register can be written by software any time.

Address: 0x8C + 0x04 Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	PTO											
W																
Reset	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1

Figure 740. UART Preset Timeout Register (UARTPTO)

Table 762. UARTPTO field descriptions

Field	Description
20:31 PTO[11:0]	Preset Timeout PTO defines the preset value of timeout counter. A value of 0, 1 or 2 is forbidden, otherwise the UARTSR[TO] status bit is immediately set. Refer to the UARTCTO register.

44.4.2.24 UART Current Timeout Register (UARTCTO)

This register contains the current timeout value in UART mode, and is used in conjunction with the UARTPTO register (see *Section 44.4.2.23*) to monitor the number of bits received by UART or to monitor the idle state of the reception line. UART timeout works in both CPU and DMA modes.

The timeout counter:

- Starts at zero and counts upward
- Is clocked with LIN_CLK / (16 * LDIV) synchronized to PBRIDGEx_CLK (when ROSE = 0)
- Is clocked with LIN_CLK / (OSR * IBRR) synchronized to PBRIDGEx_CLK (when ROSE = 1)
- Is automatically enabled when UARTCR[RXEN] = 1

*Note: Due to this synchronisation, UARTCTO reflects the internal counter's value (which is clocked with LIN_CLK/16 * LDIV when ROSE = 0 or LIN_CLK/OSR * IBRR when ROSE = 1) which was present on it, before 4 to 6 clock cycles of PBRIDGEx_CLK.*

It is reset when:

- Number of frames received is equal to NEF bits
- UARTCTO becomes equal to UARTPTO-1
- Whenever UARTPTO is written
- When DRF is set and DTU bit = 1

Address: 0x8C + 0x08 Access: User read

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	CTO											
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 741. UART Current Timeout Register (UARTCTO)

Table 763. UARTCTO field descriptions

Field	Description
20:31 CTO	Current Timeout CTO defines the current value of the timeout counter. CTO is a read-only field. CTO is reset every time UARTPTO is re-initialized, or UARTCTO = UARTPTO, or by hard/soft reset. When the CTO value matches the preset value (UARTPTO), the status bit UARTSR[TO] is set.

44.4.2.25 DMA Tx Enable Register (DMATXE)

This register enables the DMA TX interface. This register can be written and read by software any time.

Address: 0x8C + 0x0C Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	DTE															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 742. DMA Tx Enable Register (DMATXE)

Table 764. DMATXE field descriptions

Field	Description
(32–2**TX_CH_NUM):31 DTE[2**TX_CH_NUM–1:0](1)	DMA Tx channel Y enable 0 DMA Tx channel Y disabled 1 DMA Tx channel Y enabled **Note:** The actual size of the register DMATXE depends on the value of the static parameter TX_CH_NUM. When DMATXE = 0x00000000, the DMA TX interface FSM is forced (soft reset) in Idle state. Refer to the device configuration chapter for the value of TX_CH_NUM used in this device.

1. ** stands for exponentiation.

44.4.2.26 DMA Rx Enable Register (DMARXE)

This register enables the DMA RX interface. This register can be written and read by software any time.

Address: 0x8C + 0x10 Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	DRE															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 743. DMA Rx Enable Register (DMARXE)

Table 765. DMARXE field descriptions

Field	Description
(32–2**RX_CH_NUM):31 DRE[2**RX_CH_NUM–1:0](1)	DMA Rx channel Y enable 0 DMA Rx channel Y disabled 1 DMA Rx channel Y enabled **Note:** The actual size of the register DMARXE depends on the value of the static parameter RX_CH_NUM. When DMARXE = 0x00000000, the DMA RX interface FSM is forced (soft reset) in Idle state. Refer to the device configuration chapter for the value of RX_CH_NUM used in this device.

1. ** stands for exponentiation.

44.5 Programming considerations

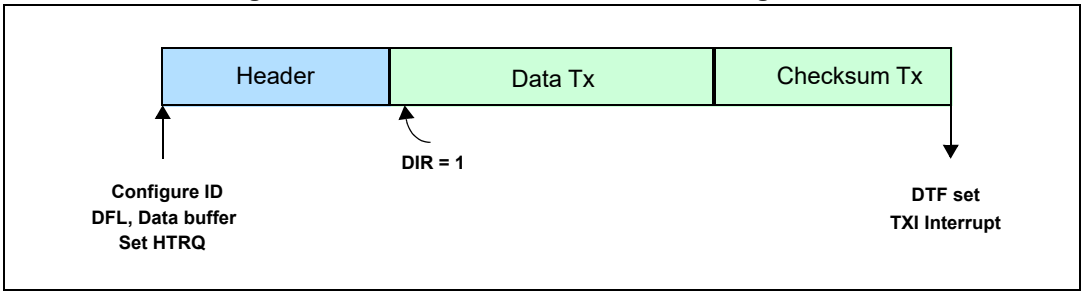
The next subsections describe the various configurations in which the LINFlexD module can be used.

44.5.1 Master node

LINFlexD acts as Master when the MME bit is set.

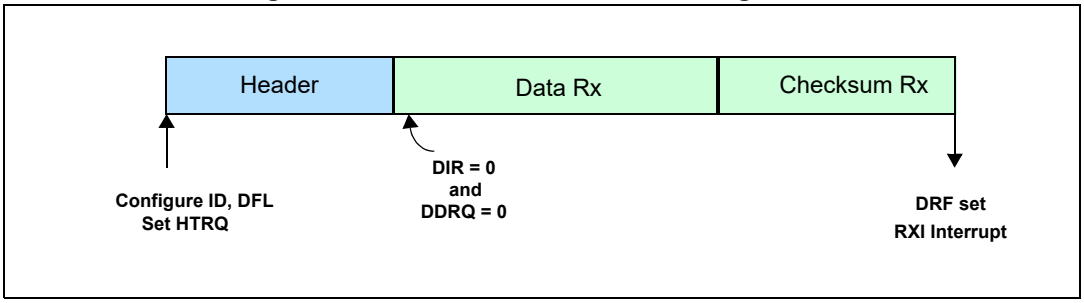
44.5.1.1 Transmitter

Figure 744. Master node: transmitter configuration



44.5.1.2 Receiver

Figure 745. Master node: receiver configuration



44.5.1.3 Transmitter, Bit Error

Figure 746. Master node: transmitter, bit error configuration

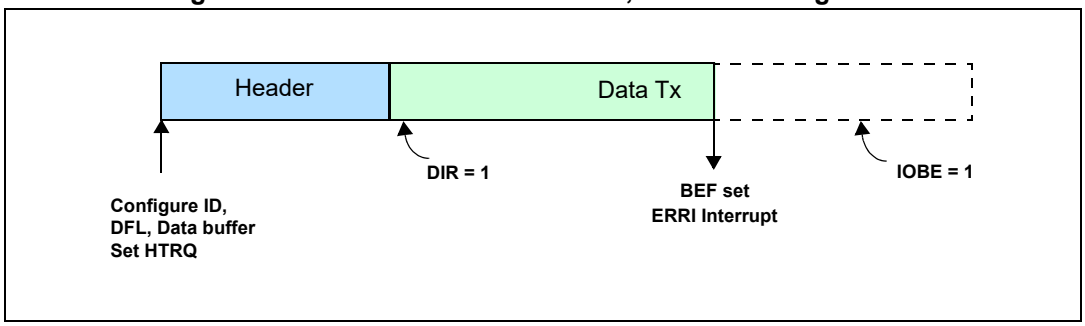
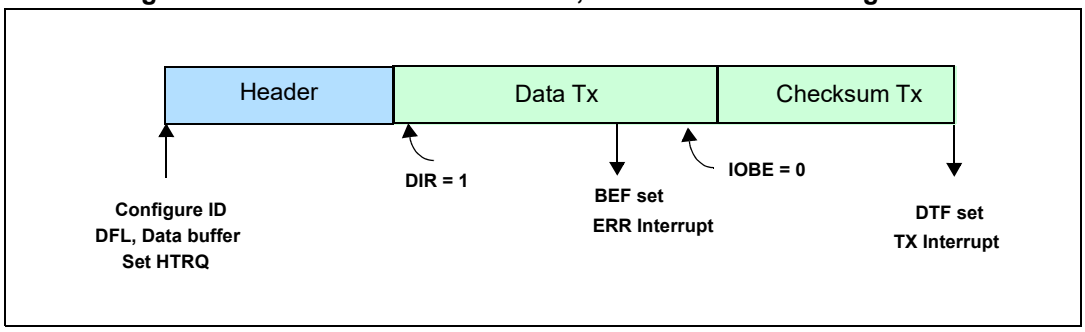
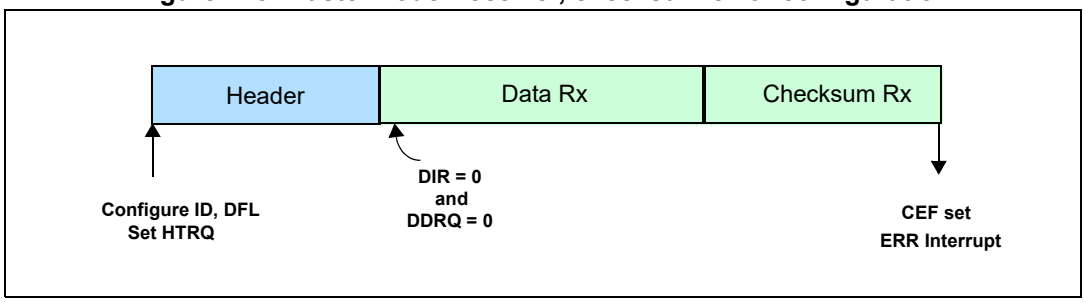


Figure 747. Master node: transmitter, checksum error configuration



44.5.1.4 Receiver, Checksum Error

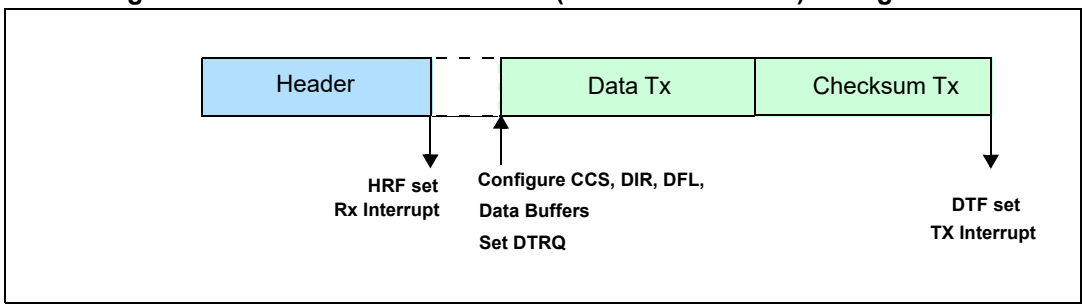
Figure 748. Master node: receiver, checksum error configuration



44.5.2 Slave node

44.5.2.1 Transmitter (no identifier filters)

Figure 749. Slave node: transmitter (no identifier filters) configuration



44.5.2.2 Receiver (no identifier filters)

Figure 750. Slave node: receiver (no identifier filters) configuration

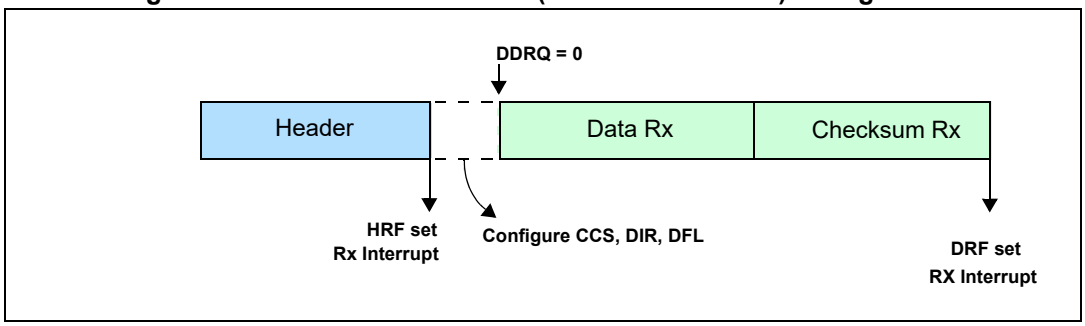
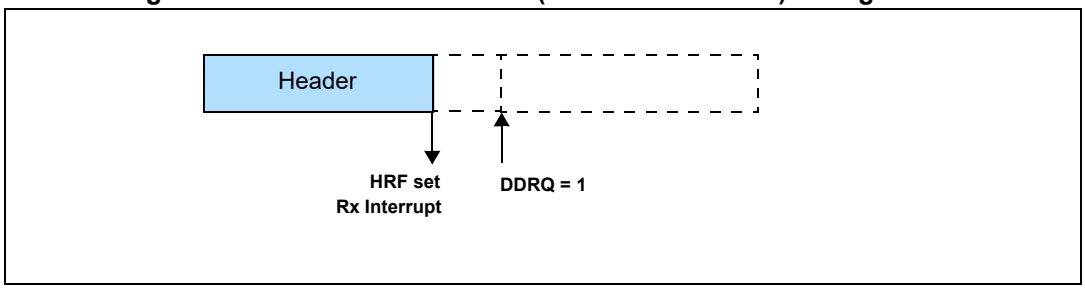
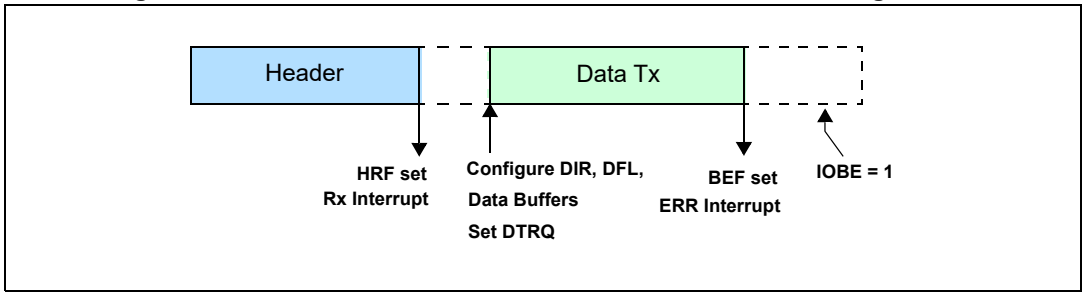


Figure 751. Slave node: receiver (no identifier filters) configuration



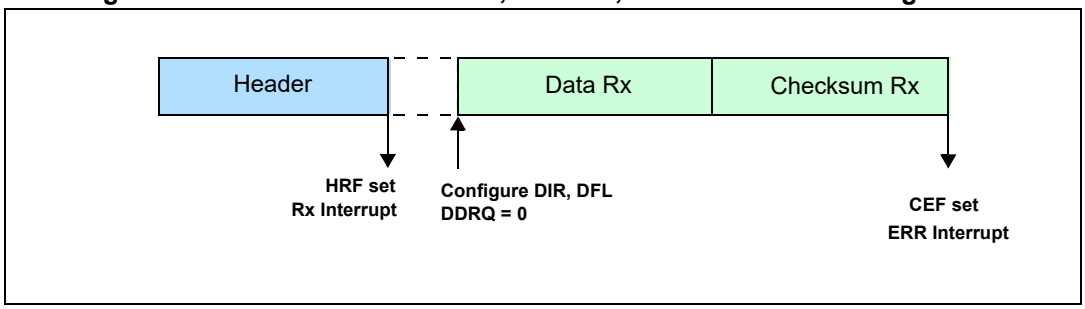
44.5.2.3 No filters, transmitter, bit error

Figure 752. Slave node: No filters, transmitter, bit error configuration



44.5.2.4 No filters, receiver, checksum error

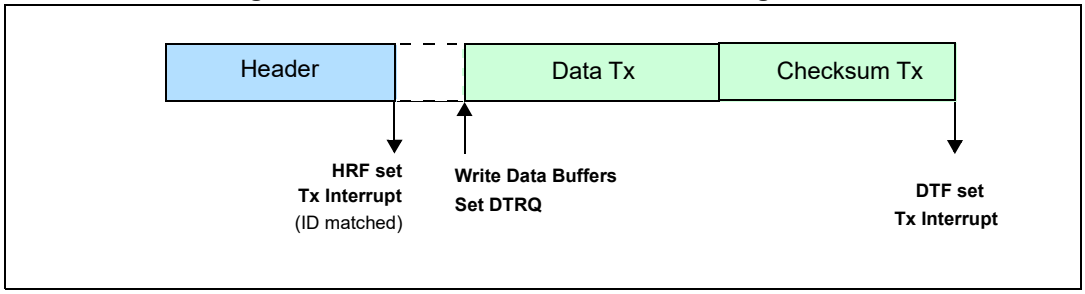
Figure 753. Slave node: No filters, receiver, checksum error configuration



44.5.2.5 At least one TX filter, BF is reset, ID matches filter

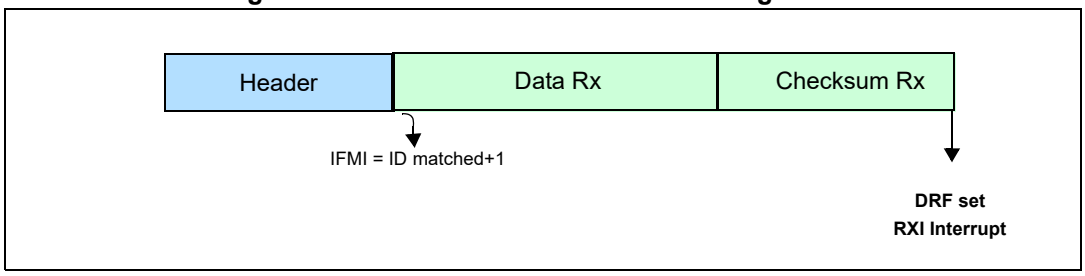
This configuration can be used in case slave never receives data, for example, sensor.

Figure 754. Slave node: one TX filter configuration



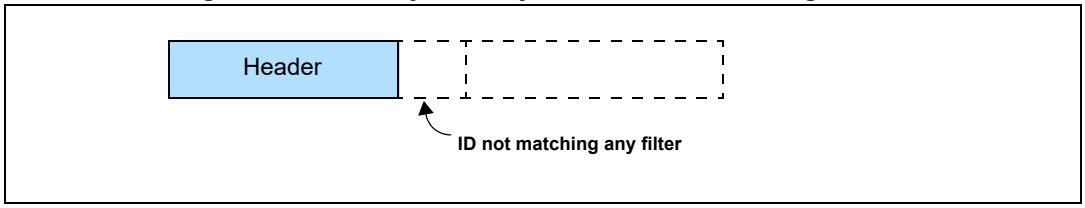
44.5.2.6 At least one RX filter, BF reset, ID matches filter

Figure 755. Slave node: one RX filter configuration



44.5.2.7 RX only, TX only, RX & TX filters, ID not matching filter, BF reset

Figure 756. RX only, TX only, RX & TX filters configuration



44.5.2.8 TX filter, BF is set

This configuration is used when:

- All TX IDs are handled by filters.
- There are not enough other filters to handle all reception IDs.

Figure 757. TX filter, BF is set configuration: ID has matched

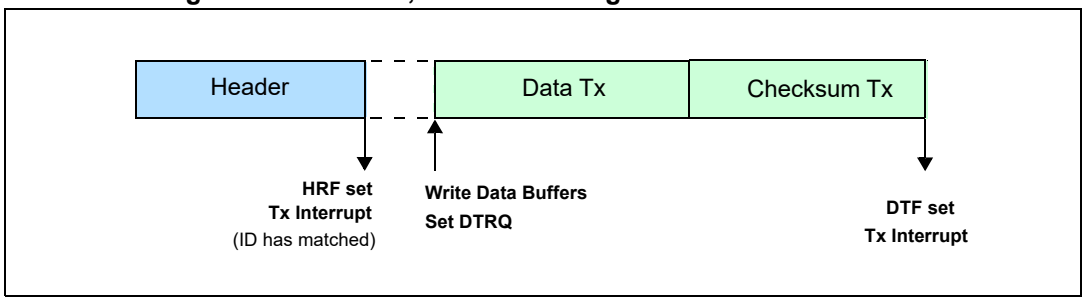
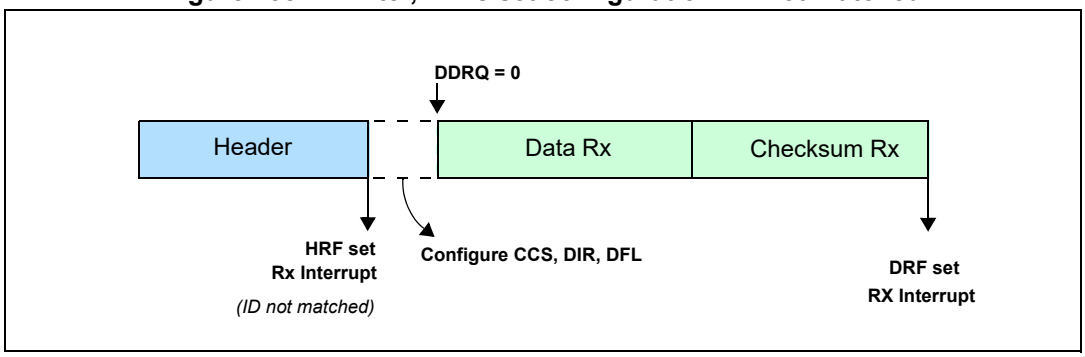


Figure 758. TX filter, BF is set configuration: ID not matched



44.5.2.9 RX filter, BF is set

Figure 759. RX filter, BF is set configuration: ID matched

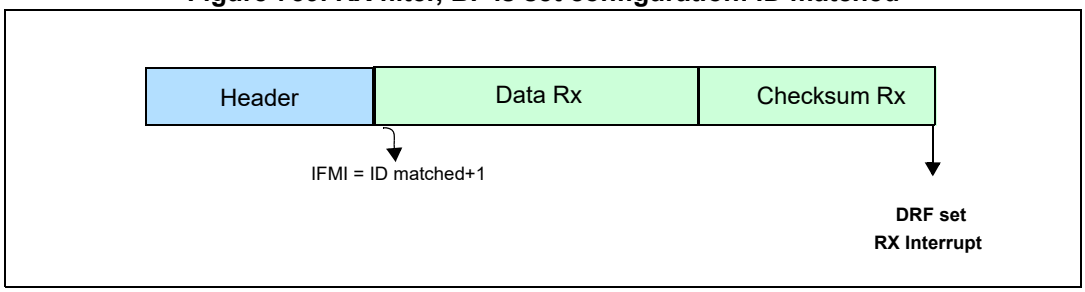


Figure 760. RX filter, BF is set configuration: ID not matched (ID is Rx)

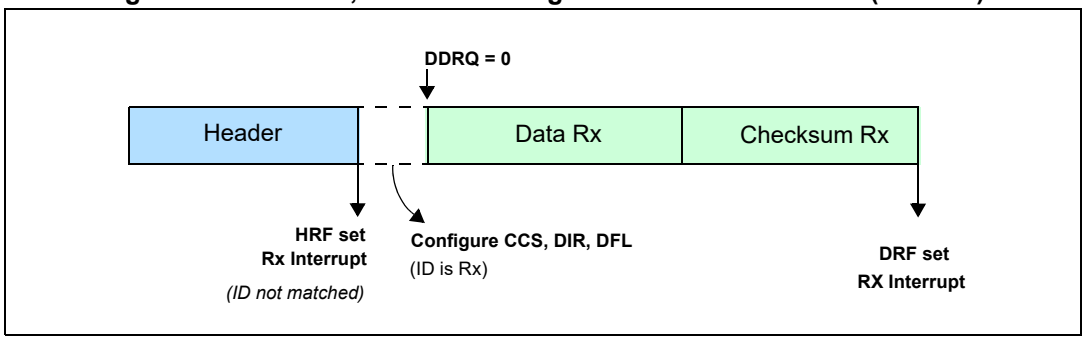
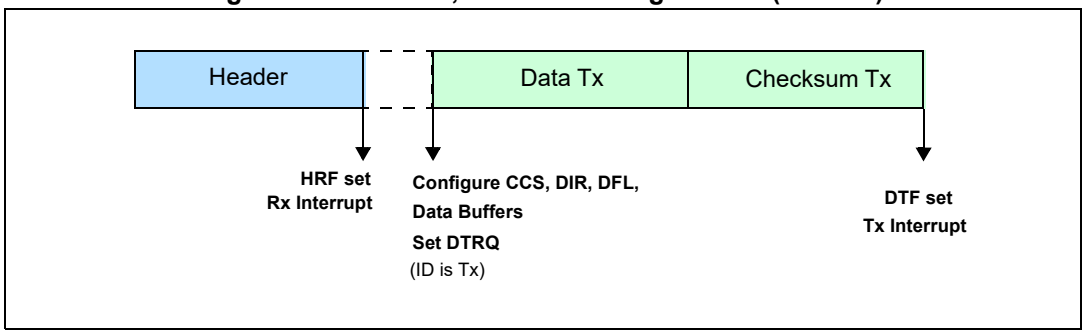


Figure 761. RX filter, BF is set configuration: (ID is Tx)



44.5.2.10 TX filter, RX filter, BF set

This configuration is used when:

- There are not enough filters.
- The filters are used for most frequently used IDs to reduce CPU usage.

Figure 762. TX filter, RX filter, BF set configuration: (ID matched)

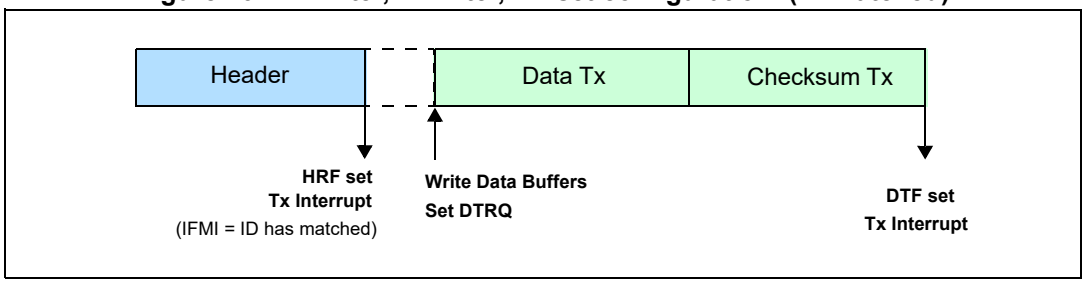


Figure 763. TX filter, RX filter, BF set configuration: (ID matched + 1)

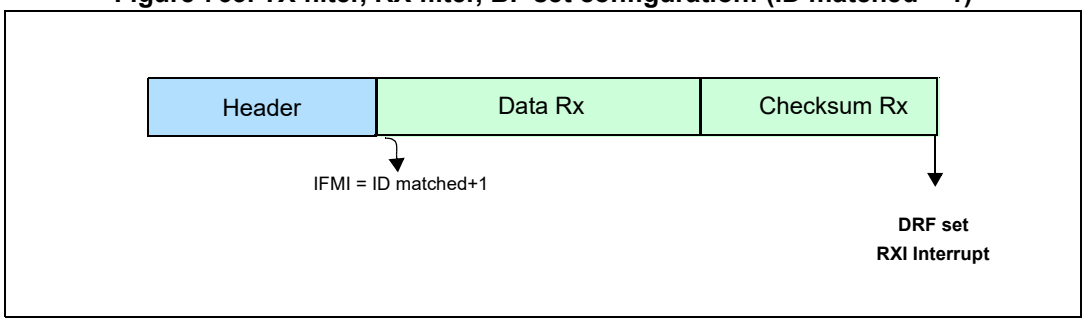
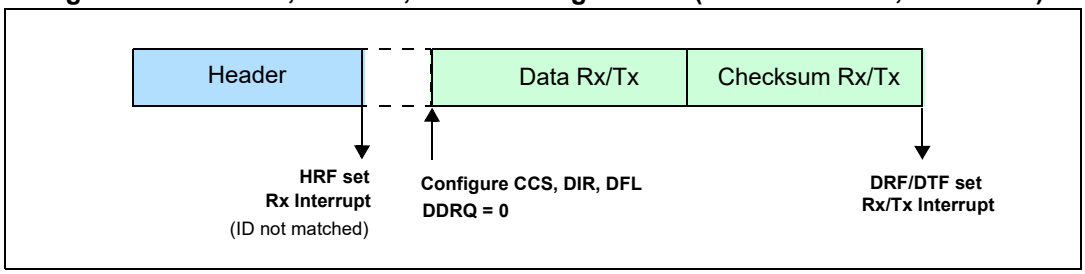


Figure 764. TX filter, RX filter, BF set configuration: (ID not matched, RX/TX Int.)



44.5.3 Extended frames

Figure 765. Extended frames (TX Interrupt)

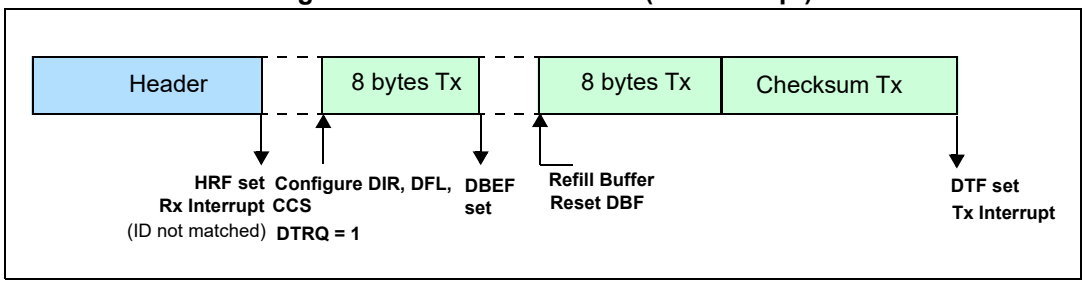
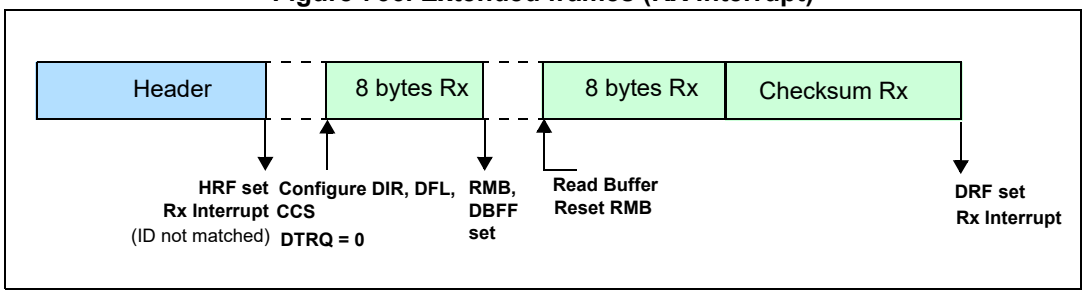


Figure 766. Extended frames (RX Interrupt)



44.5.4 Timeout

Master Node: Response (during reception only) and frame timeout are checked.

Slave Node: Header, Response (during reception only), and frame timeout are checked.

Figure 767. Response timeout

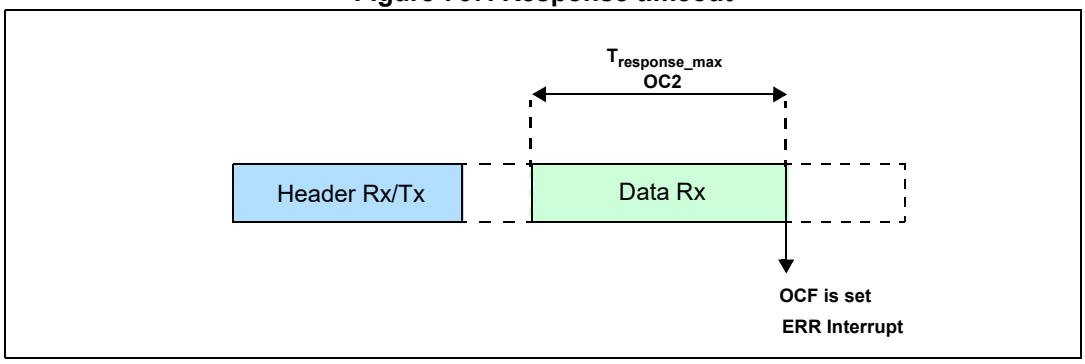
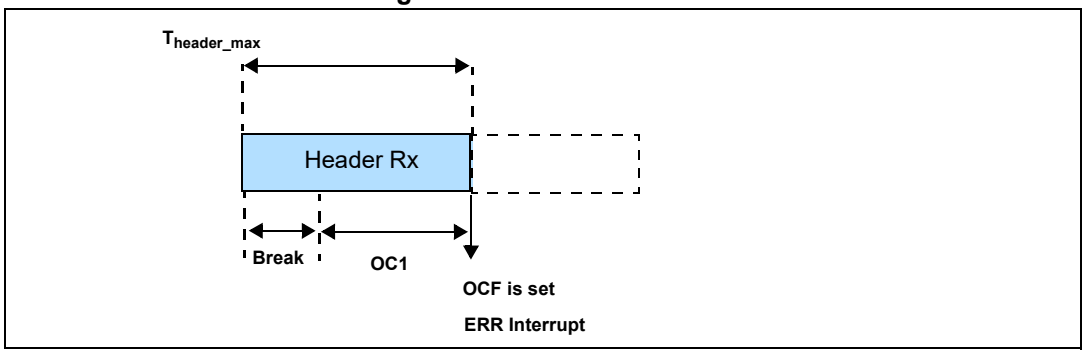
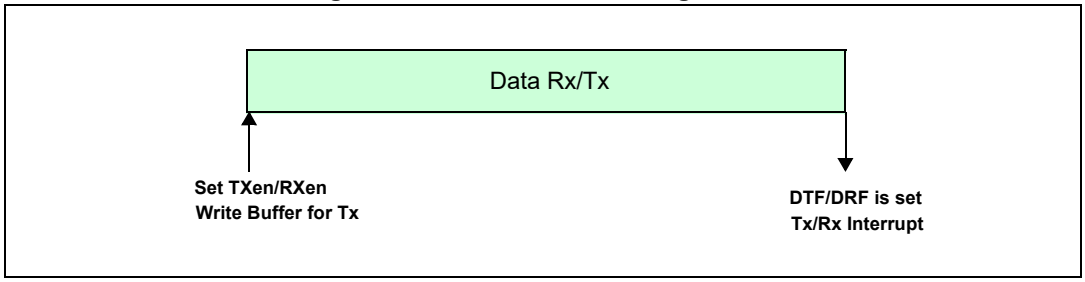


Figure 768. Header timeout



44.5.5 UART mode

Figure 769. UART mode configuration



44.5.6 Interrupts

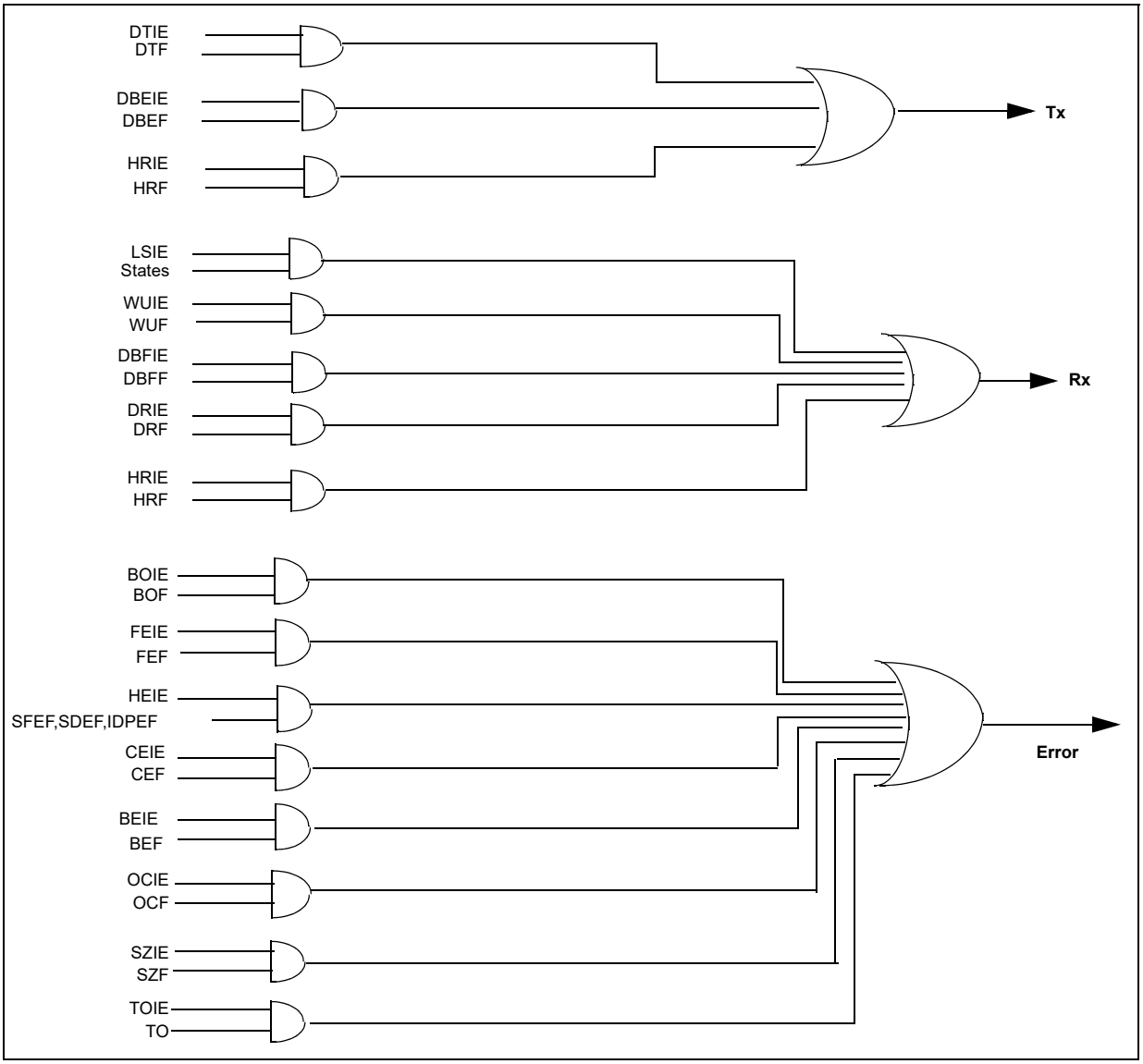
Table 766. Interrupts

Interrupt event	Event flag	Enable control bit	Interrupt vector
Stuck at zero	SZF	SZIE	Error
Output compare	OCF	OCIE	Error
Bit error	BEF	BEIE	Error
Checksum error	CEF	CEIE	Error
Header error	SFEF or SDEF or IDPEF	HEIE	Error
Frame error	FEF	FEIE	Error
Buffer overrun error	BOF	BOIE	Error
UART timeout error	TO	TOIE	Error
LIN state	Sync Del, Sync Field, Identifier Field, or Checksum Field	LSIE	Rx
Wakeup	WUF	WUIE	Rx
Data buffer full	DBFF	DBFIE	Rx

Table 766. Interrupts (continued)

Interrupt event	Event flag	Enable control bit	Interrupt vector
Data buffer empty	DBEF	DBEIE	Tx
Data reception complete	DRF	DRIE	Rx
Data transmitted	DTF	DTIE	Tx
Header received	HRF	HRIE	Rx
Header received	HRF	HRIE	Tx (if there is a filter match for Tx identifier)

Figure 770. Interrupt diagram



45 SENT Receiver (SRX)

45.1 Introduction

The Single Edge Nibble Transmission (SENT) Receiver (SRX) module is a multi-channel receiver for receiving serial data frames which are being transmitted by a sensor implementing the SENT encoding scheme and present them to the CPU for further processing.

This module is based on the SAE J2716 information report titled “SENT - Single Edge Nibble Transmission for Automotive Applications” and released on January 27, 2010 (SAE web site), Apr2007 and Feb2008. As per this standard, the SENT protocol is intended for use in applications where high resolution sensor data needs to be communicated from a sensor to an Engine Control Unit (ECU). It is intended as a replacement for the lower resolution methods of 10 bit A/Ds and PWM and as a simpler low-cost alternative to CAN or LIN. The implementation assumes that the sensor is a smart sensor containing a microprocessor or dedicated logic device (ASIC) to create the signal.

The SENT encoding scheme is a unidirectional communications scheme from the sensor / transmitting device to the controller /receiving device which does not include a coordination signal from the controller/receiving device. The sensor signal is transmitted as a series of pulses with data encoded as falling to falling edge periods.

45.1.1 Features

The SENT receiver module has the following features:

- The number of SENT channels supported are device-specific. See *Section 7.9.4: SENT Receiver (SRX) configuration* for details.
- Unified message read logic to transfer all messages received to system memory via interrupts or DMA
  - Programmable option per channel to support reading of messages via DMA or Interrupt
  - Internal round robin arbitration sequencer to loop across channels to read messages from each channel leaving CPU/DMA transparent to this process
  - All received messages are stored excluding the synchronization pulse (for Fast messages) and identifiers (for Slow Serial Messages)
  - Separate buffers for storing Fast and Slow Serial Messages
  - Separate DMA and Interrupts generated for Fast and Slow Serial Messages
- Supports compensation for variation in SENT Tx Clock up to ±25% and adjusts its measurement for drift in the Transmitter and Receiver clocks per channel
  - Calibration pulse correction factor per channel is available for use by software
- Implements for each channel all receiver diagnostics as specified by the SAE Specifications
- Internal 4-bit CRC calculator that can be configured to compute CRC in both Legacy and Recommended method (as given in SAE SENT Specification) for both Fast and Short Serial Message per channel
- Internal 6-bit CRC calculator for Enhanced Serial Messages per channel
- Supports all bit rates by having per channel configurable Rx clock periods from 1.27 µs to 90 µs
- Support for configurable number of data nibbles per channel
- Option to disable individual channel via programmable control bit
- Time stamping on start of each valid message (such as, without errors) across all channels. This time stamp is appended to each message for synchronization of messages and for application to understand how much time has elapsed between successive sensor readings
- Support for pause pulse per channel. This can be enabled by programming a bit in the control register for that channel
- Input Programmable Filter on each channel to filter any glitches on input. This filter duration is programmable by the user software.
- 32-bit register interface for programming the module's registers and reading the received messages
- Works on two clocks, one for accurate receive operations (high frequency receiver clock or protocol clock) and other for message read logic (system bus clock) and the module can work irrespective of the frequency relationship between the two clocks. See *Section 45.5: Clocks and resets* for more details on the clocks.

45.1.2 Modes of operation

45.1.2.1 Debug mode

In debug mode, message reception will be halted and all existing status and messages will be retained. DBG_FRZ bit is required to be set for this mode. On debug exit, a calibration length error (CAL_LEN_ERR) can be reported and the channels will start receiving messages from a valid calibration pulse.

45.1.2.2 Low power modes

SENT Receiver behavior is affected by chip-specific low power modes (for example, STOP or WAIT modes). These low power modes should either gate-off both the system bus clock and the protocol clocks or reduce their frequencies, depending on the application.

On exiting from these low power modes, calibration length error (CAL_LEN_ERR) can get set in channel status registers as the SENT sensor (transmitter) could be in the middle of a message transmission at that point. Thus, software should clear all error status bits after exiting the low power modes.

Modes where both system bus clock and protocol clock are gated: In this case, both message reception and interrupt / DMA request generation will halt. The internal receiver channels will freeze (since clock is not there) and will not receive any messages. Messages which are completely received before entering the low power mode can be read after the low power mode exit.

Modes where frequency of system bus clock is changed: For reduced power consumption the system bus clock can be reduced but not below a minimum value (See *Section 45.5.2: System bus clock requirements*) to avoid overflow of received messages. Protocol clock should not go below the minimum value (See *Section 45.5.3: High frequency receiver clock (protocol clock) requirements*) for accurate receive operations.

Note: Gating of clocks in the low power mode is chip-specific, see the chapter that refers how the modules behave in low power modes.

Note: If the SOC requests entry to debug or stop mode, the message being received currently is discarded and the SOC enters debug/stop mode immediately. This does not affect the messages received completely prior to entering debug mode and will still be present on the message buffer and registers until they are read out.

Note: The message read registers [Channel ‘n' Fast Message Data Read Register (n = 0 to (CH-1)) (CHn_FMSG_DATA), Channel ‘n' Fast Message CRC Read Register (n = 0 to (CH-1)) (CHn_FMSG_CRC), Channel ‘n’ Fast Message Time Stamp Read Register (n = 0 to (CH-1)) (CHn_FMSG_TS), Channel ‘n’ Serial Message Read Register (Bit 3) (n = 0 to (CH-1)) (CHn_SMSG_BIT3), Channel ‘n’ Serial Message Read Register (Bit 2) (n = 0 to (CH-1)) (CHn_SMSG_BIT2), Channel ‘n’ Serial Message Time Stamp Read Register (n = 0 to (CH-1)) (CHn_SMSG_TS), DMA Fast Message Data Read Register (DMA_FMSG_DATA), DMA Fast Message CRC Read Register (DMA_FMSG_CRC), DMA Fast Message Time Stamp Read Register (DMA_FMSG_TS), DMA Slow Serial Message Bit3 Read Register (DMA_SMSG_BIT3), DMA Slow Serial Message Bit2 Read Register (DMA_SMSG_BIT2) and DMA Slow Serial Message Time Stamp Read Register (DMA_SMSG_TS)] will appear to lose their contents in the following conditions:

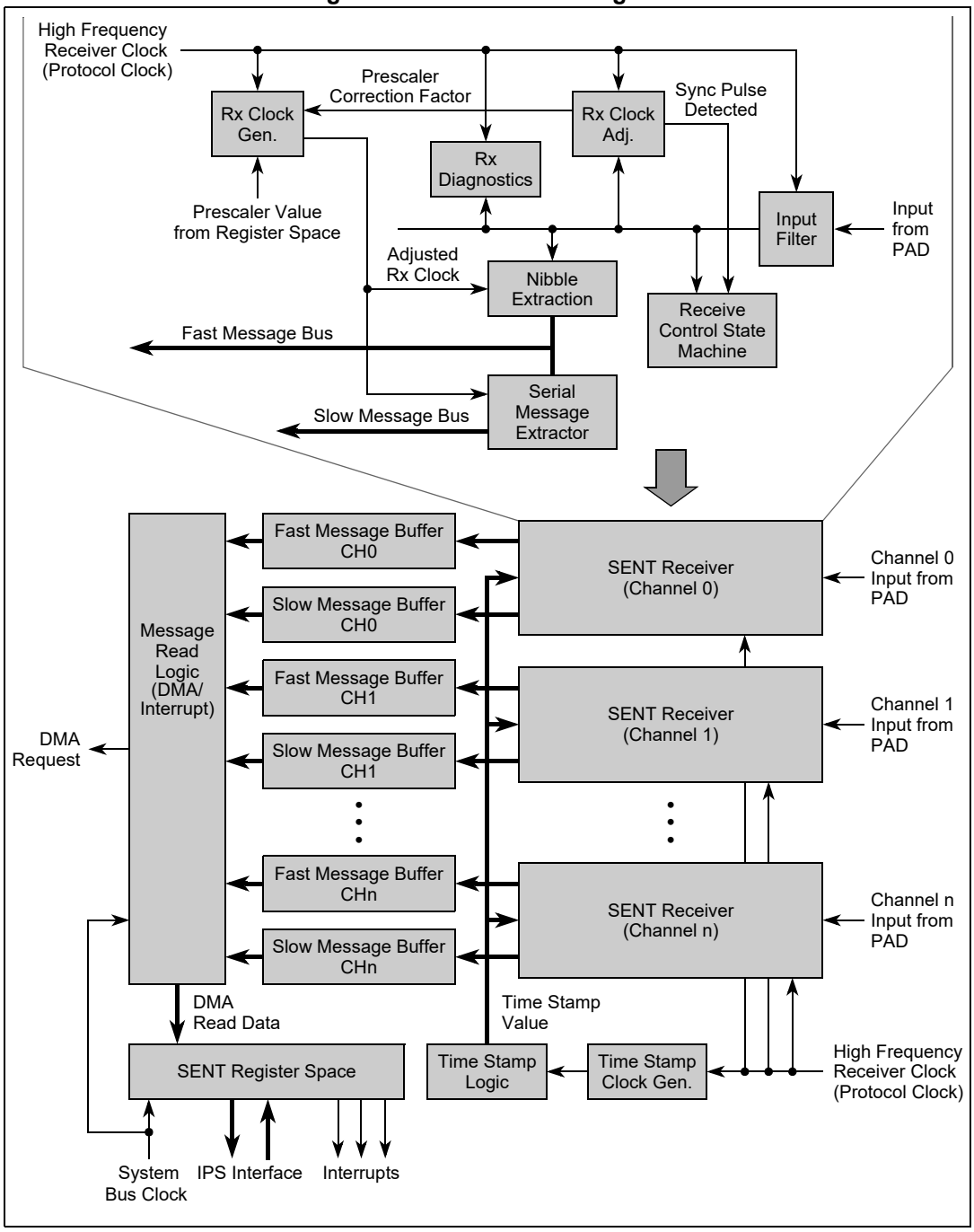
- *The very first message is being received but not yet completely received and the MCU enters debug or freeze mode, the current message reception will get discarded and message read registers (as mentioned above) will read zeros*

- *Auto clear functionality is enabled (GBL_CTRL[FAST_CLR] = 1). In this case, when first message is read, it will get clear the message read registers (due to auto clear functionality being enabled). On reading again, the message read registers (as mentioned above) might read zeros.*

If the MCU requests entry to debug or stop mode, the message being received currently by SENT Receiver is discarded and the MCU enters debug/stop mode immediately. This does not affect the messages received completely, prior to entering debug mode and these messages will still be present on the message buffer and registers until they are read out. Thus allow one message to be received completely and do not enable "Auto Clear" (GBL_CTRL[FAST_CLR] = 0), to allow messages to be read in debug or stop mode.

45.1.3 Block diagram

Figure 771. SENT block diagram



45.1.4 Design overview

The SENT Receiver Module is a unified receiver module comprising of a number of receiver channels and a unified DMA and Register Space for accessing the messages from each channel. *Figure 771* shows the top level block diagram for the SENT Receiver module.

Each channel of the SENT Receiver is composed of a Clock Generation and Adjustment block to generate the receiver clock and adjust it for frequency variation in the transmitter clock. This block also detects the synchronization/calibration pulse which is present in every SENT message.

The SENT Receiver State Machine controls the sequencing of decoding the messages and combining the information as nibbles. In parallel a diagnostic block is running that monitors the incoming messages for any errors and also verifies the CRC of the received messages. The fast and serial message extractors collate the message nibbles and add the channel number and time stamp information to the messages.A free running time stamp counter is used to add the time stamp value to each message received. The value of this counter is input to each channel and the channel logic appends the time stamp to each message. These messages are then formatted to be stored in the corresponding message read buffer. Only the messages that pass the diagnostics checks are forwarded to be stored in the buffers.

The SENT Receiver provides two mechanisms for reading the received messages, which are via DMA transfers and via Interrupts. Based on the control bit settings in the control registers, a DMA request or Interrupt is asserted to signal the DMA controller or CPU that messages are available to be read.

The free running time stamp counter, register block and DMA interface are common for all channels configured. The register space contains all the necessary registers for all channels. The register space is 32-bit aligned and accessible via 8-/16-/32-bit accesses. The registers for all un-configured channels are marked as reserved.

45.1.4.1 Functional overview

After reset, the SENT module comes up in the disabled state. All channel enable bits and the global enable bit are set to 0 and the user software must program the SENT module parameters correctly before enabling it. The user software must correctly configure all parameters to avoid any underflow or overflow in the message buffers.

Once enabled, the Receiver Clock Generation and Adjustment block detects the synchronization pulse. Until then, the state machine blocks all subsequent operations of the receiver. Once the synchronization pulse is detected, the receiver clock tick period is compensated for variation in transmitter clock tick period and the state machine then waits for the reception of data nibbles. The adjusted receiver clock is used to sample the data nibbles from the incoming serial message. The state machine waits until the status and communication nibble and configured number of data nibbles are received. The state machine then waits for the pause pulse (if configured to receive a pause pulse). The fast and serial message extractor block provides the state machine with the information on how many nibbles have been received. All diagnostics checks are computed in parallel. Once all required data nibbles are received, the CRC is checked, time stamp and channel number is added to the message. The time stamp for a Fast Message is sampled at the falling edge that comes at the end of the calibration pulse (which is also the start of the status and communication nibble pulse) and is stored in the message read buffer along with the message. This final message is stored in the message buffer as described in the subsequent sections. Any error detected during the receive process causes the status bits in the corresponding channel’s status register to be set.

The above process repeats across each enabled channel in parallel and keeps pushing error free messages into the buffers on high frequency receiver clock. The module’s message read logic synchronizes all these messages on the bus clock and pops out these messages on any DMA or CPU read access. Any DMA underflow errors are stored in the

global status register while the overflow errors are stored in the corresponding channel's status register. In case of an overflow in Fast Messages, the newer messages received will overwrite the previous message.

For detailed description on the complete functionality of each block, see *Section 45.4: Functional description*.

45.2 External signal description

This section lists all inputs to the SENT block.

45.2.1 SENT_RX [(CH-1):0]

These pins serve as the serial data input for each of the SENT channel. There is one bit per channel supported by the device. The SENT sensor input pads on the device using this module should have pull-up enabled to ensure a 1'b1 is seen at the module input when SENT sensor is not driving any data.

Note: The value of the CH is device-specific. See Section 7.9.4: SENT Receiver (SRX) configuration for details.

45.3 Memory map and register definition

This section provides a detailed description of all registers accessible in the SENT block.

45.3.1 Memory map

Table 767 gives an overview on all SENT registers. The memory map comprises of 32-bit aligned registers which can be accessed via 8-bit, 16-bit, or 32-bit accesses. Write access to reserved locations will generate transfer error. Read access to reserved locations will also generate transfer error and the read data bus will show all 0s. The Memory Map and complete module is in Big Endian Format.

The module will not check for correctness of programmed values in Register Space and software must ensure that correct values are being written.

Note: Register Address = Base Address + Address Offset, where the Base Address is defined at the MCU level and the Address Offset is defined at the module level. The number of registers are device specific. See the Device Configuration chapter for details.

The value of the CH is device-specific. See the Chapter 7: Device configuration for details.

Table 767. SENT memory map

Offset	Register Name	Access	Reset	Location
0x0000	Global Control Register (GBL_CTRL)	RW	0x0000_0010	*Section 45.3.2.1*
0x0004	Channel Enable Register (CHNL_EN)	RW	0x0000_0000	*Section 45.3.2.2*
0x0008	Global Status Register (GBL_STATUS)	W1C	0x0000_0000	*Section 45.3.2.3*
0x000C	Fast Message Ready Register (FMSG_RDY)	RW(1)	0x0000_0000	*Section 45.3.2.4*
0x0010	Slow Serial Message Ready Register (SMSG_RDY)	RW(1)	0x0000_0000	*Section 45.3.2.5*

Table 767. SENT memory map (continued)

Offset	Register Name	Access	Reset	Location
0x0018	Data Control Register 1 (DATA_CTRL1)	RW	0x1111_1111	*Section 45.3.2.6*
0x001C	Data Control Register 2 (DATA_CTRL2)	RW	0x1111_1111	*Section 45.3.2.7*
0x0028	Fast Message DMA Control Register (FDMA_CTRL)	RW	0x0000_0000	*Section 45.3.2.8*
0x002C	Slow Serial Message DMA Control Register (SDMA_CTRL)	RW	0x0000_0000	*Section 45.3.2.9*
0x0034	Fast Message Ready Interrupt Control Register (FRDY_IE)	RW	0x0000_0000	*Section 45.3.2.10*
0x0038	Slow Serial Message Ready Interrupt Control Register (SRDY_IE)	RW	0x0000_0000	*Section 45.3.2.11*
0x0040	DMA Fast Message Data Read Register (DMA_FMSG_DATA)	R	0x0000_0000	*Section 45.3.2.12*
0x0044	DMA Fast Message CRC Read Register (DMA_FMSG_CRC)	R	0x0000_0000	*Section 45.3.2.13*
0x0048	DMA Fast Message Time Stamp Read Register (DMA_FMSG_TS)	R	0x0000_0000	*Section 45.3.2.14*
0x0050	DMA Serial Message Read Register (Bit 3) (DMA_SMSG_BIT3)	R	0x0000_0000	*Section 45.3.2.15*
0x0054	DMA Serial Message Read Register (Bit 2) (DMA_SMSG_BIT2)	R	0x0000_0000	*Section 45.3.2.16*
0x0058	DMA Serial Message Time Stamp Read Register (DMA_SMSG_TS)	R	0x0000_0000	*Section 45.3.2.17*
CH registers (n = 0 to (CH-1))				
0x0060 + *n**0x10	Channel ‘n’ Clock Control Register (CHn_CLK_CTRL)	RW	0x0000_8000	*Section 45.3.2.18*
0x0064 + *n**0x10	Channel ‘n’ Status Register (CHn_STATUS)	W1C	0x0000_0000	*Section 45.3.2.19*
0x0068 + *n**0x10	Channel ‘n’ Configuration Register (CHn_CONFIG)	RW	0x0000_0104	*Section 45.3.2.20*
CH_MSG registers (n = 0 to (CH-1))				
0x0160 + *n**0x18	Channel ‘n’ Fast Message Data Read Register (CHn_FMSG_DATA)	R	0x0000_0000	*Section 45.3.2.21*
0x0164 + *n**0x18	Channel ‘n’ Fast Message CRC Read Register (CHn_FMSG_CRC)	R	0x0000_0000	*Section 45.3.2.22*
0x0168 + *n**0x18	Channel ‘n’ Fast Message Time Stamp Read Register (CHn_FMSG_TS)	R	0x0000_0000	*Section 45.3.2.23*
0x016C + *n**0x18	Channel ‘n’ Slow Serial Message Bit3 Read Register (CHn_SMSG_BIT3)	R	0x0000_0000	*Section 45.3.2.24*
0x0170 + *n**0x18	Channel ‘n’ Slow Serial Message Bit2 Read Register (CHn_SMSG_BIT2)	R	0x0000_0000	*Section 45.3.2.25*
0x0174 + *n**0x18	Channel ‘n’ Slow Serial Message Time Stamp Read Register (CHn_SMSG_TS)	R	0x0000_0000	*Section 45.3.2.26*

1. This bit is either “write 1 to clear” (w1c) or self-clearing depending on the value of the FAST_CLR bit in the *Global Control Register (GBL_CTRL)*

45.3.2 Register descriptions

This section describes in address order all the SENT registers and their individual bits.

45.3.2.1 Global Control Register (GBL_CTRL)

This register provides the global control and setting for the SENT Receiver Module. The SENT_EN bit must be programmed after all other modules settings and control bits have been written to.

Offset: 0x0000 Access: Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	TSPRSC								0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	NIB_LEN_VAR_LIMIT	0	Reserved	0	0	Reserved	FMDUIE	SMDUIE	0	0	0	FAST_CLR	0	DBG_FRZ	0	SENT_EN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0

Figure 772. Global Control Register (GBL_CTRL)

Table 768. GBL_CTRL field descriptions

Field	Description
0:7 TSPRSC	Time Stamp Prescaler Value Value in number of clock cycles of high frequency receiver clock set in order to obtain a clock frequency for the time stamp counter block. (0 to 255) Time Stamp Clock = (High Frequency Clock)/(Value + 1). Value set by user software should be such that the clock period is at least 1 µs. If the high frequency clock input is 75 MHz, then the value set in this field would be 0x4A (or 74) to obtain a 1µs clock. **Note:** Since the time stamp clock is common for all channels, the timestamp clock frequency must be at least twice the fastest receiver channel in the application. This is to ensure that the time stamp clock is fast enough to show change in timestamp from message to message.
16 NIB_LEN_VAR _LIMIT	Decides the allowable jitter/clock drift limit for nibble length measurement. 0 Allows a jitter/clock drift of approx upto 12.5% of transmitter clock tick period. This actual allowed jitter/drift value is dependent on the transmitter clock tick period and HF receiver clock period. 1 Allows a jitter/clock drift as per SAE spec (50%). **Note:** This bit should be set to 1 for proper functioning of SRX.
18	Reserved. Read returns zero.
21	Reserved. Read returns zero.
22 FMDUIE	Fast Message DMA Underflow Interrupt Enable Enables interrupt assertion when there is an underflow condition on Fast Message DMA registers. Default value is 0. 0 Interrupt is disabled. 1 Interrupt is enabled.
23 SMDUIE	Slow Serial Message DMA Underflow Interrupt Enable Enables interrupt assertion when there is an underflow condition on Slow Message DMA registers. Default value is 0. 0 Interrupt is disabled. 1 Interrupt is enabled.
27 FAST_CLR	Fast Clearing Enable bit Enables clearing of ready status (for both fast and slow) bit automatically when corresponding message is read. Default value is 1. See *Fast Message Ready Status Register (FMSG_RDY)* and *Slow Serial Message Ready Status Register (SMSG_RDY)* for details on the ready status bits. 0 Fast Clearing is disabled. Bits will be cleared by writing 1. 1 Fast Clearing is enabled.

Table 768. GBL_CTRL field descriptions (continued)

Field	Description
29 DBG_FRZ	Debug Freeze This bit will enable the debug mode support. SENT Module will freeze in debug mode if this bit is set. Default is 0. 0 No effect in debug mode 1 Freeze in debug mode
31 SENT_EN	SENT Receiver Global Enable This bit enables or disables the complete receiver module irrespective of the setting of individual channel enable bits. User software must program all channel parameters before setting this bit. Default is 0. 0 Entire module is disabled 1 Module is enabled

45.3.2.2 Channel Enable Register (CHNL_EN)

This register is used to enable individual SENT Receiver channels in the module. The user software must ensure all channel parameters and control settings have been programmed before writing to this bit. In case these parameters are changed while the channel is enabled, normal operation is not guaranteed.

Note: The following register figure and table shows the maximum possible configuration but the exact number of valid register bits are dependent on supported SENT channels on the device. Please see Device Configuration chapter for number of support SENT channels. Reads of bits beyond the supported number of channels should be ignored and these bits must be written to 0.

Offset: 0x0004 Access: Read/Write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	EN_CH15	EN_CH14	EN_CH13	EN_CH12	EN_CH11	EN_CH10	EN_CH9	EN_CH8	EN_CH7	EN_CH6	EN_CH5	EN_CH4	EN_CH3	EN_CH2	EN_CH1	EN_CH0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 773. Channel Enable Register (CHNL_EN)

Table 769. CHNL_EN field descriptions

Field	Description
16:31 EN_CH[15:0]	Enable bits for Channels 0 to 15 All channels are disabled after reset. User software must program all channel parameters before setting this bit. Once set, other channel parameters in registers should not be changed until this bit is cleared. Only Interrupt Enable and Status bits can be written to. 0 Channel is disabled. 1 Channel is enabled. Channel parameter registers are locked for writing.

45.3.2.3 Global Status Register (GBL_STATUS)

This register provides the status of various conditions in the module that are common across all channels.

Offset: 0x0008 Access: Read/Write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	FMDU	SMDU	0	0	0	0	0	0	0	0
W							w1c	w1c								
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 774. Global Status Register (GBL_STATUS)

Table 770. GBL_STATUS field descriptions

Field	Description
22 FMDU	Fast Message DMA Underflow Interrupt Status bit that indicates when there is an underflow in Fast Message buffers in any channel which has DMA enabled. This bit is set when there is no message in the DMA buffers for read and the software still reads the Fast Message DMA register. Default value is 0. 0 No underflow 1 Underflow occurred
23 SMDU	Slow Serial Message DMA Underflow Interrupt Status bit that indicates when there is an Underflow in Slow Serial Message DMA buffers on channel which has DMA enabled. The bit is set when there is no message in the DMA buffers for read and the software still reads the Slow Serial Message registers. Default value is 0. 0 No underflow 1 Underflow occurred

45.3.2.4 Fast Message Ready Status Register (FMSG_RDY)

The bits of this register are asserted when a Fast Message arrives on a corresponding channel that is not configured to be read via DMA. If a channel is not configured for DMA (via *Fast Message DMA Control Register (FDMA_CTRL)*), it is assumed to be read via Interrupt or Polling this status bit. Even if the Interrupt Enable bit for that channel is not set (via *Fast Message Ready Interrupt Control Register (FRDY_IE)*) and the channel is enabled, the message ready status bit in this register will get asserted on Fast Message reception. Software must ensure that if a channel is enabled; either DMA or Interrupt must be enabled for that channel. Proper flow cannot be guaranteed if neither is selected.

These bits can be configured to be cleared via “Write 1 to Clear” or “Fast Clearing Mechanism”. The configuration is done by writing to the FAST_CLR bit in the *Global Control Register (GBL_CTRL)*.

Note: Reads of bits beyond those for the supported number of channels should be ignored. Writes to these bits are unadvisable and may cause unexpected behavior.

Note: Under certain conditions, the Single Edge Nibble Transmission (SENT) Receiver (SRX) stalls and the Fast Message Data Ready bit for the SENT channel (FMSG_RDY[F_RDYn]) will no longer get set to indicate that a fast message is available. Reads of any of the fast message registers by the MCU core will stall and not complete. The registers affected are: DMA_FMSG_DATA (Direct Memory Access (DMA) Fast Message Data Read Register), DMA_FMSG_CRC (DMA Fast Message Cyclic Redundancy Check Register), DMA_FMSG_TS (DMA Fast Message Time-stamp Register), CHn_FMSG_DATA (Channel Fast Message Data Read Register), CHn_FMSG_CRC (Channel Cyclic Redundancy Check Register), CHn_FMSG_TS (Channel Fast Message Time-stamp Register).

This condition occurs when the SENT Receiver is configured as follows:

- *SENT high frequency clock is set higher than the Peripheral Bridge clock;*
- *Polling of ready status bit (FMSG_RDY[F_RDYn]) is used to detect a new message reception instead of interrupt or DMA request;*
- *The fast clear mechanism is used (GBL_CTRL[FAST_CLR] = 1) or the Software Clear (GBL_CTRL[FAST_CLR] = 0) is used for clearing the ready status bits (FMSG_RDY[F_RDYn]); and*
- *an overflow status condition is detected in the SENT Receiver Channel Status register (CHn_STATUS[FMSG_OFLW] = 1).*

The overflow occurs when two messages are allowed to queue in the internal buffers of the SENT Receiver.

Program the clock dividers in MCU for peripheral bridge clock and SENT clock such that SENT clock is always less than or equal to the peripheral bridge clock. In this case, the stall does not occur and MCU transactions to SENT registers will complete.

Offset: 0x000C Access: Read/Write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	F_RDY15	F_RDY14	F_RDY13	F_RDY12	F_RDY11	F_RDY10	F_RDY9	F_RDY8	F_RDY7	F_RDY6	F_RDY5	F_RDY4	F_RDY3	F_RDY2	F_RDY1	F_RDY0
W	*(1)	*(1)	*(1)	*(1)	*(1)	*(1)	*(1)	*(1)	*(1)	*(1)	*(1)	*(1)	*(1)	*(1)	*(1)	*(1)
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. This bit is either “write 1 to clear” (w1c) or self-clearing depending on the value of the FAST_CLR bit in the *Global Control Register (GBL_CTRL)*.

Figure 775. Fast Message Ready Status Register (FMSG_RDY)

Table 771. FMSG_RDY field descriptions

Field	Description
16:31 F_RDY[15:0]	Fast Message Data Ready Status Bit Indicates that a Fast Message is ready to be read by CPU from that specific channel. This bit is asserted only when **DMA is NOT** enabled for that channel (from the DMA Control Register) and an error free message is received, irrespective of the value of corresponding enable bit in Message Ready Interrupt Control Register. Reading the message from the corresponding memory-mapped Fast Message Read Registers would automatically clear this bit (Fast Clearing Mechanism). **Note:** These bits are either “write 1 to clear” (w1c) or self-clearing depending on the value of the FAST_CLR BIT in the Global Control Register (GBL_CTRL). 0 No Fast Message is available to be read out via Interrupt 1 Fast Message is available to be read out via Interrupt The clearing mechanism is selected by user by programming the FAST_CLR bit in the Global Control register. See Section *Global Control Register (GBL_CTRL)* for details on FAST_CLR bit. **Note:** When using the fast clearing option, these bits will not clear till the complete message is read out which requires the CPU to read all 3 fast message registers via 32-bit read accesses. For 8/16-bit accesses fast clearing mechanism should not be used as it is not supported. **Note:** When the fast clearing option is not configured, the bits will be cleared only if 1 is written to those bits.

45.3.2.5 Slow Serial Message Ready Status Register (SMSG_RDY)

The bits of this register are asserted when a Slow Serial Message arrives on a corresponding channel that is not configured to be read via DMA. If a channel is not configured for DMA (via *Fast Message DMA Control Register (FDMA_CTRL)* and *Slow Serial Message DMA Control Register (SDMA_CTRL)*), it is assumed to be read via Interrupt or Polling this status bit. Even if the Interrupt Enable bit for that channel is not set (via *Fast Message Ready Interrupt Control Register (FRDY_IE)* and *Slow Serial Message Ready Interrupt Enable Register (SRDY_IE)*) and the channel is enabled, the message

ready status bit in this register will get asserted on Slow Message reception. Software must ensure that if a channel is enabled; either DMA or Interrupt must be enabled for that channel. Erroneous flow can happen if neither is selected.

These bits can be configured to be cleared via “Write 1 to Clear” or “Fast Clearing Mechanism”. The configuration is done by writing to the FAST_CLR bit in the *Global Status Register (GBL_STATUS)*.

Note: Reads of bits beyond those for the supported number of channels should be ignored. Writes to these bits are unadvisable and may cause unexpected behavior.

Offset: 0x0010 Access: Read/Write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	S_RDY15	S_RDY14	S_RDY13	S_RDY12	S_RDY11	S_RDY10	S_RDY9	S_RDY8	S_RDY7	S_RDY6	S_RDY5	S_RDY4	S_RDY3	S_RDY2	S_RDY1	S_RDY0
W	*(1)	*(1)	*(1)	*(1)	*(1)	*(1)	*(1)	*(1)	*(1)	*(1)	*(1)	*(1)	*(1)	*(1)	*(1)	*(1)
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. This bit is either “write 1 to clear” (w1c) or self-clearing depending on the value of the FAST_CLR bit in the *Global Control Register (GBL_CTRL)*.

Figure 776. Slow Serial Message Ready Status Register (SMSG_RDY)

Table 772. SMSG_RDY field descriptions

Field	Description
16:31 S_RDY[15:0]	Slow Serial Message Data Ready Status Bit Indicates that a Slow Serial Message is ready to be read by CPU from that specific channel. This bit is asserted only when **DMA is NOT** enabled for that channel (from the DMA Control Register) and an error free message is received, irrespective of the value of corresponding enable bit in Message Ready Interrupt Control Register. Reading the message from the corresponding memory-mapped Serial Message Read Registers would automatically clear this bit (Fast Clearing Mechanism). 0 No Slow Serial Message is available to be read out via Interrupt 1 Slow Serial Message is available to be read out via Interrupt The clearing mechanism is selected by user by programming the FAST_CLR bit in the Global Control register. See Section *Global Control Register (GBL_CTRL)* for details on FAST_CLR bit. **Note:** When using the fast clearing option, these bits will not clear till the complete message is read out which requires the CPU to read all 3 Fast Message registers via 32-bit read accesses. For 8/16-bit accesses fast clearing mechanism should not be used as it is not supported. When the fast clearing option is not configured, the bits will be cleared only if 1 is written to those bits.

45.3.2.6 Data Control Register 1 (DATA_CTRL1)

The Data Control Register 1 (DATA_CTRL1) is described below.

Note: Reset value changes depending on the number of channels that are enabled on the device.

Offset: 0x0018 Access: Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	NIBBCH0			0	NIBBCH1			0	NIBBCH2			0	NIBBCH3		
W																
Reset	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	NIBBCH4			0	NIBBCH5			0	NIBBCH6			0	NIBBCH7		
W																
Reset	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1

Figure 777. Data Control Register 1 (DATA_CTRL1)

Table 773. DATA_CTRL1 field descriptions

Field	Description
1:3 NIBBCH0	Number of Data Nibbles supported in Channel 0 000 Invalid value. Must not be used 001 Number of data nibbles to be supported by respective channel 010 Number of data nibbles to be supported by respective channel 011 Number of data nibbles to be supported by respective channel 100 Number of data nibbles to be supported by respective channel 101 Number of data nibbles to be supported by respective channel 110 Number of data nibbles to be supported by respective channel 111 Invalid value. Must not be used
5:7 NIBBCH1	Number of Data Nibbles supported in Channel 1 000 Invalid value. Must not be used 001 Number of data nibbles to be supported by respective channel 010 Number of data nibbles to be supported by respective channel 011 Number of data nibbles to be supported by respective channel 100 Number of data nibbles to be supported by respective channel 101 Number of data nibbles to be supported by respective channel 110 Number of data nibbles to be supported by respective channel 111 Invalid value. Must not be used
9:11 NIBBCH2	Number of Data Nibbles supported in Channel 2 000 Invalid value. Must not be used 001 Number of data nibbles to be supported by respective channel 010 Number of data nibbles to be supported by respective channel 011 Number of data nibbles to be supported by respective channel 100 Number of data nibbles to be supported by respective channel 101 Number of data nibbles to be supported by respective channel 110 Number of data nibbles to be supported by respective channel 111 Invalid value. Must not be used

Table 773. DATA_CTRL1 field descriptions (continued)

Field	Description
13:15 NIBBCH3	Number of Data Nibbles supported in Channel 3 000 Invalid value. Must not be used 001 Number of data nibbles to be supported by respective channel 010 Number of data nibbles to be supported by respective channel 011 Number of data nibbles to be supported by respective channel 100 Number of data nibbles to be supported by respective channel 101 Number of data nibbles to be supported by respective channel 110 Number of data nibbles to be supported by respective channel 111 Invalid value. Must not be used
17:19 NIBBCH4	Number of Data Nibbles supported in Channel 4 000 Invalid value. Must not be used 001 Number of data nibbles to be supported by respective channel 010 Number of data nibbles to be supported by respective channel 011 Number of data nibbles to be supported by respective channel 100 Number of data nibbles to be supported by respective channel 101 Number of data nibbles to be supported by respective channel 110 Number of data nibbles to be supported by respective channel 111 Invalid value. Must not be used
21:23 NIBBCH5	Number of Data Nibbles supported in Channel 5 000 Invalid value. Must not be used 001 Number of data nibbles to be supported by respective channel 010 Number of data nibbles to be supported by respective channel 011 Number of data nibbles to be supported by respective channel 100 Number of data nibbles to be supported by respective channel 101 Number of data nibbles to be supported by respective channel 110 Number of data nibbles to be supported by respective channel 111 Invalid value. Must not be used
25:27 NIBBCH6	Number of Data Nibbles supported in Channel 6 000 Invalid value. Must not be used 001 Number of data nibbles to be supported by respective channel 010 Number of data nibbles to be supported by respective channel 011 Number of data nibbles to be supported by respective channel 100 Number of data nibbles to be supported by respective channel 101 Number of data nibbles to be supported by respective channel 110 Number of data nibbles to be supported by respective channel 111 Invalid value. Must not be used
29:31 NIBBCH7	Number of Data Nibbles supported in Channel 7 000 Invalid value. Must not be used 001 Number of data nibbles to be supported by respective channel 010 Number of data nibbles to be supported by respective channel 011 Number of data nibbles to be supported by respective channel 100 Number of data nibbles to be supported by respective channel 101 Number of data nibbles to be supported by respective channel 110 Number of data nibbles to be supported by respective channel 111 Invalid value. Must not be used

45.3.2.7 Data Control Register 2 (DATA_CTRL2)

Note: Reset value changes depending on the number of channels that are enabled on the device.

Offset: 0x001C Access: Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	NIBBCH8			0	NIBBCH9			0	NIBBCH10			0	NIBBCH11		
W																
Reset	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	NIBBCH12			0	NIBBCH13			0	NIBBCH14			0	NIBBCH15		
W																
Reset	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1

Figure 778. Data Control Register 2 (DATA_CTRL2)

Table 774. DATA_CTRL2 field descriptions

Field	Description
1:3 NIBBCH8	Number of Data Nibbles supported in Channel 8 000 Invalid value. Must not be used 001 Number of data nibbles to be supported by respective channel 010 Number of data nibbles to be supported by respective channel 011 Number of data nibbles to be supported by respective channel 100 Number of data nibbles to be supported by respective channel 101 Number of data nibbles to be supported by respective channel 110 Number of data nibbles to be supported by respective channel 111 Invalid value. Must not be used
5:7 NIBBCH9	Number of Data Nibbles supported in Channel 9 000 Invalid value. Must not be used 001 Number of data nibbles to be supported by respective channel 010 Number of data nibbles to be supported by respective channel 011 Number of data nibbles to be supported by respective channel 100 Number of data nibbles to be supported by respective channel 101 Number of data nibbles to be supported by respective channel 110 Number of data nibbles to be supported by respective channel 111 Invalid value. Must not be used
9:11 NIBBCH10	Number of Data Nibbles supported in Channel 10 000 Invalid value. Must not be used 001 Number of data nibbles to be supported by respective channel 010 Number of data nibbles to be supported by respective channel 011 Number of data nibbles to be supported by respective channel 100 Number of data nibbles to be supported by respective channel 101 Number of data nibbles to be supported by respective channel 110 Number of data nibbles to be supported by respective channel 111 Invalid value. Must not be used

Table 774. DATA_CTRL2 field descriptions (continued)

Field	Description
13:15 NIBBCH11	Number of Data Nibbles supported in Channel 11 000 Invalid value. Must not be used 001 Number of data nibbles to be supported by respective channel 010 Number of data nibbles to be supported by respective channel 011 Number of data nibbles to be supported by respective channel 100 Number of data nibbles to be supported by respective channel 101 Number of data nibbles to be supported by respective channel 110 Number of data nibbles to be supported by respective channel 111 Invalid value. Must not be used
17:19 NIBBCH12	Number of Data Nibbles supported in Channel 12 000 Invalid value. Must not be used 001 Number of data nibbles to be supported by respective channel 010 Number of data nibbles to be supported by respective channel 011 Number of data nibbles to be supported by respective channel 100 Number of data nibbles to be supported by respective channel 101 Number of data nibbles to be supported by respective channel 110 Number of data nibbles to be supported by respective channel 111 Invalid value. Must not be used
21:23 NIBBCH13	Number of Data Nibbles supported in Channel 13 000 Invalid value. Must not be used 001 Number of data nibbles to be supported by respective channel 010 Number of data nibbles to be supported by respective channel 011 Number of data nibbles to be supported by respective channel 100 Number of data nibbles to be supported by respective channel 101 Number of data nibbles to be supported by respective channel 110 Number of data nibbles to be supported by respective channel 111 Invalid value. Must not be used
25:27 NIBBCH14	Number of Data Nibbles supported in Channel 14 000 Invalid value. Must not be used 001 Number of data nibbles to be supported by respective channel 010 Number of data nibbles to be supported by respective channel 011 Number of data nibbles to be supported by respective channel 100 Number of data nibbles to be supported by respective channel 101 Number of data nibbles to be supported by respective channel 110 Number of data nibbles to be supported by respective channel 111 Invalid value. Must not be used
29:31 NIBBCH15	Number of Data Nibbles supported in Channel 15 000 Invalid value. Must not be used 001 Number of data nibbles to be supported by respective channel 010 Number of data nibbles to be supported by respective channel 011 Number of data nibbles to be supported by respective channel 100 Number of data nibbles to be supported by respective channel 101 Number of data nibbles to be supported by respective channel 110 Number of data nibbles to be supported by respective channel 111 Invalid value. Must not be used

45.3.2.8 Fast Message DMA Control Register (FDMA_CTRL)

Note: The following register figure and table shows the maximum possible configuration but the exact number of valid register bits are dependent on supported SENT channels on the

device. Please see Chapter 7: Device configuration for number of support SENT channels. Reads of bits beyond the supported number of channels should be ignored and writes to these bits are unadvisable and may cause unexpected behavior.

Offset: 0x0028 Access: Read/Write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	FDMA_EN15	FDMA_EN14	FDMA_EN13	FDMA_EN12	FDMA_EN11	FDMA_EN10	FDMA_EN9	FDMA_EN8	FDMA_EN7	FDMA_EN6	FDMA_EN5	FDMA_EN4	FDMA_EN3	FDMA_EN2	FDMA_EN1	FDMA_EN0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 779. Fast Message DMA Control Register (FDMA_CTRL)

Table 775. FDMA_CTRL field descriptions

Field	Description
16:31 FDMA_EN[15:0]	Enable DMA for Fast Messages on Channels 0 to 15 These bits are writable when corresponding Fast Message Ready Interrupt Enable bits are set to 0. 0 DMA for Fast Messages is disabled. 1 DMA for Fast Messages is enabled.

45.3.2.9 Slow Serial Message DMA Control Register (SDMA_CTRL)

Note: The following register figure and table shows the maximum possible configuration but the exact number of valid register bits are dependent on supported SENT channels on the device. Please see Device Configuration chapter for number of support SENT channels. Reads of bits beyond the supported number of channels should be ignored and writes to these bits are unadvisable and may cause unexpected behavior.

Offset: 0x002C Access: Read/Write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	SDMA_EN15	SDMA_EN14	SDMA_EN13	SDMA_EN12	SDMA_EN11	SDMA_EN10	SDMA_EN9	SDMA_EN8	SDMA_EN7	SDMA_EN6	SDMA_EN5	SDMA_EN4	SDMA_EN3	SDMA_EN2	SDMA_EN1	SDMA_EN0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 780. Slow Serial Message DMA Control Register (SDMA_CTRL)

Table 776. SDMA_CTRL field descriptions

Field	Description
16:31 SDMA_EN[15:0]	Enable DMA for Slow Serial Messages on Channels 0 to 15 These bits are writable when corresponding Slow Serial Message Ready Interrupt Enable bits are set to 0. 0 DMA for Slow Serial Messages is disabled. 1 DMA for Slow Serial Messages is enabled.

45.3.2.10 Fast Message Ready Interrupt Control Register (FRDY_IE)

Note: The following register figure and table shows the maximum possible configuration but the exact number of valid register bits are dependent on supported SENT channels on the device. Please see Device Configuration chapter for number of support SENT channels.Reads of bits beyond the supported number of channels should be ignored and writes to these bits are unadvisable and may cause unexpected behavior.

Offset: 0x0034 Access: Read/Write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	FRDY_IE15	FRDY_IE14	FRDY_IE13	FRDY_IE12	FRDY_IE11	FRDY_IE10	FRDY_IE9	FRDY_IE8	FRDY_IE7	FRDY_IE6	FRDY_IE5	FRDY_IE4	FRDY_IE3	FRDY_IE2	FRDY_IE1	FRDY_IE0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 781. Fast Message Ready Interrupt Enable Register (FRDY_IE)

Table 777. FRDY_IE field descriptions

Field	Description
16:31 FRDY_IE[15:0]	Enable for Fast Message Ready Interrupt on Channels 0 to 15 These bits are writable when corresponding DMA enable bits are set to 0 in the Fast Message DMA Control Register. The availability of message is indicated via assertion of corresponding bit in Fast Message Ready Status Register irrespective of the value set in corresponding interrupt enable bit and DMA is not enabled. When a bit is asserted in this register, an interrupt for corresponding channel is generated. 0 Interrupt on reception of Fast Messages is disabled. 1 Interrupt on reception of Fast Messages is enabled.

45.3.2.11 Slow Serial Message Ready Interrupt Enable Register (SRDY_IE)

Note: The following register figure and table shows the maximum possible configuration but the exact number of valid register bits are dependent on supported SENT channels on the device. Please see Chapter 7: Device configuration for number of support SENT channels. Reads of bits beyond the supported number of channels should be ignored and writes to these bits are unadvisable and may cause unexpected behavior.

Offset: 0x0038 Access: Read/Write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	SRDY_IE15	SRDY_IE14	SRDY_IE13	SRDY_IE12	SRDY_IE11	SRDY_IE10	SRDY_IE9	SRDY_IE8	SRDY_IE7	SRDY_IE6	SRDY_IE5	SRDY_IE4	SRDY_IE3	SRDY_IE2	SRDY_IE1	SRDY_IE0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 782. Slow Serial Message Ready Interrupt Enable Register (SRDY_IE)

Table 778. SRDY_IE field descriptions

Field	Description
16:31 SRDY_IE[(CH-1):0]	Enable for Slow Serial Message Ready Interrupt on Channels 0 to 15 These bits are writable when corresponding DMA enable bits are set to 0 in the Slow Serial DMA Control Register. The availability of message is indicated via assertion of corresponding bit in Slow Serial Message Ready Status Register irrespective of the value set in corresponding interrupt enable bit and DMA is not enabled. When a bit in this register is asserted, an interrupt for corresponding channel is generated. 0 Interrupt on reception of Slow Serial Messages is disabled. 1 Interrupt on reception of Slow Serial Messages is enabled.

45.3.2.12 DMA Fast Message Data Read Register (DMA_FMSG_DATA)

DMA Source Size must be set to 32 bits when reading this register (DMA_FMSG_DATA) via DMA.

Offset: 0x0040 Access: User read only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	CHNUM				SCNIB				DNIB1				DNIB2			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	DNIB3				DNIB4				DNIB5				DNIB6			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 783. DMA Fast Message Data Read Register (DMA_FMSG_DATA)

Table 779. DMA_FMSG_DATA field descriptions

Field	Description
0:3 CHNUM	Channel Number Indicates the channel number of the received message being read by DMA. Valid values are 0 to ‘15’
4:7 SCNIB	Status and Communication Nibble of message
8:11 DNIB1	Data Nibble 1 Always supported
12:15 DNIB2	Data Nibble 2 Should be ignored if number of data nibbles supported by channel is 1
16:19 DNIB3	Data Nibble 3 Should be ignored if number of data nibbles supported by channel is 2 or less
20:23 DNIB4	Data Nibble 4 Should be ignored if number of data nibbles supported by channel is 3 or less
24:27 DNIB5	Data Nibble 5 Should be ignored if number of data nibbles supported by channel is 4 or less
28:31 DNIB6	Data Nibble 6 Should be ignored if number of data nibbles supported by channel is 5 or less

45.3.2.13 DMA Fast Message CRC Read Register (DMA_FMSG_CRC)

DMA Source Size must be set to 32 bits when reading this register (DMA_FMSG_CRC) via DMA.

Offset: 0x0044 Access: User read only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	CRC4b			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

Figure 784. DMA Fast Message CRC Read Register (DMA_FMSG_CRC)

Table 780. DMA_FMSG_CRC field descriptions

Field	Description
12:15 CRC4b	4-bit CRC value of the message

45.3.2.14 DMA Fast Message Time Stamp Read Register (DMA_FMSG_TS)

The resolution of the time stamp is set by programming the Time Stamp Prescaler Value in the *Section 45.3.2.1: Global Control Register (GBL_CTRL)* register. DMA Source Size must be set to 32 bits when reading this register (DMA_FMSG_TS) via DMA.

Offset: 0x0048 Access: User read only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	TS															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TS															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 785. DMA Fast Message Time Stamp Read Register (DMA_FMSG_TS)

Table 781. DMA_FMSG_TS field descriptions

Field	Description
0:31 TS	Time Stamp for received message Indicates the relative time when the falling edge that comes at the end of the calibration pulse (which is also the start of the status and communication nibble pulse) was detected

45.3.2.15 DMA Slow Serial Message Bit3 Read Register (DMA_SMSG_BIT3)

The DMA_SMSG_BIT3 is common for all types of slow serial messages. Bit C defines the type of enhanced packet and bit TYPE defines whether it is short serial or enhanced serial message. Since this register is used to read both Short and Enhanced Serial Messages, the register fields have been placed in such a way to match the bit positions in actual message defined by SAE Specification as shown in *Figure 801*, *Figure 802* and *Figure 803*. *Figure 801* and *Figure 802* shows the enhanced serial messages format to be used when reading this register for these two types of messages. *Figure 803* shows the short serial message.

DMA Source Size must be set to 32 bits when reading this register via DMA.

Offset: 0x0050 Access: User read only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	CHNUM				TYPE	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	CFG	ID7_6/ ID3_2		ID5_4/ ID1_0		0	ID3_0/ DATA15_12				0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 786. DMA Slow Serial Message Bit3 Read Register (DMA_SMSG_BIT3)

Table 782. DMA_SMSG_BIT3 Field Descriptions

Field	Description
0:3 CHNUM	Channel Number Indicates the channel number of the received message being read by DMA. Valid values are 0 to ‘15’
4 TYPE	Serial Message Type Indicates the type of Serial Message received 0 Short Serial Message 1 Enhanced Serial Message
20	Placed as 0 by Specification
21 CFG	Configuration bit ‘C’ Indicates the type of Enhanced Serial message. 0 Enhanced Serial Message with 8-bit ID field 1 Enhanced Serial Message with 4-bit ID field
22:23 ID7_6/ID3_2	ID Field Value depends on setting of ‘C’ bit. If ‘C’ bit is 0, this field is ID[7:6] and if the ‘C’ bit is 1, this field is ID[3:2].
24:25 ID5_4/ID1_0	ID Field Value depends on setting of ‘C’ bit. If ‘C’ bit is 0, this field is ID[5:4] and if the ‘C’ bit is 1, this field is ID[1:0].

Table 782. DMA_SMSG_BIT3 Field Descriptions (continued)

Field	Description
26	0 as per SAE Specification
27:30 ID3_0/DATA15_12	ID or Data Field Value depends on setting of 'C' bit. If 'C' bit is 0, this field is ID[3:0] and if the 'C' bit is 1, this field is Data[15:12].
31	0 as per SAE Specification

Note: If the TYPE bit in the register reads '0', then remainder bits (for CFG, ID & DATA) will be redundant and read zeros. The short serial message will be located in DMA_SMSG_BIT2 register fields. If TYPE bit is '1', then the message in register is Enhanced Serial Message and using CFG bit, software can decode the ID & DATA fields.

45.3.2.16 DMA Slow Serial Message Bit2 Read Register (DMA_SMSG_BIT2)

This register (DMA_SMSG_BIT2) is common for all types of slow serial messages. Hence, the register fields have been placed in such a way to match the bit positions in actual message defined by SAE Specification as shown in *Figure 801*, *Figure 802* and *Figure 803*. *Figure 801* and *Figure 802* shows the enhanced serial messages format to be used when reading this register for these two types of messages. *Figure 803* shows the short serial message.

DMA Source Size must be set to 32 bits when reading this register via DMA.

Offset: 0x0054 Access: User read only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	SMCRC					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	ID3_0/ DATA11_8				DATA7_0							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 787. DMA Slow Serial Message Bit2 Read Register (DMA_SMSG_BIT2)

Table 783. DMA_SMSG_BIT2 field descriptions

Field	Description
10:15 SMCRC	6-bit CRC value of the message When TYPE bit in *DMA Slow Serial Message Bit3 Read Register (DMA_SMSG_BIT3)* is set to 0, this CRC is of 4-bits for Short Serial Messages and when TYPE bit is set to 1, this CRC is of 6-bits for Enhanced Serial Messages.
20:23 ID3_0/DATA11_8	ID or Data Field When TYPE bit in is set to 0, this field is ID [3:0] for Short Serial Messages and when TYPE bit is set to 1, this field is DATA [11:8] for Enhanced Serial Messages.
24:31 DATA7_0	Data Field Data field [7:0] for short serial messages and enhanced serial messages.

45.3.2.17 DMA Slow Serial Message Time Stamp Read Register (DMA_SMSG_TS)

The resolution of the time stamp is set by programming the Time Stamp Prescaler Value in the *Global Control Register (GBL_CTRL)* register.

DMA Source Size must be set to 32 bits when reading this register (DMA_SMSG_TS) via DMA.

Offset: 0x0058 Access: User read only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	TS															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TS															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 788. DMA Slow Serial Message Time Stamp Read Register (DMA_SMSG_TS)

Table 784. DMA_SMSG_TS field descriptions

Field	Description
0:31 TS	Time Stamp for the received message Indicates the relative time of detection of the falling edge that comes at the end of the calibration pulse (which is also the start of the status and communication nibble pulse) of the last Fast Message that completed the Serial Message. Time Stamp for Slow Serial Messages is not latched when the Slow Serial Message starts but when the Slow Serial Message is completely received.

45.3.2.18 Channel *n* Clock Control Register (CH*n*_CLK_CTRL)

Note: The value of the CH is device-specific. See the Device Configuration chapter for details.

The SENT module generates the Receiver Sensor Clock (or simply Receiver Clock) internally to sample the incoming filtered sensor data. The user software must program the prescaler value (for Receiver Clock) to match the ideal Sensor Tx Clock period. The generated Receiver Clock is compensated for variations, clock drift and jitter that might occur in the Sensor Tx Clock. See *Section 45.4.5: Adjustment for variation in sensor (Tx) clock* for more details.

This register configures the generation and subsequent compensation logic of the Receiver Clock. User must program the contents of this register before enabling the channel by writing to the corresponding channel enable bit in the Channel Enable Register (CHNL_EN).

Offset: 0x0060 + *n**0x10 (*n* = 0 to (CH-1)) Access: Read/Write

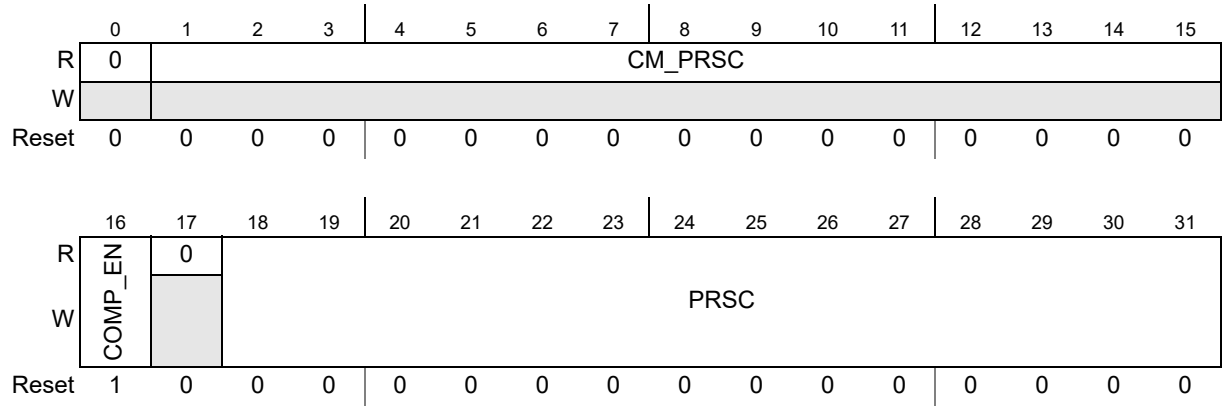


Figure 789. Channel *n* Clock Control Register (CH*n*_CLK_CTRL))

Table 785. CH*n*_CLK_CTRL field descriptions

Field	Description
1:15 CM_PRSC	Compensated Prescaler Value This is the compensated prescaler value calculated by Clock Compensation logic and this value is used to compensate for sensor Tx clock variation. This value is indicated as a read only field for use by user software, if needed. This is a 15-bit value. '= Rx Prescaler Value' – No variation, hence no compensation or Compensation is disabled '< Rx Prescaler Value' – Tx Clock is SLOWER than the programmed receive clock '> Rx Prescaler Value' – Tx Clock is FASTER than the programmed receive clock Thus, the new Receiver Clock Tick Period can be computed by software as Compensated Prescaler Value × High Frequency Receiver Clock Period **Note:** The value provided in this field is ceiling value of actual tick period as measured by the Receiver.

Table 785. CH*n*_CLK_CTRL field descriptions (continued)

Field	Description
16 COMP_EN	Compensation Enable To enable compensation logic to adjust the receiver clock against variation in Tx clock on this channel. User must set the Prescaler Value before enabling this bit. 0 Compensation is disabled. 1 Compensation is enabled.
18:31 PRSC	Rx Prescaler Value This factor is required to generate required Receiver Clock tick from the High Frequency Receiver Clock. This value can only be written when Channel Enable bit is Zero. Hence this value must be programmed before enabling the channel. If required receiver clock tick period (T_{rx_clk}) is not divisible by the high frequency receiver clock (protocol clock) period (T_{hf_clk}), the required prescaler value will have a fractional part which cannot be programmed. In this case then user is advised to program the prescaler value to the lower integer value. 0 Prescaler is bypassed. Should not be used. Non-Zero Value Enables Prescaler The required value is computed by software as follows: **Equation 35 Prescaler_value = floor (Trx_clk / Thf_clk)** (both clock periods taken in µs). Please see *Section 45.5.3: High frequency receiver clock (protocol clock) requirements* before setting this value. For example, if the required receiver clock tick period is 43 µs and the High Frequency Receiver clock is selected to be 62.5 MHz (or 0.016 µs) then, Prescaler Value = 43/0.016 = 2687.5. Hence user must program 2687 in this register.

45.3.2.19 Channel *n* Status Register (CH*n*_STATUS)

This registers stores information about certain events and error conditions that occur in the channel during the message reception process. The assertion of these bit is not linked to any message but it is a general alert to the CPU that a particular condition (error or normal event) happened in the recent past. Messages are automatically discarded when an error is detected.

User software can clear these bit at any time by writing 1 to them. This register can be updated at any time irrespective of the setting in the bit for the corresponding channel.

Note: The value of the CH is device-specific. See the Chapter 7: Device configuration for details.

Offset: 0x0064 + *n**0x10 (*n* = 0 to (CH-1)) Access: Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	BUS_IDLE	0	0	0	CAL_RESYNC	CAL_20_25	SMSG_OFLW	FMSG_OFLW	0	PP_DIAG_ERR	CAL_LEN_ERR	CAL_DIAG_ERR	NIB_VAL_ERR	SMSG_CRC_ERR	FMSG_CRC_ERR	NUM_EDGES_ERR
W	w1c				w1c	w1c	w1c	w1c		w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

Figure 790. Channel *n* Status Register (CH*n*_STATUS)

Table 786. CH*n*_STATUS field descriptions

Field	Description
0 BUS_IDLE	Bus Idle Status This bit indicates that the sensor interface has been idle for more than the period defined by the programmed Bus Idle Count value (in *Channel n Configuration Register (CHn_CONFIG)*). CPU should write 1 to this clear this bit once it is read. 0 Bus is not idle. 1 Channel has been idle for more than the allowed value.
4 CAL_RESYNC	Successive Calibration Check (Option 2 of SAE SENT Spec) Resynchronized This bit indicates that Successive Calibration Check (Option 2 of SAE Spec) has failed three times and has resynchronized to assume 3rd message to be correct and will be received. The successive calibration check error will also be asserted during re-synchronization (to indicate the 3rd error being detected). 0 No interrupt 1 Interrupt Status condition has occurred.
5 CAL_20_25	Calibration Variation 20–25% Interrupt Status This bit indicates that the calibration pulse received on this channel has variation in between ±20% to ±25% from 56 ticks. 0 No Interrupt 1 Interrupt Status condition has occurred.
6 SMSG_OFLW	Slow Serial Message Overflow Status This bit indicates overflow occurred on this channel. Overflow will cause the older messages (which are not read out) to be overwritten with newly received messages. However, any message that is currently being read out by DMA or CPU will not get over-written. 0 No Interrupt 1 Interrupt Status condition has occurred.

Table 786. CH*n*_STATUS field descriptions (continued)

Field	Description
7 FMSG_OFLW	Fast Message Overflow Status This bit indicates overflow occurred on this channel. Overflow happens when effective data read rate by CPU is slower than data receive rate on channel. On overflow, new incoming messages will overwrite the previous messages (that are not read out) stored in the receiver buffers. However, any message that is currently being read out by DMA or CPU will not get over-written. 0 No Interrupt 1 Interrupt Status condition has occurred
9 PP_DIAG_ERR	This diagnostic checks status bit indicates that the ratio of calibration pulse length to overall message length (with pause pulse) is more than ±1.5625% between two messages. This check is valid only for messages with pause pulse. This bit indicates whether this diagnostic has failed or passed on this channel, and is cleared by writing a 1 to it. 0 Error Check has passed. 1 Error Check has failed.
10 CAL_LEN_ERR	This diagnostic checks status bit indicates that Calibration pulse is more than 56 ticks ±25%. This bit indicates whether this diagnostic has failed or passed on this channel, and is cleared by writing a 1 to it. 0 Error Check has passed. 1 Error Check has failed.
11 CAL_DIAG_ERR	Successive Calibration pulses differ by ±1.56%. SAE spec defines it be 1.5625% but the accuracy of this check depends on the frequency of protocol clock (see *Section 45.5.3: High frequency receiver clock (protocol clock) requirements*). This diagnostic checks status bit indicates whether this diagnostic has failed or passed on this channel, and is cleared by writing a 1 to it. 0 Error Check has passed. 1 Error Check has failed.
12 NIB_VAL_ERR	Any nibble data value <0 or >15. This diagnostic checks status bit indicates whether this diagnostic has failed or passed on this channel, and is cleared by writing a 1 to it. 0 Error Check has passed. 1 Error Check has failed.
13 SMSG_CRC_ERR	Checksum error in Slow Serial Message This diagnostic checks status bit indicates whether this diagnostic has failed or passed on this channel, and is cleared by writing a 1 to it. 0 Error Check has passed. 1 Error Check has failed.

Table 786. CH*n*_STATUS field descriptions (continued)

Field	Description
14 FMSG_CRC_ERR	Checksum error in Fast Message This diagnostic checks status bit indicates whether this diagnostic has failed or passed on this channel, and is cleared by writing a 1 to it. 0 Error Check has passed. 1 Error Check has failed.
15 NUM_EDGES_ERR	Not the expected number of negative edges between calibration pulse. This diagnostic checks status bit indicates whether this diagnostic has failed or passed on this channel. This bit will not set in case of “Option 2” for successive calibration pulse check is selected. This error will only set when "Option 1" for successive calibration check is selected (SUCC_CAL_CHK). This bit is cleared by writing a 1 to it. 0 Error Check has passed. 1 Error Check has failed.

45.3.2.20 Channel *n* Configuration Register (CH*n*_CONFIG)

This register (CHn_CONFIG) provides the configurability for interrupt assertion for various events and error as explained in *Channel n Status Register (CHn_STATUS)* and control bits for selection of various receive options that might be required by the user software. This register also configures the width of noise glitches (on the sensor input) to be filtered out by the input programmable filter.

User must program the contents of this register before enabling the channel by writing to the corresponding channel enable bit in the Channel Enable Register (CHNL_EN).

Note: The value of the CH is device-specific. See the Chapter 7: Device configuration for details.

Offset: 0x0068 + *n**0x10 (*n* = 0 to (CH-1)) Access: Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	BUS_IDLE_CNT				IE_CAL_RESYNC	IE_CAL_20_25	IE_SMSG_OFLW	IE_FMSG_OFLW	FCRC_CHK_OFF	IE_PP_DIAG_ERR	IE_CAL_LEN_ERR	IE_CAL_DIAG_ERR	IE_NIB_VAL_ERR	IE_SMSG_CRC_ERR	IE_FMSG_CRC_ERR	IE_NUM_EDGES_ERR
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	DCHNG_INT	CAL_RNG	PP_CHKSEL	FCRC_TYPE	FCRC_SC_EN	SCRC_TYPE	PAUSE_EN	SUCC_CAL_CHK	FIL_CNT							
Reset	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0

Figure 791. Channel *n* Configuration Register (CH*n*_CONFIG)

Table 787. CH*n*_CONFIG field descriptions

Field	Description
0:3 BUS_IDLE_CNT	Bus Idle Count This register value defines the maximum allowable idle period on the sensor interface of this channel. The value is defined as follows: 0000 Disabled 0001 127*2 Receiver Clock Tick Counts 0010 127*4 Receiver Clock Tick Counts 0100 127*8 Receiver Clock Tick Counts 1000 127*16 Receiver Clock Tick Counts
4 IE_CAL_RESYNC	Successive Calibration Check Resynchronized Interrupt Enable This bit enables interrupt assertion when Successive Calibration diagnosis has failed three times in case of “Option 2” being selected for Successive Calibration Check Method and the check resynchronizes the take the third message to be correct. 0 Interrupt is disabled. 1 Interrupt is enabled.
5 IE_CAL_20_25	Calibration Variation 20 - 25% Interrupt Enable This bit enables interrupt assertion when corresponding status bit is set. 0 Interrupt is disabled. 1 Interrupt is enabled.
6 IE_SMSG_OFLW	Slow Serial Message Overflow Interrupt Enable This bit enables interrupt assertion when overflow occurs on reception of Slow Serial Messages in corresponding channel 0 Interrupt is disabled. 1 Interrupt is enabled.
7 IE_FMSG_OFLW	Fast Message Overflow Interrupt Enable This bit enables interrupt assertion when overflow occurs on reception of Fast Messages in corresponding channel. 0 Interrupt is disabled. 1 Interrupt is enabled.
8 FCRC_CHK_OFF	Fast Message CRC Check Off: This bit can be used to switch off CRC check in Fast Message 0 Check is enabled. 1 Check is disabled/off.
9 IE_PP_DIAG_ERR	Ratio of calibration pulse length to message length varies by more than ±1.5625% between two frames. Valid for messages with pause pulse. This bit enables interrupt assertion when its respective diagnostic check fails on the corresponding channel. 0 Interrupt is disabled. 1 Interrupt is enabled.
10 IE_CAL_LEN_ERR	Calibration pulse is wider than 56 ticks ±25%. This bit enables interrupt assertion when its respective diagnostic check fails on the corresponding channel. 0 Interrupt is disabled. 1 Interrupt is enabled.

Table 787. CH*n*_CONFIG field descriptions (continued)

Field	Description
11 IE_CAL_DIAG_ERR	Successive Calibration pulses differ by more than ±1.56%. SAE spec defines it be 1.5625% but the accuracy of this check depends on the protocol clock. For instance for 62.5 MHz, this check will fail for 1.56% and passes for 1.55%. This bit enables interrupt assertion when its respective diagnostic check fails on the corresponding channel. 0 Interrupt is disabled. 1 Interrupt is enabled.
12 IE_NIB_VAL_ERR	Any nibble data value <0 or >15 This bit enables interrupt assertion when its respective diagnostic check fails on the corresponding channel. 0 Interrupt is disabled. 1 Interrupt is enabled.
13 IE_SMSG_CRC_ERR	Checksum error in Slow Serial Message This bit enables interrupt assertion when its respective diagnostic check fails on the corresponding channel. 0 Interrupt is disabled. 1 Interrupt is enabled.
14 IE_FMSG_CRC_ERR	Checksum error in Fast Message This bit enables interrupt assertion when its respective diagnostic check fails on the corresponding channel. 0 Interrupt is disabled. 1 Interrupt is enabled.
15 IE_NUM_EDGES_ERR	Not the expected number of negative edges between calibration pulse. This bit enables interrupt assertion when its respective diagnostic check fails on the corresponding channel. 0 Interrupt is disabled. 1 Interrupt is enabled.
16 DCHNG_INT	Enable for Interrupt on Reception of Fast Message with Changed Data This bit is used to enable storing and generating corresponding DMA or Message Ready Interrupt Request for Fast Message only when any of the data nibbles after different from previously received Fast Message on that channel. If a new Fast Message is received that has same values for all data nibbles as that of previous message then the current message will be discarded and will not be stored in the buffer, and neither Fast Message Ready Interrupt nor DMA request will be generated. 0 All Fast Messages will be received even if data nibbles are same. 1 Only Fast Messages with differing values of data nibbles will be received.
17 CAL_RNG	Valid Calibration Pulse Range Selection This bit is used to control whether 20% variation or 25% variation in Calibration Pulse will be tolerated when detecting a Calibration Pulse 0 20% variation is acceptable. 1 25% variation is acceptable.

Table 787. CH*n*_CONFIG field descriptions (continued)

Field	Description
18 PP_CHKSEL	Pause Pulse Diagnostic Check Selection This bit controls which diagnostic checks to run when messages are enabled to be received with pause pulses. 0 Both Successive Calibration Pulse Check and Pause Pulse Diagnostic are run. 1 Only Pause Pulse Diagnostic is run. Successive Calibration Pulse Check is disabled.
19 FCRC_TYPE	Fast Message CRC Type This indicates the type of CRC method to be used for CRC Diagnostic Check on the received fast messages 0 XOR based implementation (Recommended by SAE Specification) 1 Legacy LUT based implementation
20 FCRC_SC_EN	Fast Message CRC Status and Communication Nibble Enable This bit enables the inclusion of Status and Communication Nibble when CRC is calculated for Fast Messages. 0 Status and Communication Nibble not included in CRC calculation 1 Status and Communication Nibble is included in CRC calculation
21 SCRC_TYPE	Slow Serial Message CRC Type This indicates the type of CRC method to be used for CRC Diagnostic Check on the received slow serial messages. 0 XOR based implementation (Recommended by SAE Specification) 1 Legacy LUT based implementation
22 PAUSE_EN	Pause Pulse Enable Enables the channel receiver to detect a pause pulse. 0 Detection of Pause Pulse is disabled. 1 Detection of Pause Pulse enabled.
23 SUCC_CAL_CHK	Successive Calibration Pulse Check Method This bit indicates the method to be used for performing the successive calibration pulse check. Default value is 1. 0 Option 2 such as Low Latency Option as per SAE Specification 1 Option 1 such as Preferred but High Latency Option as per SAE Specification
24:31 FIL_CNT	Input Filter Sample Count This indicates twice the number of Protocol Clock (of High Frequency Receiver Clock) cycles required for channel input from device's pad to remain stable before it is sampled as 0 or 1. This field defines the width of glitches to be filtered out by the input programmable filter on that channel. Only one bit should be set in this field and the output of filter will be offset by twice the same number of clocks plus additional 3 clocks due to synchronization. Default value is '4'. 0x00: No filtering. Input from device's pad is only synchronized. Non-Zero: Filtering is enabled

45.3.2.21 Channel *n* Fast Message Data Read Register (CH*n*_FMSG_DATA)

Offset: 0x0160 + *n**0x18 (*n* = 0 to (CH-1)) Access: User read only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	CHNUM				SCNIB				DNIB1				DNIB2			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	DNIB3				DNIB4				DNIB5				DNIB6			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. The value of the CH is device-specific. See the Device Configuration chapter for details.

Figure 792. Channel *n* Fast Message Data Read Register (CH*n*_FMSG_DATA)

Table 788. CH*n*_FMSG_DATA field descriptions

Field	Description
0:3 CHNUM	Channel Number Indicates the channel number of the received message being read by DMA. Valid values are 0 to ‘15’
4:7 SCNIB	Status and Communication Nibble of message
8:11 DNIB1	Data Nibble 1 Always supported
12:15 DNIB2	Data Nibble 2 Should be ignored if number of data nibbles supported by channel is 1
16:19 DNIB3	Data Nibble 3 Should be ignored if number of data nibbles supported by channel is 2 or less
20:23 DNIB4	Data Nibble 4 Should be ignored if number of data nibbles supported by channel is 3 or less
24:27 DNIB5	Data Nibble 5 Should be ignored if number of data nibbles supported by channel is 4 or less
28:31 DNIB6	Data Nibble 6 Should be ignored if number of data nibbles supported by channel is 5 or less

45.3.2.22 Channel *n* Fast Message CRC Read Register (CH*n*_FMSG_CRC)

Offset: 0x0164 + *n**0x18 (*n* = 0 to (CH-1)) Access: User read only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	CRC4b			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

1. The value of the CH is device-specific. See the Device Configuration chapter for details.

Figure 793. Channel *n* Fast Message CRC Read Register (CH*n*_FMSG_CRC)

Table 789. CH*n*_FMSG_CRC field descriptions

Field	Description
12:15 CRC4b	4-bit CRC value of the message

45.3.2.23 Channel *n* Fast Message Time Stamp Read Register (CH*n*_FMSG_TS)

Offset: 0x0168 + *n**0x18 (*n* = 0 to (CH-1)) Access: User read only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	TS															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TS															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. The value of the CH is device-specific. See the Device Configuration chapter for details.

Figure 794. Channel *n* Fast Message Time Stamp Read Register (CH*n*_FMSG_TS)

Table 790. CH*n*_FMSG_TS field descriptions

Field	Description
0:31 TS	Time Stamp for received message Indicates the relative time when the falling edge that comes at the end of the calibration pulse (which is also the start of the status and communication nibble pulse) was detected

45.3.2.24 Channel n Slow Serial Message Bit3 Read Register (CHn_SMSG_BIT3)

Offset: 0x016C + *n**0x18 (*n* = 0 to (CH-1)) Access: User read only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	CHNUM				TYPE	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	CFG	ID7_6/ ID3_2		ID5_4/ ID1_0		0	ID3_0/ DATA15_12				0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. The value of the CH is device-specific. See the Device Configuration chapter for details.

Figure 795. Channel *n* Slow Serial Message Bit3 Read Register (CH*n*_SMSG_BIT3)

Table 791. CH*n*_SMSG_BIT3 Field Descriptions

Field	Description
0:3 CHNUM	Channel Number Indicates the channel number of the received message being read by DMA. Valid values are 0 to ‘15’
4 TYPE	Serial Message Type Indicates the type of Serial Message received. 0 Short Serial Message 1 Enhanced Serial Message
20	Placed as 0 by Specification
21 CFG	Configuration bit ‘C’ Indicates the type of Enhanced Serial message. 0 Enhanced Serial Message with 8-bit ID field 1 Enhanced Serial Message with 4-bit ID field
22:23 ID7_6/ID3_2	ID Field Value depends on setting of ‘C’ bit. If ‘C’ bit is 0, this field is ID[7:6] and if the ‘C’ bit is 1, this field is ID[3:2].
24:25 ID5_4/ID1_0	ID Field Value depends on setting of ‘C’ bit. If ‘C’ bit is 0, this field is ID[5:4] and if the ‘C’ bit is 1, this field is ID[1:0].
26	0 as per SAE Specification
27:30 ID3_0/DATA15_12	ID or Data Field Value depends on setting of ‘C’ bit. If ‘C’ bit is 0, this field is ID[3:0] and if the ‘C’ bit is 1, this field is Data[15:12].
31	0 as per SAE Specification

45.3.2.25 Channel n Serial Message Bit2 Read Register (CHn_SMSG_BIT2)

Offset: 0x0170 + *n**0x18 (*n* = 0 to (CH-1)) Access: User read only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	SMCRC					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	ID3_0/ DATA11_8				DATA7_0							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. The value of the CH is device-specific. See the Device Configuration chapter for details.

Figure 796. Channel *n* Slow Serial Message Bit2 Read Register (CH*n*_SMSG_BIT2)

Table 792. CH*n*_SMSG_BIT2 field descriptions

Field	Description
10:15 SMCRC	6-bit CRC value of the message When TYPE bit in *Section 45.3.2.24: Channel n Slow Serial Message Bit3 Read Register (CHn_SMSG_BIT3)* is set to 0, this CRC is of 4-bits for Short Serial Messages and when TYPE bit is set to 1, this CRC is of 6-bits for Enhanced Serial Messages.
20:23 ID3_0/DATA11_8	ID or Data Field When TYPE bit in *Section 45.3.2.24: Channel n Slow Serial Message Bit3 Read Register (CHn_SMSG_BIT3)* is set to 0, this field is ID [3:0] for Short Serial Messages and when TYPE bit is set to 1, this field is DATA [11:8] for Enhanced Serial Messages.
24:31 DATA7_0	Data Field Data field [7:0] for short serial messages and enhanced serial messages.

45.3.2.26 Channel *n* Serial Message Time Stamp Read Register (CH*n*_SMSG_TS)

Offset: 0x0174 + *n**0x18 (*n* = 0 to (CH-1)) Access: User read only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	TS															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TS															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 797. Channel *n* Slow Serial Message Time Stamp Read Register (CH*n*_SMSG_TS)

Table 793. CH*n*_SMSG_TS field descriptions

Field	Description
0:31 TS	Time Stamp for the received message Indicates the relative time of detection of the falling edge that comes at the end of the calibration pulse (which is also the start of the status and communication nibble pulse) of the last Fast Message that completed the Serial Message. Time Stamp for Slow Serial Messages is not latched when the Slow Serial Message starts but when the Slow Serial Message is completely received.

45.3.3 Register Classification for Safety

The registers of this module are classified as per Safety Requirements as follows:

Safety Critical: There are no safety critical registers in this module.

Safety Relevant: The register bits/fields in this category are safety relevant as any random error in these bits will lead to unwanted interference to CPU/DMA. The registers containing these bits must be protected by periodic CRC or equivalent check implemented at SoC Level. The following register bits/fields are classified in this category:

- *Global Control Register (GBL_CTRL)*
- *Channel Enable Register (CHNL_EN)*
- *Fast Message Ready Status Register (FMSG_RDY)*
- *Slow Serial Message Ready Status Register (SMSG_RDY)*
- *Data Control Register 1 (DATA_CTRL1)* and *Data Control Register 2 (DATA_CTRL2)*
- *Fast Message DMA Control Register (FDMA_CTRL)*
- *Slow Serial Message DMA Control Register (SDMA_CTRL)*
- *Fast Message Ready Interrupt Control Register (FRDY_IE)*
- *Slow Serial Message Ready Interrupt Enable Register (SRDY_IE)*
- Fast Message DMA Read Register Set and Slow Message DMA Read Register Set

Safety Latent: The register bits/fields in this category should be protected by LBIST at SoC level. The following register bits/fields are classified in this category:

- *Global Status Register (GBL_STATUS)*
- *Channel n Status Register (CHn_STATUS)*
- *Channel n Configuration Register (CHn_CONFIG)*

Non-Safety: The following register bits are classified in this category:

- All other registers and bits fall in this category

45.4 Functional description

45.4.1 Initialization sequence

All the channels are required to be configured before enabling them by writing into corresponding channel enable bit in the Channel Enable Register (CHNL_EN). There are some registers which must be configured and some which are to be changed if reset value is not as required for that channel. After enabling the channel, no configuration should be changed and if required to be changed, the user needs to first disable the channel and then

reconfigure the channel parameters. The following must to be configured before enabling the corresponding channel:

- Time Stamp Prescaler Value in *Global Control Register (GBL_CTRL)* to generate the time stamping clock
- Rx Prescaler Value in *Channel n Clock Control Register (CHn_CLK_CTRL)* to generate the channel's receiver clock
- Compensation enable in *Channel n Clock Control Register (CHn_CLK_CTRL)*
- Channel Configuration settings in the *Channel n Configuration Register (CHn_CONFIG)*
- Number of Data Nibbles in Channel in the *Data Control Register 1 (DATA_CTRL1)* and *Data Control Register 2 (DATA_CTRL2)*
- Control bits that control whether messages received in channel are to read by interrupt or DMA in *Fast Message DMA Control Register (FDMA_CTRL)*, *Slow Serial Message DMA Control Register (SDMA_CTRL)*, *Fast Message Ready Interrupt Control Register (FRDY_IE)* and *Slow Serial Message DMA Control Register (SDMA_CTRL)*

Note: When configuring the Single Edge Nibble Transmission (SENT) Receiver (SRX) to receive message with the Option 1 of the successive calibration pulse check method (CHn_CONFIG[SUCC_CAL_CHK] = 1), the number of expected edges error (CHn_STATUS[NUM[EDGES_ERR]) gets randomly asserted. Option 2 is not affected as the number of expected edges are not checked in this mode.

The error occurs randomly when the channel input (on the MCU pin) goes from idle to toggling of the calibration pulse.

Note: The Successive Calibration Pulse Check Method Option 1 and Option 2 are defined as follows:

- *Option 1 : Preferred but High Latency Option per SAE specification*
- *Option 2 : Low Latency Option per SAE specification*

To avoid getting the error, the sensor should be enabled first (by the MCU software) and when it starts sending messages, the SENT module should be enabled in the SENT Global Control register (by making GBL_CTRL[SENT_EN] = 1). The delay in start of the two can be controlled by counting a fixed delay in software between enabling the sensor and enabling the SENT module. The first message will not be received but subsequent messages will get received and there will be no false assertions of the number of expected edges error status bit (CHn_STATUS[NUM[EDGES_ERR]).

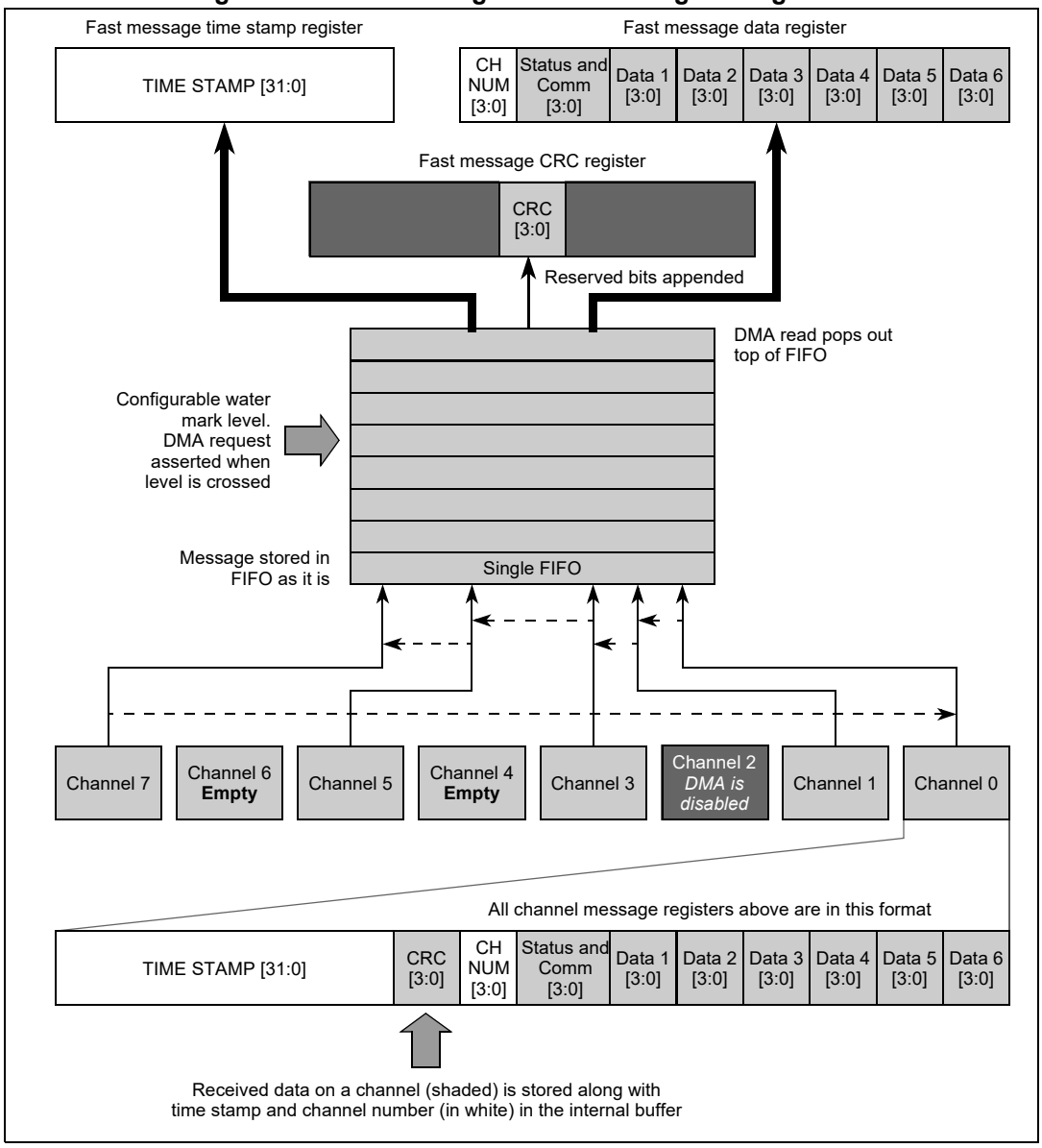
Alternatively, software can count the period from SENT enable (GBL_CTRL[SENT_EN] = 1) to the first expected calibration pulse. If the number of expected edges error status bit (CHn_STATUS[NUM[EDGES_ERR]) is asserted, software can simply clear it as there have no messages which have been completely received.

Alternatively, the software can clear this bit at the start and move ahead. When pause pulse is enabled, then NUM_EDGES will not assert spuriously for subsequent messages which do not have errors in them or cause overflows.

45.4.2 DMA read logic

The SENT module generates separate DMA requests for all Fast Messages and Slow Serial Messages received. The DMA requests are common for all channels. Each packet stored in the channels' buffers are appended with a time stamp taken from a free running time keeping counter and the channel number on which it was received. Details on this time stamp counter can be found in *Section 45.4.8: Time stamp logic*.

Figure 798. Fast Message DMA Read Logic using FIFO



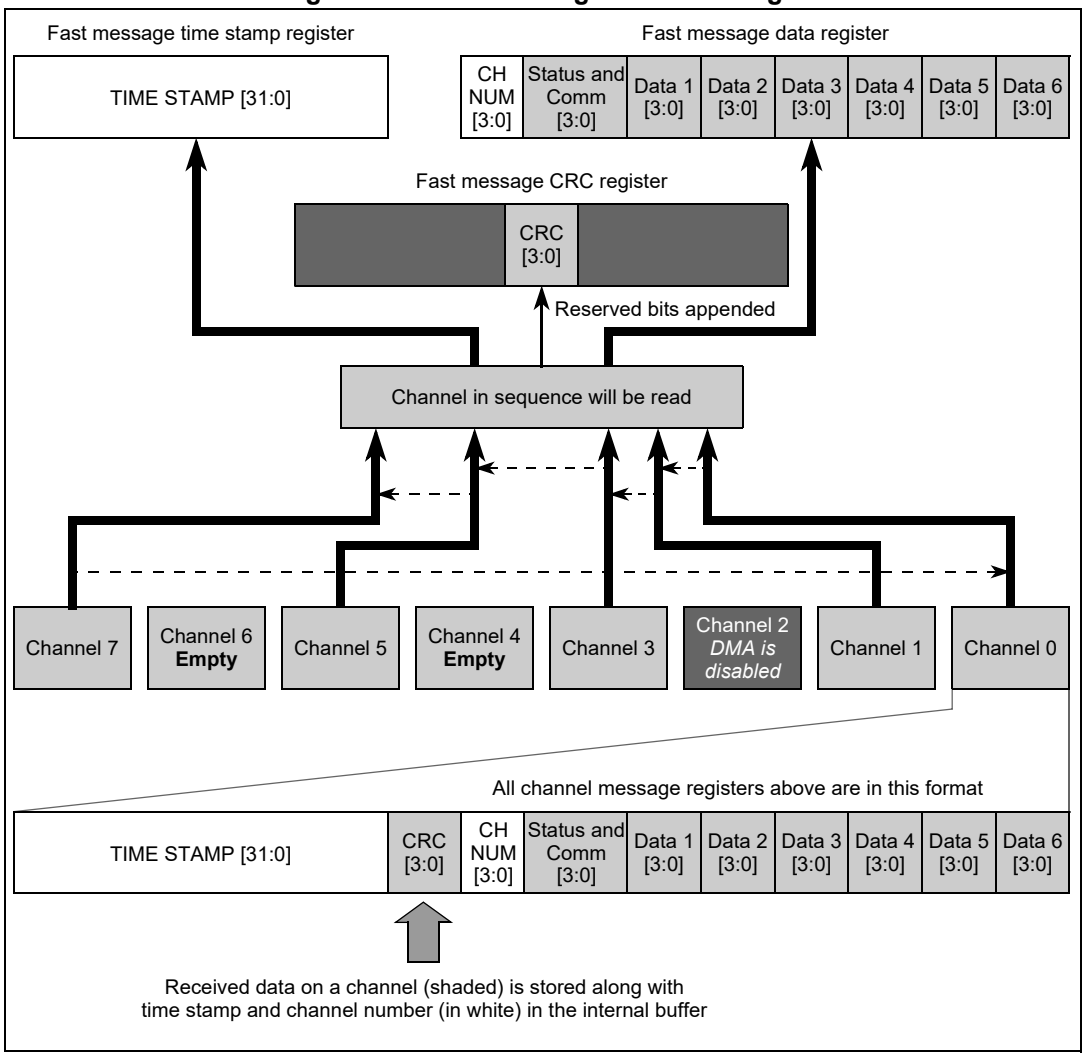
45.4.2.1 DMA request for Fast Message reading

Fast Messages received are stored in a single system bus clock domain buffer to be read out by the DMA Controller via the DMA Fast Message Read register set. Note that the DMA Controller must read all 3 DMA Read Registers to obtain the complete message.

The data flow will happen as follows:

- A DMA request will be asserted when any of the DMA enabled channels has successfully received one complete Fast Message. The request is kept asserted till all messages (across DMA enabled channels) have been read out.
- DMA Controller should read the complete message by accessing each register of the DMA Fast Message Read register set, the registers being:
  - *DMA Fast Message Data Read Register (DMA_FMSG_DATA)*
  - *DMA Fast Message CRC Read Register (DMA_FMSG_CRC)*
  - *DMA Fast Message Time Stamp Read Register (DMA_FMSG_TS)*
- A fixed round robin sequencing runs, as shown in *Figure 799*, and this logic checks for messages in sequence 0→1→3 →5→7 and back to channel 0 and so on. Channel 2 is skipped as it's not DMA enabled) and channel 4 and 6 are skipped as message is not received completely. This sequence continues like this.
- The sequence in which messages are read through DMA interface is not deterministic and depends on the sequence in which they are completely received.
- When the sequencing logic detects a message ready to be read, it will update the DMA Fast Message Read register set for DMA read and move to the next channel when this message is read out completely (such as, 3 read accesses to the above mentioned registers)
- The DMA controller should be programmed to read only one message in one DMA transfer (such as,12 bytes). If it is programmed to read more than one message in one DMA transfer, underflow in buffers might occur
- User software should ensure there is no overflow in channels. In case overflow occurs, the message in the channel buffer will get overwritten by the newly received message. However, when a DMA read is in progress, the buffers will not be overwritten if overflow occurs to maintain data integrity.
- The Round Robin sequencing logic loops through each DMA enabled channel and halts at any channel having data till it is read out by the DMA Controller.

Figure 799. Fast Message DMA read logic



Note: The data buffer related to the SENT data transferred by the DMA in the system memory is in the following order DMA_FMSG_TS, then DMA_FMSG_CRC and then DMA_FMSG_DATA.

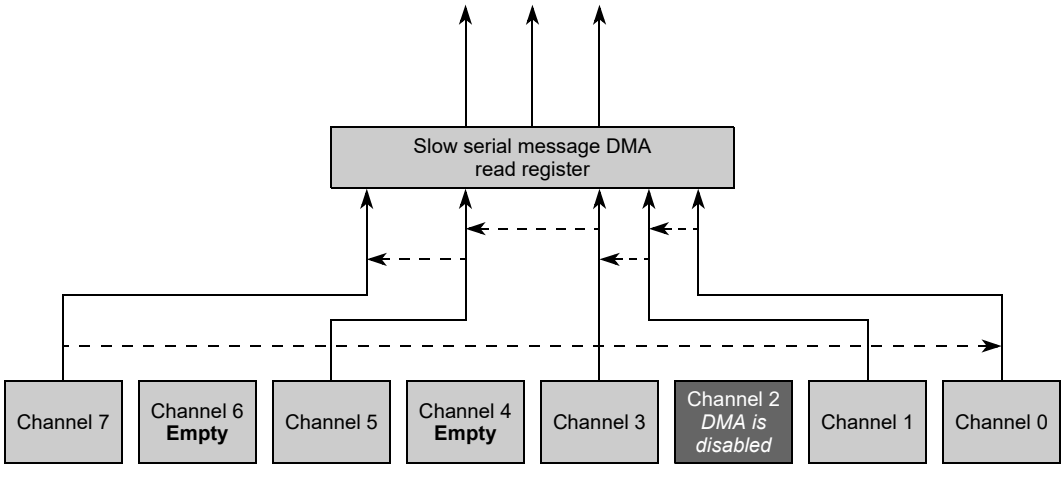
45.4.2.2 DMA request for Slow Serial Message reading

A separate DMA request (different from DMA request for Fast Message) is generated for all slow serial messages received on all channels. The data flow will happen as follows:

- Request will be asserted when any of the DMA enabled channels has successfully received one complete serial message (short or enhanced) and kept asserted till all messages have been read from their respective memory-mapped registers.
- Similar to DMA read for Fast Messages, data for slow serial messages will be read from DMA Slow Serial Message Read register set in memory map, the register set comprising of:
  - *DMA Slow Serial Message Bit3 Read Register (DMA_SMSG_BIT3)*
  - *DMA Slow Serial Message Bit2 Read Register (DMA_SMSG_BIT2)*
  - *DMA Slow Serial Message Time Stamp Read Register (DMA_SMSG_TS)*
- A fixed round robin sequencing logic will be running and it will update the DMA Slow Serial Message Read register set for DMA read and move to the next channel when this message is read out completely (such as, 3 DMA read accesses to the above mentioned registers).
- The channel number and time stamp will also be appended for every message.
- A message type bit (TYPE) will also be appended to distinguish between short and enhanced serial messages.
- The DMA controller should be programmed to read only one message in one DMA transfer (such as, 12 bytes). If it is programmed to read more than one message in one DMA transfer, underflow in buffers might occur.
- Since serial messages receive rate is very slow, user software should ensure there is no overflow.
- The fixed round robin sequence logic for DMA read is the same as that used for Fast Messages (see *DMA request for Fast Message reading*).

Figure 800 shows the sequencing of channels for serial messages to be read when DMA read is done on DMA Slow Serial Message Read register set.

Figure 800. Slow Serial Message DMA read logic



Note: The arrows in the figure indicate next DMA transfer. One DMA Read Register set (3 registers) will be read in one transfer.

The formats in which the 3 types of Serial Messages will be stored in the buffers are shown in figures below.

Figure 801. Enhanced Serial Message with 8-bit ID field

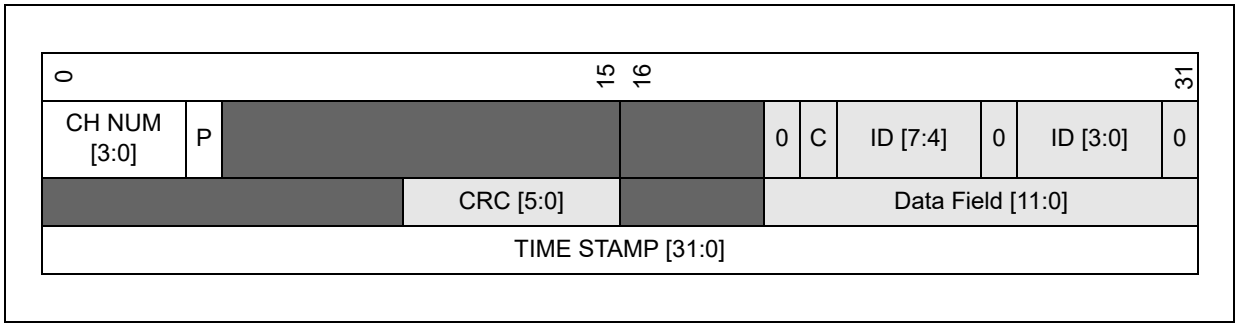


Figure 802. Enhanced Serial Message with 4-bit ID field

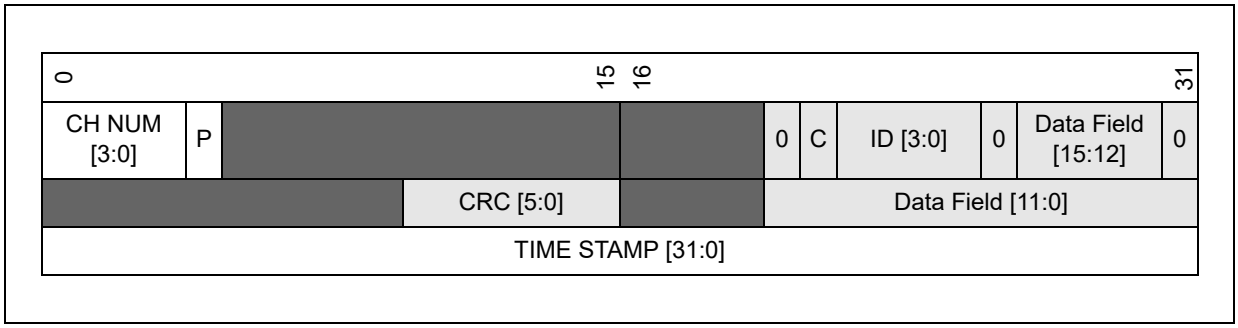
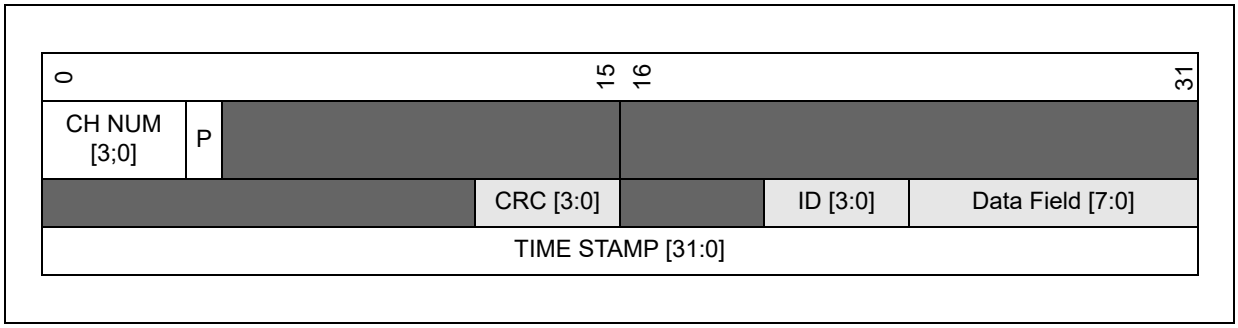


Figure 801 and *Figure 802* shows the enhanced serial messages. Bit C (shown in figure) or bit CFG (as shown in *DMA Slow Serial Message Bit3 Read Register (DMA_SMSG_BIT3)* description) defines the type of enhanced packet and bit P (shown in figure) or bit TYPE (shown in *DMA Slow Serial Message Bit3 Read Register (DMA_SMSG_BIT3)* description) defines whether it is short serial or enhanced serial message. CRC, ID and Data fields are stored in format as received and shown in SAE specification with just one exception that CRC is shifted to align with 16-bit boundary.

Figure 803 shows the short serial message. Field locations have been placed in such a way to allow a common register to read all the slow serial messages and also match the bit positions in actual message defined by SAE Specification.

Figure 803. Short serial message



Three 32-bit accesses to be made by DMA to get one complete message with time stamp.

45.4.3 Message reading via interrupts

There is one interrupt request per channel for a Fast message received and one for a serial message received. Each channel is individually configurable to enable or disable interrupt on reception of Fast or Slow Serial messages separately. In case of interrupt request of Fast Message, an interrupt will be asserted when any of the channels has received a complete Fast Message without errors and that channel is not configured for DMA Reads. The interrupt routine should be coded in such a way to take the least amount of overhead (in CPU cycles) while entering the routine and start reading the available messages. The routine should be capable of accessing all channels that are enabled to give out messages via interrupts through the Register Interface.

In response to these interrupts, the CPU is expected to read interrupt status register for fast or Slow Messages (depending on which interrupt was asserted) and store it in system memory. Using this status register value, the interrupt routine can decide from which channels the message is to be read. For instance if bit corresponding to channel 0 is set then routine should read from the address location corresponding to Channel 0 Fast Message Read registers and move onto the next asserted bit. The interrupt routine should scan through all the bits of this stored interrupt status register value and read corresponding channel's fast or Slow Message registers. In this manner all available messages can be read. After scanning through all the bits of status register stored in system memory, routine can read again the message ready interrupt status register to find out if any other channel has received in the mean time. This will save another interrupt routine time mainly from stacking/unstacking delay.

It is assumed that the interrupt controller will be able to respond fast enough to avoid any overflow as only one message per channel will be stored and the system bus clock frequency is sufficient to avoid this.

45.4.3.1 Suggested software interrupt service routine steps

These are suggested step to have an interrupt routine that can read all messages quickly and avoid any underflow/overflow. Actual implementations can vary.

When separate Fast Message and Slow Serial Message Interrupts are used:

The Fast Message Interrupt should have higher priority than Slow Serial Message Interrupt. The suggested ISR steps for a Fast Message Interrupts can be:

- Read the Fast Message Ready Register and store it in a variable
- Shift out bits starting from MSB from this variable
- If bit is 1, perform read operation on corresponding channel's memory-mapped Fast Message Read Register set (3 registers). Three read accesses will fetch complete message. Shift out new bit. Repeat Step.
- Else if bit is 0, shift out new bit
- When all bits are shifted out, read the Fast Message Status Registers again to ensure no new message has arrived during the ISR.
- If there are no bits set, exit the ISR, else repeat this process.

The ISR steps for Slow Serial Message Interrupt can be same as for Fast Message ISR.

When common Fast Message and Slow Serial Message Interrupts are used:

The suggested ISR steps can be:

- Read the Fast Message Status Register and store it in a variable
- Shift out bits starting from MSB from this variable
- If bit is 1, perform read operation on corresponding channel's memory-mapped Fast Message Read Register. Threeread accesses will fetch complete message . Shift out new bit. Repeat Step.
- Else if bit is 0, shift out new bit
- When all bits are shifted out, read the Slow Serial Message Status Register into a new variable
- Shift out bits starting from MSB from this new variable
- If bit is 1, perform read operation on corresponding channel's memory-mapped Slow Serial Message Read Register. Three read accesses will fetch complete message. Shift out new bit. Repeat Step.
- Else if bit is 0, shift out new bit
- When all bits are shifted out, software can again check both Fast and Slow Serial Message Ready register to ready any message that might have been received while the routine was being serviced, or can choose to exit the routine.

45.4.4 Overflow behavior

This section describes the overflow behavior.

45.4.4.1 Fast messages buffers

Overflow in Fast Message buffers can occur when the CPU or DMA does not read messages from that channel for a long period of time. The receiver channel stores two messages in its buffers and indicates an overflow when these buffers are full.

When an overflow event occurs, the newly received message overwrite the previously stored message and the overflow status bit is set. However, if overflow asserts while CPU/DMA is reading that channel (such as, all **3** read registers have not yet been read), then that message is not overwritten. While messages are being overwritten, the slow message reception is not hampered.

45.4.4.2 Slow message buffers

Overflow in Slow Message buffers can occur when the CPU or DMA does not read messages from that channel for a long period of time. The receiver channel stores two messages in its buffers and indicates an overflow when these buffers are full.

When an overflow event occurs, further reception of slow messages is halted until the buffers are read out by CPU or DMA. Since it is less likely that CPU or DMA do not read the read registers for 32 or 36 fast message length duration, an overflow in slow messages would not be as frequent as in fast messages.

45.4.5 Adjustment for variation in sensor (Tx) clock

The clock used by the Channel Receiver (referred to as Receiver Clock, or Rx_CLK) to sample the message nibbles is generated from a High Frequency Receiver Clock or Protocol Clock. This high frequency receiver clock is divided using a prescaler counter

which generates a tick when it rollovers on reaching a count equal to the Prescaler Value programmed in the channel's clock control register. The receive clock is generated separately for each channel. By varying this Prescaler Factor, we can control the period of the receive clock and make it close to the Transmitter clock used by the sensor for the particular channel.

After reset and once channel is enabled, the compensated prescaler value is set to Zero and the user software programs the Prescaler Value to obtain the desired receiver clock frequency for the particular channel. Thus, the receiver starts working by assuming that the transmitter is sending messages on the frequency that is programmed in its channel clock control register. The channel's input after filtering is continuously measured by the Prescaler Counter. The compensation logic checks each pulse length and determines the calibration pulse from them. When a calibration pulse is detected, the compensated prescaler value is computed and stored in the Channel's Clock Control Register (*Channel n Clock Control Register (CHn_CLK_CTRL)*). Software can use this value to determine the variation in Tx Clock.

Note: A pause pulse may incidentally be detected as a synchronization/calibration pulse during start up after reset. The correction value would be computed on the pause pulse too but since the next synchronization pulse would follow, the logic would again detect the falling edge on the synchronization pulse and the correction value would now be correctly computed on the correct synchronization pulse.

The Clock Compensation or Correction happens for every messages that is received and the compensated prescaler value remains constant for that message.

Note: At ambient temperatures above 114 degree C, Single Edge Nibble Transmission (SENT) Receiver (SRX) generates CAL_LEN_ERR and NUM_EDGES error when calibration pulse length from sensor is 25% higher than normal length. As temperature is increased, the frequency of occurrence of these errors also increases.

By configuring the SENT high frequency clock in the SRX module to run at a slower frequency (40 MHz) will help to reduce the occurrence of these errors.

45.4.5.1 Adjustment for nibble length variation

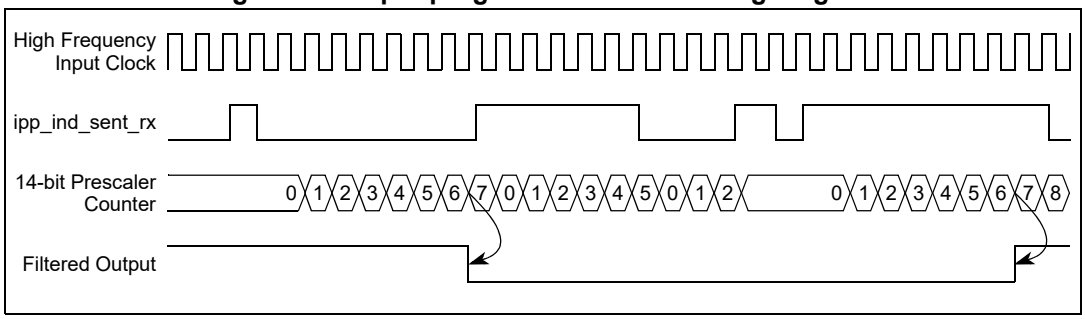
Section 6.2 of SAE Specifications specifies the maximum allowable limits for clock drift and jitter for the transmitter and receiver separately. This can cause the length of nibbles to vary and can cause incorrect sampling of data nibbles. This variation is adjusted by the SENT Receiver when nibbles are being sampled.

45.4.6 Input programmable filter

The Input Programmable Filter ensures that only valid input pin transitions are received by the SENT Receiver module. The input programmable filter is an 8-bit programmable up counter that increments on the high frequency receiver clock. The sensor input signal from device's pad is synchronized by high frequency receiver clock. When a state change occurs in the sensor input signal, the 8-bit counter resets and starts counting up on the high frequency receiver clock. As long as the new state is stable on the pin, the counter remains incrementing. If a counter overflows occurs, the new pin value (such as, sensor input) is validated and latched in the filtered output flop (also running on high frequency receiver clock). If the opposite edge appears on the sensor input before validation (such as, overflow), the counter is reset and the input is not passed onto the filtered output. At the next pin transition, the counter starts counting again. Any pulse that is shorter than a full range of the masked counter is regarded as a glitch and it is not passed on to the filter

output. The number of samples (or the match value of the up counter) is programmable by the user in the channel's configuration register (*Channel n Configuration Register (CHn_CONFIG)*). When zero is programmed into the registers, it bypasses the filter and the synchronized input is output as the filtered output. A non-zero value delays the filtered output by the number of clocks proportional to the programmed value in the Channel Configuration Register (*Channel n Configuration Register (CHn_CONFIG)*). A timing diagram of the input filter when FIL_CNT is 4, is shown in *Figure 804*.

Figure 804. Input programmable filter timing diagram



45.4.7 Receiver diagnostics

Most of the receiver diagnostics are controlled by the Fast Message State Machine. The following checks are conducted by the receiver in each channel.

- Calibration pulse length < 56 clock ticks – 25% or > 56 clock ticks + 25%
- Not the expected number of falling edges between calibration pulses (Message length is pre-defined by each sensor device and programmed by user for each channel)
- Checksum error. Two 4-bit CRC checks and one 6-bit CRC check according to the message type. User has a programmable option to select the method of CRC to use such as, Legacy or XOR-based.
- Any nibble data values measured as < 0 or > 15
- Successive calibration pulses differ by > +1.5625% (1/64) or < -1.5625%. The accuracy of this check will depend on the frequency of the protocol clock used.
- For messages with pause pulse, an additional diagnostic is done; however, this diagnostic will not fail any message (to reduce latency in message reception) but the failure of this diagnostic will be indicated in the channel's status register.

On detection of any of the above errors, the receiver rejects the current packet until a new calibration pulse is detected. Occurrence of an error is flagged in the respective channel's status register (*Channel n Status Register (CHn_STATUS)*). If error interrupt is enabled, an error interrupt will be generated. There is a single error interrupt generated for all errors, so the CPU must read the status register to find out which error occurred. Once error condition is flagged, subsequent errors (except NUM_EDGES_ERR) are automatically masked out until a valid calibration pulse is detected, after which all checks are automatically enabled. This is done to avoid multiple assertions of the interrupt to CPU in case the channel become unstable and send wrong data continuously.

45.4.7.1 Calibration pulse length check

The calibration pulse length check happens in conjunction with Clock Compensation Logic since current calibration pulse is detected in this block. The Clock Compensation Logic

detects a calibration pulse of length between 42 and 70 tick counts. This check will assert the CAL_LEN_ERR status bit on failure.

45.4.7.2 Successive calibration pulse check

The successive calibration pulse check checks if successive calibration pulses differ by > +1.5625% (1/64) or < -1.5625%. The accuracy of this check will depend on the frequency of the protocol clock used. Specifically, the higher the frequency, the closer the check will be to 1.5625%. So if protocol clock is T_{HF_CLK} and channel tick period is T_{TX_CLK} then the check will be of 1.5625% ± {100 × (4 × T_{HF_CLK}) / (42 × T_{TX_CLK})}. So it won't allow variance in clock of a channel by 1.5625% - {100 × (4 × T_{HF_CLK}) / (42 × T_{TX_CLK})}. This check will assert the CAL_DIAG_ERR status bit on failure.

When Option 2 of successive calibration pulse check is enabled, the CAL_RESYNC bit will be asserted on every third successive error detected by this check.

45.4.7.3 Not the expected number of edges check

The number of negative edges between two successive calibration pulses is counted and compared with expected number of edges in case of preferred option of successive calibration pulse check. The expected number of nibbles is the sum of following:

- Status and Communication nibble (1 No.)
- Data nibbles (Programmable as per user software)
- CRC nibble (1 No.)
- Pause Pulse, if configured (1 No.)

This check ("Not the expected number of edges check") will assert the NUM_EDGES_ERR status bit on failure. The check does not run when Option 2 (low latency) of successive calibration pulse check is enabled. This check is not masked after any other error is detected so it could be redundant if it is flagged along with CAL_DIAG_ERR and CAL_LEN_ERR. However, this is true only when CPU is clearing error status bits as and when they are asserted. If errors are accumulated, the above cannot be deduced.

45.4.7.4 Nibble value check

As part of this check, the lengths of the following nibbles are checked to remain in between 0 and 15 ticks:

- Status and Communication nibble
- All Data nibbles
- CRC nibble

The nibble value check will assert the NIB_VAL_ERR status bit on failure. This error can be flagged along with NUM_EDGES_ERR in which case the former can be ignored as the number of pulses in the message is incorrect. However, this is true only when CPU is clearing error status bits as and when they are asserted. In case, errors are accumulated, the above cannot be deduced.

45.4.7.5 CRC check

The CRC check will compute both 4-bit and 6-bit CRC as requested by State Machine control logic. Two 4-bit CRCs and one 6-bit CRC are supported to compute the CRC on Fast/Slow Serial Message (4-bit CRC) and Enhanced Serial Message (6-bit CRC) respectively. The SENT Receiver Module allows the user to calculate the 4-bit CRC using

either the legacy CRC implementation method or the new XOR-based CRC implementation via a control bit in the channel's corresponding control register. Please refer to SAE SENT Specification for details on CRC algorithm.

Asserting the CHn_CONFIG_REG[11] bit, causes the Status and Communication nibble to be included in the 4-bit CRC calculation for Fast Serial Messages. De-asserting this bit enables normal operation as described above.

This check will assert the FMSG_CRC_ERR or SMSG_CRC_ERR status bit on failure, depending on whether the check is run for Fast Message or Slow Message.

The FMSG_CRC_ERR can be flagged along with the NUM_EDGES_ERR, in which case the CRC error can be ignored as the number of pulses in the message was incorrect. However, this is true only when CPU is clearing error status bits as and when they are asserted. If errors are accumulated, the above cannot be deduced.

Note: The Single Edge Nibble Transmitter (SENT) Receiver does not properly round off incoming data to nearest nibble. For example, if the data nibble input to the SENT receiver is 3.5 to 3.87, it is rounded off to 3 instead of 4 as the jitter tolerance is 12.5% instead of 50%. As a result, the Channel n Fast Message Data Read Register (CHn_FMSG_DATA), or Channel n Fast Message Cyclic Redundancy Check (CRC) register (CHn_FMSG_CRC) register values may not match the expected values or the Message is not received at all because of the CRC mismatch.

Disable the CRC check by setting (to 0b1) the Fast Message CRC Check Off bit in the Channel Configuration register (CHn_CONFIG) and perform the CRC check of messages in software.

45.4.7.6 Pause pulse diagnostic

The pause pulse diagnostic checks whether the ratio of calibration pulse length to full message length (including pause pulse) varies from one message to another by less than 1.5625% (as per specification). In order to have reduced latency of operation, message reception is not gated due to this check and no messages are discarded. However, when this check fails, a status bit is asserted to let the software know of this condition. The successive calibration pulse check can be configured to operate along with this check. This configurability is added to avoid conflicting results from the two checks.

There can be some inaccuracy in measuring pause pulse length as SAE specification specifies maximum clock drift/jitter up to maximum nibble length only and the frequency of high frequency receiver clock (protocol clock) used, allows clock compensation logic to ensure accuracy up to maximum nibble length. Hence there could be some inaccuracy in this check when sampling pause pulses that are larger than the maximum nibble pulse length.

This check will assert the PP_DIAG_ERR status bit on failure. This check does not run when reception of pause pulse is disabled.

When the Single Edge Nibble Transmission (SENT) Receiver (SRX) is configured to receive a pause pulse (Channel 'n' Configuration Register - CHn_CONFIG[PAUSE_EN] = 1) and the length of the pause pulse from the sensor is in range of the valid calibration pulse length, the NUM_EDGES error can get asserted spuriously (Channel 'n' Status Register - CHn_STATUS(NUM_EDGES_ERR] = 1) when there is any diagnostic error (other than number of expected edges error) or overflow in the incoming messages from the sensor.

Software can distinguish a spurious NUM_EDGES_ERR error from a real one by monitoring other error bits. The following tables will help distinguish between a false and real assertion

of NUM_EDGES_ERR error and other errors. Software should handle the first error detected as per application needs and other bits can be evaluated based on these tables. *Table 794* contains information due to erratum behavior and *Table 795* below contains clarification of normal NUM_EDGES_ERR behavior.

Table 794. Behavior of NUM_EDGES_ERR

First Error Detected	Other error bits asserted	Cause for extra error bits getting asserted	Action
NIB_VAL_ERR	NUM_EDGES_ERR asserted twice	Upon detection of the first error, the state machine goes into a state where it waits for a calibration pulse, the first NUM_EDGES_ERR error is for the current message as the state machine does not detect an end of message. The second error comes when both the Pause pulse and the Calibration pulse are seen as back to back calibration pulses and no edges in between.	Ignore both NUM_EDGES_ERR error
FMSG_CRC_ERR	NUM_EDGES_ERR asserted twice	Upon detection of the first error, the state machine goes into a state where it waits for a calibration pulse, the first NUM_EDGES_ERR error is for the current message as the state machine does not detect an end of message. The second error comes when both the Pause pulse and the Calibration pulse are seen as back to back calibration pulses and no edges in between.	Ignore both NUM_EDGES_ERR errors
CAL_LEN_ERR	NUM_EDGES_ERR asserted once	Since the calibration pulse is not detected as a valid calibration pulse, the internal edges counter does not detect the end of one message and start of bad message (which has CAL_LEN_ERR); hence the NUM_EDGES_ERR gets asserted.	Ignore NUM_EDGES_ERR error
FMSG_OFLW	NUM_EDGES_ERR asserted once (random occurrence)	A message buffer overflow may lead the state machine to enter a state where it waits for a calibration pulse (behavior also seen in ERR007404). When in this state, the state machine can detect both a Pause pulse and a Calibration pulse as back to back calibration pulses and no edges in between. Then, the NUM_EDGES_ERR can get asserted. Since entry into this state is random, the error can be seen occasionally.	Ignore NUM_EDGES_ERR error

Table 795. Expected behavior, clarification of NUM_EDGES_ERR cases

First Error Detected	Other error bits asserted	Cause for extra error bits getting asserted	Action
NUM_EDGES_ERR (when edges are less than expected)	NIB_VAL_ERR is asserted	When the actual number of edges in the message are less than expected, then a pause pulse gets detected as a nibble since the state machine expects nibbles when actually there is a pause pulse present. This generates NIB_VAL_ERR.	Ignore the NIB_VAL_ERR
NUM_EDGES_ERR (when edges are more than expected)	NIB_VAL_ERR and PP_DIAG_ERR are asserted	When the actual number of edges in a message are more than expected, then after receiving the programmed number of data nibbles, the state machine expects a pause pulse. However, the pause pulse comes later and gets detected as a nibble and hence NIB_VAL_ERR is asserted. Since the message length is not correct, PP_DIAG_ERR is also asserted.	Ignore NIB_VAL_ERR and PP_DIAG_ERR

45.4.8 Time stamp logic

A single counter maintains the time stamp for all channels which can be used by the software to determine the relative time of arrival of the messages for each channel. The time stamp counter is a 32-bit counter.

The clock for this counter is derived from the high frequency receiver clock using a Time Stamp Prescaler Counter that divides the high frequency receiver clock to generate the clock of required resolution. The user must program the time stamp prescaler value (in *Global Control Register (GBL_CTRL)*) in order to generate the time stamp clock. The period of the time stamp clock must be less than that of the fastest receiver channel being used, or it can be set at some fixed value determined by application, such as 1 µs. For example, if the high frequency receiver clock is 60 MHz, the required prescaler value to generate a time stamp clock of 1 µs resolution is 59.

Note: The rollover of time stamp values should be determined in user software, by comparing the time stamp values in previous and current messages that are read out. User software should adjust the time stamp values in these messages accordingly, after reading them.

The time stamp for a Fast Message is sampled at the falling edge that comes at the end of the calibration pulse (which is also the start of the status and communication nibble pulse) and is stored in the message read buffer. However, if the current message is found to be in error, the message buffer is discarded. Thus, the time stamp will be available to user software for only those messages which do not have any error.

The time stamp for a serial message is sampled at the end of the message such as when the serial message is completely received and when the message is found to have no errors. the time stamp is stored in the message read buffer to be read by the user software. Again, for messages with error the message and time stamp are discarded.

45.4.8.1 Limitation

Since the same counter is used for all channels, it may happen that more than 1 message is received within the same time stamp across different channels. Hence the software would not be able to determine the sequence of arrival of messages which have the same time stamp value.

45.4.9 Bus Idle Diagnostic

This diagnostic checks the connected sensors for inactivity. SENT module can indicate if the idle period on a particular channel has crossed the period defined by the programmed value in Bus Idle Count (in *Channel n Configuration Register (CHn_CONFIG)*). Status bit (bus_idle in *Channel n Status Register (CHn_STATUS)*) will be set when idle period crosses the allowed value. CPU should write 1 to this clear this bit once it is read.

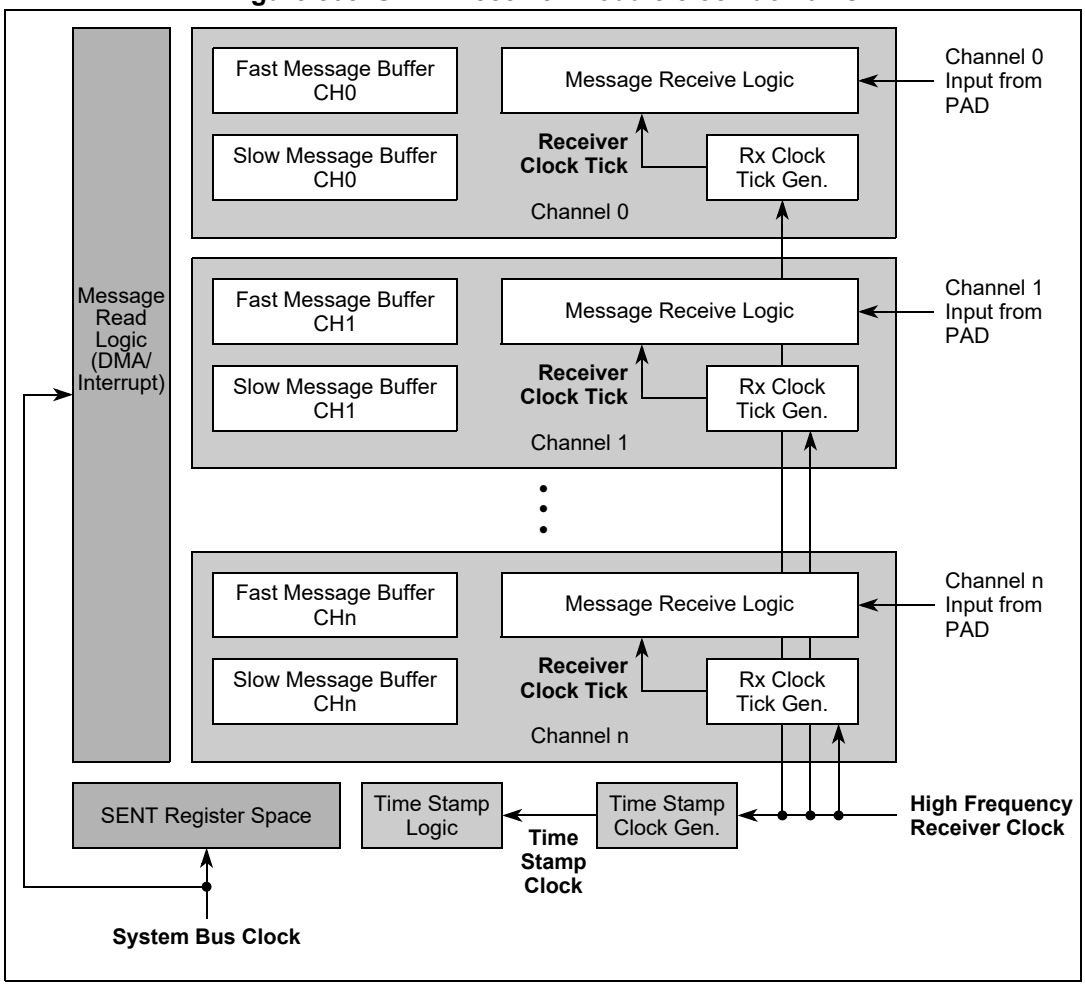
45.5 Clocks and resets

The SENT Receiver works on two clocks. First is the System Bus Clock that is used to program the registers and read messages via DMA or Interrupt from the module's Register Interface. Second is the High Frequency Receiver Clock or Protocol Clock that is used for accurate message receiving operations. There is a single asynchronous reset which is the system asynchronous reset and one internal reset for each of the channel logic.

45.5.1 Clocking strategy

Figure 805 shows the clock domain in which each module is functioning.

Figure 805. SENT Receiver module clock domains



45.5.2 System bus clock requirements

Since all the received messages are read on the System Bus Clock it is necessary that both the interrupt service routine and the DMA controller must be able to access all channels with messages ready to be read within a time to prevent overflow in any channel. The minimum time available to the CPU or DMA is the case when the smallest Fast Message is received back to back on all channels while working on the smallest receive clock tick period.

Now, the smallest Fast Message size with valid checksum as per SAE Specifications is 154 ticks. At 3 µs receive clock tick with a -20% variation in the transmitter clock tick, the effective receiver clock tick is 2.4 µs (after compensation). This gives us the time in which the minimum size Fast Message is received in any channel as 154 $\times$ 2.4 = 369.6 µs. Now since messages are arriving in parallel, 369.6 µs is the time available for CPU or DMA to access all channels to prevent an overflow in the channel that received the very first message.

Thus, if assume that 'N_{CHNL}' channels are being used and all channels have messages ready to be read together from either DMA or CPU then the time available to read a message in one channel will be $369.6/N_{CHNL}$ µs or ~$370/N_{CHNL}$ µs. Since the CPU or the DMA needs to make three 32-bit accesses in order to read a complete message and taking the time required for one 32-bit read access as 'T_{RD}' (assuming it to be same for DMA and CPU); we can determine the minimum frequency for the System Bus Clock as:

For CPU,

Equation 36 $$3 \times T_{RD} + T_{ISR} + T_{WAKUP} < 370/N_{CHNL}$$

where T_{ISR} is the overhead in initiating an ISR and T_{WAKUP} is the time to wakeup from low power mode

For DMA,

Equation 37 $$3 \times T_{RD} + T_{WAKUP} < 370/N_{CHNL}$$

Replacing $T_{RD} = N_{RD} \times T_{SYS_CLK}$ and $T_{ISR} = N_{ISR} \times T_{SYS_CLK}$, where N_{RD} is the number of System Bus Clock cycles in one 32-bit read access, N_{ISR} is the number of System Bus Clock Cycles in initiating an interrupt service routine and T_{SYS_CLK} is the period of the same clock, we get

For CPU,

Equation 38 $$(3 \times N_{RD} + N_{ISR}) \times T_{SYS_CLK} + T_{WAKUP} < 370/N_{CHNL}$$

(All units is µs)

For DMA,

Equation 39 $$3 \times N_{RD} \times T_{SYS_CLK} + T_{WAKUP} < 370/N_{CHNL}$$

(All units is µs)

45.5.3 High frequency receiver clock (protocol clock) requirements

For accurate receive operations, the protocol is required to satisfy a minimum frequency defined as per formula below.

Equation 40 $F_{HF_CLK} > 332 / (3 \times T_{Tx_CLK})$

where F_{HF_CLK} is the High Frequency Receiver Clock or Protocol Clock in MHz and T_{Tx_CLK} is the Transmitter or Sensor Clock Frequency in µs.

Note: T_{TX_CLK} in the above equation should be chosen based on the fastest sensor clock among all channels in the device.

46 FlexRay Communication Controller (FlexRay)

46.1 Introduction

46.1.1 Reference

The following documents are referenced in this chapter. They may be consulted for further information on the FlexRay Communication Controller's operation.

- FlexRay Communications System Protocol Specification, Version 2.1 Rev A[n]
- FlexRay Communications System Electrical Physical Layer Specification, Version 3.0

46.1.2 Color coding

Throughout this chapter, some items are displayed in an italicized, colored font.

FlexRay protocol parameters, constants, and variables are highlighted with *blue italics*. An example is the parameter *gdActionPointOffset*.

FlexRay protocol states are highlighted in *green italics*. An example is the state *POC:normal active*.

46.1.3 Overview

The communication controller (CC) is a FlexRay communication controller that implements the *FlexRay Communications System Protocol Specification, Version 2.1 Rev A.*

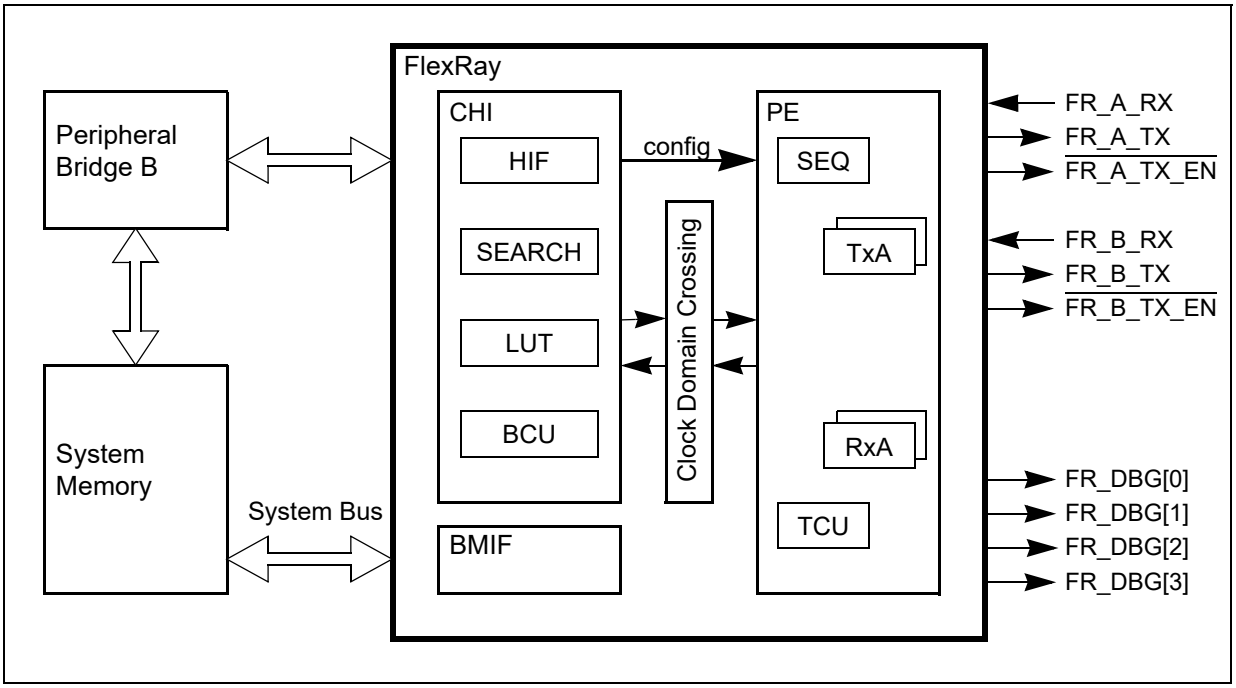
The CC has three main components:

- Controller host interface (CHI)
- Protocol engine (PE)
- Clock domain crossing unit (CDC)

A block diagram of the CC with its surrounding modules is given in *Figure 806*.

n. The FlexRay Specifications have been developed for automotive applications. The FlexRay Specifications have been neither developed nor tested for non-automotive applications.

Figure 806. FlexRay block diagram



The protocol engine has two transmitter units, TxA and TxB, and two receiver units, RxA and RxB, for sending and receiving frames through the two FlexRay channels. The time control unit (TCU) is responsible for maintaining global clock synchronization to the FlexRay network. The overall activity of the protocol engine (PE) is controlled by the sequencer engine (SEQ).

The controller host interface provides host access to the module's configuration, control, and status registers, as well as to the message buffer configuration, control, and status registers. The message buffers themselves, which contain the frame header and payload data received or to be transmitted, as well as the slot status information, are stored in the FlexRay memory area.

The clock domain crossing unit implements signal crossing from the controller host interface (CHI) clock domain to the PE clock domain and vice versa, to allow for asynchronous PE and CHI clock domains.

The CC stores the frame header and payload data of frames received or of frames to be transmitted in the FlexRay memory area. The application accesses the FlexRay memory area to retrieve and provide the frames to be processed by the CC. In addition to the frame header and payload data, the CC stores the synchronization frame related tables in the FlexRay memory area for application processing.

The FlexRay memory area is located in the system memory of the MCU. The CC has access to the FlexRay memory area via its bus master interface (BMIF). The host provides the start address of the FlexRay memory area within the system memory by programming the *Section 46.5.2.5: System Memory Base Address Register (FR_SYMBADR)*. All FlexRay memory area related offsets are stored in offset registers. The physical address pointer into the FlexRay memory area is calculated using the offset values the FlexRay memory base address.

Note: The CC does not provide a memory protection scheme for the FlexRay memory area.

46.1.4 Features

The CC provides the following features:

- FlexRay Communications System Protocol Specification, Version 2.1 Rev A-compliant protocol implementation
- FlexRay Communications System Electrical Physical Layer Specification, Version 3.0-compliant bus driver interface
- Single channel support
  - FlexRay Port A can be configured to be connected either to physical FlexRay channel A or physical FlexRay channel B.
- FlexRay bus data rates of 10 Mbit/s, 8 Mbit/s, 5 Mbit/s, and 2.5 Mbit/s supported
- 64 configurable message buffers with
  - Individual frame ID filtering
  - Individual channel ID filtering
  - Individual cycle counter filtering
- Message buffer header, status, and payload data stored in dedicated FlexRay memory area
  - Allows for flexible and efficient message buffer implementation
  - Ensures consistent data access by means of a buffer-locking scheme
  - Application can lock multiple buffers at the same time
- Size of message buffer payload data section configurable from 0–254 bytes
- Two independent message buffer segments with configurable size of payload data section
  - Each segment can contain message buffers assigned to the static segment and message buffers assigned to the dynamic segment at the same time
- Zero padding for transmit message buffers in static segment
  - Applied when the frame payload length exceeds the size of the message buffer data section
- Transmit message buffers configurable with state/event semantics
- Message buffers can be configured as
  - Receive message buffer
  - Transmit message buffer
- Individual message buffers can be reconfigured
  - Individual message buffers can be disabled safely
  - Disabled message buffers can be reconfigured
- Two independent receive FIFOs
  - One receive FIFO per channel
  - Up to 255 entries for each FIFO
  - Global frame ID filtering, based on both value/mask filters and range filters
  - Global channel ID filtering
  - Global message ID filtering for the dynamic segment
- Four configurable slot error counters
- Four dedicated slot status indicators
  - Used to observe slots without using receive message buffers

- Measured value indicators for clock synchronization
  - Internal synchronization frame ID and synchronization frame measurement tables can be copied into the FlexRay memory area
- Fractional macroticks are supported for clock correction
- Maskable interrupt sources provided via individual and combined interrupt lines
- One absolute timer
- One timer that can be configured as absolute or relative
- SECDED for protocol engine data RAM
- SEDDED for CHI lookup table RAM

46.1.5 Modes of operation

This section describes the basic operational power modes of the CC.

46.1.5.1 Disabled mode

The CC enters Disabled mode during hard reset or when the host clears the 'MEN' field in the Module Configuration Register (FR_MCR). The host can clear this field by writing '0' and only when the module is in *POC:default config* mode. The Disabled mode can be checked by reading the module enable field, MEN, in the Module Configuration Register (FR_MCR).

No communication is performed on the FlexRay bus.

All registers with the write access conditions *Any Time* and *Disabled mode* can be accessed for writing as stated in *Section 46.5: Memory map and register description*.

The application configures the CC by accessing the configuration fields in the Module Configuration Register (FR_MCR) as described in *Section 46.7.2.1: Module initialization*.

46.1.5.1.1 Leave Disabled mode

The CC leaves Disabled mode and enters Normal mode, when the application writes 1 to FR_MCR[MEN].

Note: Once the CC is enabled, it cannot be disabled later.

46.1.5.2 Normal mode

In this mode, the CC is fully functional. The CC indicates that it is in Normal mode by asserting FR_MCR[EN].

46.1.5.2.1 Enter Normal mode

This mode is entered when the application requests that the CC leave Disabled mode. If Normal mode is entered after leaving Disabled mode, the application has to perform the protocol initialization described in *Section 46.7.2.2: Protocol initialization* to achieve full FlexRay functionality.

Depending on the values of the SCM, CHA, and CHB fields in the Module Configuration Register (FR_MCR), the corresponding FlexRay bus driver ports are enabled and driven.

46.1.5.3 System memory test

46.1.5.3.1 Bus Master Interface Functional Test for AMBA/AHB Bus

This section describes the test access to test the AMBA/AHB Bus Master Interface.

46.1.5.3.1.1 AMBA Read Access

This describes the sequence to initiate a AMBA read access

1. write word offset of address to read from into FR_PEDRAR[SYSMEMADDR]
2. and set the read/write indicator FR_PEDRAR[RW] to 1

The module clears the FR_PEDRAR[DAD] bit while access is running

The BMIF initiates a read from 64 bit-aligned byte address:
readaddr = ({SYSMEMADDR,0} + SMBA) & 64'hFFFFFFF8

The module sets the FR_PEDRAR[DAD] bit when access has finished

3. the data read and corresponds to {SYSMEMADDR,0} + SMBA are provided in the FR_PEDRDR register

46.1.5.3.1.2 AMBA Write Access (Prepare)

This describes how to provide data for the AMBA write access.

1. write 16-bit data to be written into FR_PEDRDR register
2. write 16-bit offset of address to be written to into FR_PEDRAR[SYSMEMADDR]
3. and set the read/write indicator FR_PEDRAR[RW] to 0

The module clears the FR_PEDRAR[DAD] bit.

The BMIF buffers this write request,

4. the module sets the FR_PEDRAR[DAD] bit when data are buffered

46.1.5.3.1.3 AMBA Write Access (Initiate)

The BMIF initiates the AMBA write access if any of sequences listed above was executed.
writeaddr = ({SYSMEMADDR,0} + SMBA) & 64'hFFFFFFF8.

Note: The read access to force the buffer write is not necessary in all cases, but suggested to have similar test sequences.

46.1.5.3.1.4 16 bit write initiate

A 16-bit write can be initiated as follows

- write to byte 0 and byte 1:
  initiate read from any address, to empty the buffer
  write with FR_PEDRAR[SYSMEMADDR[2:1]] = 2'h 00
  initiate read from any address, to force buffer write
- write to byte 2 and byte 3:
  initiate read from any address, to empty the buffer
  write with FR_PEDRAR[SYSMEMADDR[2:1]] = 2'h 01
  initiate read from any address, to force buffer write
- write to byte 4 and byte 5:
  initiate read from any address, to empty the buffer

write with FR_PEDRAR[SYSMEMADDR[2:1]] = 2'h 10
initiate read from any address, to force buffer write

- write to byte 6 and byte 7:
  initiate read from any address, to empty the buffer
  write with FR_PEDRAR[SYSMEMADDR[2:1]] = 2'h 11
  initiate read from any address, to force buffer write

46.1.5.3.1.5 32 bit write

- write to byte 0,1,2,3:
  initiate read from any address, to empty the buffer
  write with FR_PEDRAR[SYSMEMADDR[2:1]] = 2'h 00
  write with FR_PEDRAR[SYSMEMADDR[2:1]] = 2'h 01
  initiate read from any address, to force buffer write
- write to byte 4,5,6,7:
  initiate read from any address, to empty the buffer
  write with FR_PEDRAR[SYSMEMADDR[2:1]] = 2'h 10
  write with FR_PEDRAR[SYSMEMADDR[2:1]] = 2'h 11
  initiate read from any address, to force buffer write

46.1.5.3.1.6 64 bit write

- write to byte 0,1,2,3,4,5,6,7:
  initiate read from any address, to empty the buffer
  write with FR_PEDRAR[SYSMEMADDR[2:1]] = 2'h 00
  write with FR_PEDRAR[SYSMEMADDR[2:1]] = 2'h 01
  write with FR_PEDRAR[SYSMEMADDR[2:1]] = 2'h 10
  write with FR_PEDRAR[SYSMEMADDR[2:1]] = 2'h 11
  initiate read from any address, to force buffer write

46.2 External Signal Description

This section lists and describes the CC signals connected to external pins. These signals are summarized in *Table 796* and described in detail in *Section 46.2.1: Detailed signal description*.

Note: The off-chip FR_A_RX, FR_A_TX, and $\overline{\text{FR_A_TX_EN}}$ signals are available on each package option. The availability of the other off-chip signals depends on the package option and is chip-specific.

Table 796. External Signal Properties

Name	Direction	Active	Reset	Function
FR_A_RX	Input	—	—	Receive Data Channel A
FR_A_TX	Output	—	1	Transmit Data Channel A
$\overline{\text{FR_A_TX_EN}}$	Output	Low	1	Transmit Enable Channel A
FR_B_RX	Input	—	—	Receive Data Channel B
FR_B_TX	Output	—	1	Transmit Data Channel B
$\overline{\text{FR_B_TX_EN}}$	Output	Low	1	Transmit Enable Channel B
FR_DBG[0]	Output	—	0	Debug Strobe Signal 0
FR_DBG[1]	Output	—	0	Debug Strobe Signal 1
FR_DBG[2]	Output	—	0	Debug Strobe Signal 2
FR_DBG[3]	Output	—	0	Debug Strobe Signal 3

46.2.1 Detailed signal description

This section provides a detailed description of the CC signals connected to external pins.

46.2.1.1 FR_A_RX – Receive Data Channel A

The FR_A_RX signal carries the receive data for channel A from the corresponding FlexRay bus driver.

46.2.1.2 FR_A_TX – Transmit Data Channel A

The FR_A_TX signal carries the transmit data for channel A to the corresponding FlexRay bus driver.

46.2.1.3 $\overline{\text{FR_A_TX_EN}}$ – Transmit Enable Channel A

The $\overline{\text{FR_A_TX_EN}}$ signal indicates to the FlexRay bus driver that the CC is attempting to transmit data on channel A.

46.2.1.4 FR_B_RX – Receive Data Channel B

The FR_B_RX signal carries the receive data for channel B from the corresponding FlexRay bus driver.

46.2.1.5 FR_B_TX – Transmit Data Channel B

The FR_B_TX signal carries the transmit data for channel B to the corresponding FlexRay bus driver

46.2.1.6 $\overline{\text{FR_B_TX_EN}}$ – Transmit Enable Channel B

The $\overline{\text{FR_B_TX_EN}}$ signal indicates to the FlexRay bus driver that the CC is attempting to transmit data on channel B.

46.2.1.7 FR_DBG[3], FR_DBG[2], FR_DBG[1], FR_DBG[0] – Strobe Signals

These signals provide the selected debug strobe signals. For details on the debug strobe signal selection, refer to *Section 46.6.16: Strobe signal support*.

46.3 Controller Host Interface clocking

The clock for the CHI is derived from the system bus clock and has the same phase and frequency as the system bus clock. There are two constraints for the minimum CHI clock frequency.

- The first constraint corresponds to the number of utilized message buffers and is specified in *Section 46.7.5: Number of usable message buffers*.
- The second constraint corresponds to the value of the TIMEOUT field in the *Section 46.5.2.32: System Memory Access Timeout Register (FR_SYMATOR)* and is specified in *Section 46.7.1.1: Configure System Memory Access Timeout Register (FR_SYMATOR)*.

46.4 Protocol engine clocking

The clock for the protocol engine can be generated by two sources. The first source is the internal crystal oscillator and the second source is an internal PLL. The clock source to be used is selected by the clock source select field, CLKSEL, in the *Section 46.5.2.4: Module Configuration Register (FR_MCR)*

Note: See the device clocking details for the exact clock sources used.

46.4.1 Oscillator clocking

If the protocol engine is clocked by the internal crystal oscillator, a 40 MHz crystal or CMOS-compatible clock must be connected to the oscillator pins. The crystal or clock must fulfill the requirements given by the *FlexRay Communications System Protocol Specification, Version 2.1 Rev A.*

46.4.2 PLL clocking

If the protocol engine is clocked by the internal PLL, the PLL clock should be divided down appropriately to generate the 40 MHz and 180 degree phase-shifted 40 MHz clocks and provide them to the PE. For more details, see the clocking chapter of the device reference manual.

46.5 Memory map and register description

The CC occupies 8 KB (8192 bytes) of address space starting at the CC base address defined by the memory map of the MCU.

Note: The following registers are 16-bit write accessible only:

- *Strobe Signal Control Register (FR_STBSCR)*
- *PE DRAM Access Register (FR_PEDRAR)*
- *PE DRAM Data Register (FR_PEDRDR)*

- *Sync Frame ID Rejection Filter Register (FR_SFIDRFR)*
- *Slot Status Selection Register (FR_SSSR)*
- *Slot Status Counter Condition Register (FR_SSCCR)*
- *Receive Shadow Buffer Index Register (FR_RSBIR)*
- *Receive FIFO Range Filter Configuration Register (FR_RFRFCFR)*
- *Message Buffer Cycle Counter Filter Register (FR_MBCCFRn)*
- *Message Buffer Frame ID Register (FR_MBFIDRn)*
- *Message Buffer Index Register (FR_MBIDXRn)*
- *Message Buffer Data Field Offset Register (FR_MBDORn)*
- *LRAM ECC Error Test Register (FR_LEETRn)*

46.5.1 Memory map

The complete memory map of the CC is shown in *Table 797*. The addresses presented here are the offsets relative to the CC base address, which is defined by the MCU address map.

Table 797. FlexRay memory map

Address Offset	Register Name	Location
Module Configuration and Control		
0x0000	Module Version Register (FR_MVR)	*Section 46.5.2.3*
0x0002	Module Configuration Register (FR_MCR)	*Section 46.5.2.4*
0x0004	System Memory Base Address High Register (FR_SYMBADHR)	*Section 46.5.2.5*
0x0006	System Memory Base Address Low Register (FR_SYMBADLR)	*Section 46.5.2.5*
0x0008	Strobe Signal Control Register (FR_STBSCR)	*Section 46.5.2.6*
0x000A	Reserved	
0x000C	Message Buffer Data Size Register (FR_MBDSR)	*Section 46.5.2.7*
0x000E	Message Buffer Segment Size and Utilization Register (FR_MBSSUTR)	*Section 46.5.2.8*
PE Access Registers		
0x0010	PE DRAM Access Register (FR_PEDRAR)	*Section 46.5.2.9*
0x0012	PE DRAM Data Register (FR_PEDRDR)	*Section 46.5.2.10*
Interrupt and Error Handling		
0x0014	Protocol Operation Control Register (FR_POCR)	*Section 46.5.2.11*
0x0016	Global Interrupt Flag and Enable Register (FR_GIFER)	*Section 46.5.2.12*
0x0018	Protocol Interrupt Flag Register 0 (FR_PIFR0)	*Section 46.5.2.13*
0x001A	Protocol Interrupt Flag Register 1 (FR_PIFR1)	*Section 46.5.2.14*
0x001C	Protocol Interrupt Enable Register 0 (FR_PIER0)	*Section 46.5.2.15*
0x001E	Protocol Interrupt Enable Register 1 (FR_PIER1)	*Section 46.5.2.16*
0x0020	CHI Error Flag Register (FR_CHIERFR)	*Section 46.5.2.17*
0x0022	Message Buffer Interrupt Vector Register (FR_MBIVEC)	*Section 46.5.2.18*

Table 797. FlexRay memory map (continued)

Address Offset	Register Name	Location
0x0024	Channel A Status Error Counter Register (FR_CASERCR)	*Section 46.5.2.19*
0x0026	Channel B Status Error Counter Register (FR_CBSERCR)	*Section 46.5.2.20*
Protocol Status		
0x0028	Protocol Status Register 0 (FR_PSR0)	*Section 46.5.2.21*
0x002A	Protocol Status Register 1 (FR_PSR1)	*Section 46.5.2.22*
0x002C	Protocol Status Register 2 (FR_PSR2)	*Section 46.5.2.23*
0x002E	Protocol Status Register 3 (FR_PSR3)	*Section 46.5.2.24*
0x0030	Macrotick Counter Register (FR_MTCTR)	*Section 46.5.2.25*
0x0032	Cycle Counter Register (FR_CYCTR)	*Section 46.5.2.26*
0x0034	Slot Counter Channel A Register (FR_SLTCTAR)	*Section 46.5.2.27*
0x0036	Slot Counter Channel B Register (FR_SLTCTBR)	*Section 46.5.2.28*
0x0038	Rate Correction Value Register (FR_RTCORVR)	*Section 46.5.2.29*
0x003A	Offset Correction Value Register (FR_OFCORVR)	*Section 46.5.2.30*
0x003C	Combined Interrupt Flag Register (FR_CIFR)	*Section 46.5.2.31*
0x003E	System Memory Access Time-Out Register (FR_SYMATOR)	*Section 46.5.2.32*
Sync Frame Counter and Tables		
0x0040	Sync Frame Counter Register (FR_SFCNTR)	*Section 46.5.2.33*
0x0042	Sync Frame Table Offset Register (FR_SFTOR)	*Section 46.5.2.34*
0x0044	Sync Frame Table Configuration, Control, Status Register (FR_SFTCCSR)	*Section 46.5.2.35*
Sync Frame Filter		
0x0046	Sync Frame ID Rejection Filter Register (FR_SFIDRFR)	*Section 46.5.2.36*
0x0048	Sync Frame ID Acceptance Filter Value Register (FR_SFIDAFVR)	*Section 46.5.2.37*
0x004A	Sync Frame ID Acceptance Filter Mask Register (FR_SFIDAFMR)	*Section 46.5.2.38*
Network Management Vector		
0x004C	Network Management Vector Register 0 (FR_NMVR0)	*Section 46.5.2.39*
0x004E	tNework Management Vector Register 1 (FR_NMVR1)	
0x0050	Network Management Vector Register 2 (FR_NMVR2)	
0x0052	Network Management Vector Register 3 (FR_NMVR3)	
0x0054	Network Management Vector Register 4 (FR_NMVR4)	
0x0056	Network Management Vector Register 5 (FR_NMVR5)	
0x0058	Network Management Vector Length Register (FR_NMVLR)	*Section 46.5.2.40*
Timer Configuration		
0x005A	Timer Configuration and Control Register (FR_TICCR)	*Section 46.5.2.41*

Table 797. FlexRay memory map (continued)

Address Offset	Register Name	Location
0x005C	Timer 1 Cycle Set Register (FR_TI1CYSR)	*Section 46.5.2.42*
0x005E	Timer 1 Macrotick Offset Register (FR_TI1MTOR)	*Section 46.5.2.43*
0x0060	Timer 2 Configuration Register 0 (FR_TI2CR0)	*Section 46.5.2.44*
0x0062	Timer 2 Configuration Register 1 (FR_TI2CR1)	*Section 46.5.2.45*
Slot Status Configuration		
0x0064	Slot Status Selection Register (FR_SSSR)	*Section 46.5.2.46*
0x0066	Slot Status Counter Condition Register (FR_SSCCR)	*Section 46.5.2.47*
Slot Status		
0x0068	Slot Status Register 0 (FR_SSR0)	*Section 46.5.2.48*
0x006A	Slot Status Register 1 (FR_SSR1)	
0x006C	Slot Status Register 2 (FR_SSR2)	
0x006E	Slot Status Register 3 (FR_SSR3)	
0x0070	Slot Status Register 4 (FR_SSR4)	
0x0072	Slot Status Register 5 (FR_SSR5)	
0x0074	Slot Status Register 6 (FR_SSR6)	
0x0076	Slot Status Register 7 (FR_SSR7)	
0x0078	Slot Status Counter Register 0 (FR_SSCR0)	*Section 46.5.2.49*
0x007A	ot Status Counter Register 1 (FR_SSCR1)	
0x007C	Slot Status Counter Register 2 (FR_SSCR2)	
0x007E	Slot Status Counter Register 3 (FR_SSCR3)	
MTS Generation		
0x0080	MTS A Configuration Register (FR_MTSACFR)	*Section 46.5.2.50*
0x0082	MTS B Configuration Register (FR_MTSBCFR)	*Section 46.5.2.51*
Shadow Buffer Configuration		
0x0084	Receive Shadow Buffer Index Register (FR_RSBIR)	*Section 46.5.2.52*
Receive FIFO – Configuration		
0x0086	Receive FIFO Watermark and Selection Register (FR_RFWMSR)	*Section 46.5.2.53*
0x0088	Receive FIFO Start Index Register (FR_RFSIR)	*Section 46.5.2.54*
0x008A	Receive FIFO Depth and Size Register (FR_RFDSR)	*Section 46.5.2.55*
Receive FIFO – Control		
0x008C	Receive FIFO A Read Index Register (FR_RFARIR)	*Section 46.5.2.56*
0x008E	Receive FIFO B Read Index Register (FR_RFBRIR)	*Section 46.5.2.57*

Table 797. FlexRay memory map (continued)

Address Offset	Register Name	Location
Receive FIFO – Filter		
0x0090	Receive FIFO Message ID Acceptance Filter Value Register (FR_RFMIDAFVR)	*Section 46.5.2.58*
0x0092	Receive FIFO Message ID Acceptance Filter Mask Register (FR_RFMIDAFMR)	*Section 46.5.2.59*
0x0094	Receive FIFO Frame ID Rejection Filter Value Register (FR_RFFIDRFVR)	*Section 46.5.2.60*
0x0096	Receive FIFO Frame ID Rejection Filter Mask Register (FR_RFFIDRFMR)	*Section 46.5.2.61*
0x0098	Receive FIFO Range Filter Configuration Register (FR_RFRFCFR)	*Section 46.5.2.62*
0x009A	Receive FIFO Range Filter Control Register (FR_RFRFCTR)	*Section 46.5.2.63*
Dynamic Segment Status		
0x009C	Last Dynamic Transmit Slot Channel A Register (FR_LDTXSLAR)	*Section 46.5.2.64*
0x009E	Last Dynamic Transmit Slot Channel B Register (FR_LDTXSLBR)	*Section 46.5.2.65*
Protocol Configuration		
0x00A0 0x00A2 … 0x00DC	Protocol Configuration Register 0 (FR_PCR0) Protocol Configuration Register 1 (FR_PCR1) … Protocol Configuration Register 30 (FR_PCR30)	*Section 46.5.2.66.1* *Section 46.5.2.66.2* … *Section 46.5.2.66.31*
0x00DE	Stop Watch Count High Register (FR_STPWHR)	*Section 46.5.2.67*
0x00E0	Stop Watch Count Low Register (FR_STPWLR)	*Section 46.5.2.67*
Event Output and StopWatch Configuration		
0x00E2	Protocol Event Output Enable and StopWatch Control Register (FR_PEOER)	*Section 46.5.2.68*
0x00E4	Reserved	
Receive FIFO – Configuration		
0x00E6	Receive FIFO Start Data Offset Register (FR_RFSDOR)	*Section 46.5.2.69*
0x00E8	Receive FIFO System Memory Base Address High Register (FR_RFSYMBADHR)	*Section 46.5.2.70*
0x00EA	Receive FIFO System Memory Base Address Low Register (FR_RFSYMBADLR)	*Section 46.5.2.70*
0x00EC	Receive FIFO Periodic Timer Register (FR_RFPTR)	*Section 46.5.2.71*
Receive FIFO – Control		
0x00EE	Receive FIFO Fill Level and POP Count Register (FR_RFFLPCR)	*Section 46.5.2.72*
ECC Registers		
0x00F0	ECC Error Interrupt Flag and Enable Register (FR_EEIFER)	*Section 46.5.2.73*

Table 797. FlexRay memory map (continued)

Address Offset	Register Name	Location
0x00F2	ECC Error Report and Injection Control Register (FR_EERICR)	*Section 46.5.2.74*
0x00F4	ECC Error Report Address Register (FR_EERAR)	*Section 46.5.2.75*
0x00F6	ECC Error Report Data Register (FR_EERDR)	*Section 46.5.2.76*
0x00F8	ECC Error Report Code Register (FR_EERCR)	*Section 46.5.2.77*
0x00FA	ECC Error Injection Address Register (FR_EEIAR)	*Section 46.5.2.78*
0x00FC	ECC Error Injection Data Register (FR_EEIDR)	*Section 46.5.2.79*
0x00FE	ECC Error Injection Code Register (FR_EEICR)	*Section 46.5.2.80*
0x0100–0x07FE	Not Implemented	
Message Buffers Configuration, Control and Status		
0x0800	Message Buffer Configuration, Control, Status Register 0 (FR_MBCCSR0)	*Section 46.5.2.81*
0x0802	Message Buffer Cycle Counter Filter Register 0 (FR_MBCCFR0)	*Section 46.5.2.82*
0x0804	Message Buffer Frame ID Register 0 (FR_MBFIDR0)	*Section 46.5.2.83*
0x0806	Message Buffer Index Register 0 (FR_MBIDXR0)	*Section 46.5.2.84*
0x0808	Message Buffer Configuration, Control, Status Register 1 (FR_MBCCSR1)	*Section 46.5.2.81*
0x080A	Message Buffer Cycle Counter Filter Register 1 (FR_MBCCFR1)	*Section 46.5.2.82*
0x080C	Message Buffer Frame ID Register 1 (FR_MBFIDR1)	*Section 46.5.2.83*
0x080E	Message Buffer Index Register 1 (FR_MBIDXR1)	*Section 46.5.2.84*
…	…	…
0x09F8	Message Buffer Configuration, Control, Status Register 63 (FR_MBCCSR63)	*Section 46.5.2.81*
0x09FA	Message Buffer Cycle Counter Filter Register 63 (FR_MBCCFR63)	*Section 46.5.2.82*
0x09FC	Message Buffer Frame ID Register 63 (FR_MBFIDR63)	*Section 46.5.2.83*
0x09FE	Message Buffer Index Register 63 (FR_MBIDXR63)	*Section 46.5.2.84*
0x0A00-0x0FFF	Not Implemented	—
0x1000 0x1002 … 0x1086	Message Buffer Data Field Offset Register 0 (FR_MBDOR0) Message Buffer Data Field Offset Register 1 (FR_MBDOR1) … Message Buffer Data Field Offset Register 67 (FR_MBDOR67)	*Section 46.5.2.85*
0x1088-0x108E	Reserved	
0x1090 0x1092 … 0x109A	LRAM ECC Error Test Register 0 (FR_LEETR0) LRAM ECC Error Test Register 1 (FR_LEETR1) … LRAM ECC Error Test Register 5 (FR_LEETR5)	*Section 46.5.2.86*
0x109C–0x1FFE	Not Implemented	

46.5.2 Register descriptions

This section provides detailed descriptions of all registers in ascending address order, presented as 16-bit wide entities

46.5.2.1 Register Reset

All registers except the Message Buffer Cycle Counter Filter Registers (FR_MBCCFRn), Message Buffer Frame ID Registers (FR_MBFIDRn), and Message Buffer Index Registers (FR_MBIDXRn) are reset to their reset value on system reset. The registers mentioned above are located in physical memory blocks and, thus, they are not affected by reset. For some register fields, additional reset conditions exist. These additional reset conditions are described in the detailed description of the register. The additional reset conditions are explained in *Table 798*.

Table 798. Additional register reset conditions

Condition	Description
Protocol RUN Command	The register field is reset when the application writes RUN command “0101” to the POCCMD field in the *Section 46.5.2.11: Protocol Operation Control Register (FR_POCR)*.
Message Buffer Disable	The register field is reset when the application has disabled the message buffer. This happens when the application writes 1 to the message buffer disable trigger bit FR_MBCCSRn[EDT] while the message buffer is enabled (FR_MBCCSRn[EDS] = 1) and the CC grants a disable to the application by clearing the FR_MBCCSRn[EDS] bit.

46.5.2.2 Register write access

This section describes the write access restriction terms that apply to all registers.

46.5.2.2.1 Register write access restriction

For each register bit and register field, the write access conditions are specified in the detailed register description. A description of the write access conditions is given in *Table 799*. If, for a specific register field, none of the given write access conditions is fulfilled, any write attempt to this register field is ignored without any notification. The values of the fields are not changed. The condition term [A or B] indicates that the register or field can be written to if at least one of the conditions is fulfilled. The condition term [A and B] indicates that the register or field can be written to if both conditions are fulfilled.

Table 799. Register Write Access Restrictions

Condition	Indication	Description
Anytime	—	No write access restriction.
Disabled mode	FR_MCR[MEN] = 0	Write access only when CC is in Disabled mode.
Normal mode	FR_MCR[MEN] = 1	Write access only when CC is in Normal mode.

Table 799. Register Write Access Restrictions (continued)

Condition	Indication	Description
POC:config	FR_PSR0[PROTSTATE] = *POC:config*	Write access only when protocol is in the *POC:config* state.
MB_DIS	FR_MBCCSRn[EDS] = 0	Write access only when related Message Buffer is disabled.
MB_LCK	FR_MBCCSRn[LCKS] = 1	Write access only when related Message Buffer is locked.
IDL	FR_EEIRICR[BSY] = 0	Write access only when ECC configuration is idle.
Functional Test Mode	FR_MCR[TMODE] = 1	Write access only when CC is in Functional Test Mode.

46.5.2.2.2 Register Write Access Requirements

All registers can be accessed with 8-bit, 16-bit, and 32-bit wide operations.

For some of the registers, at least a 16-bit wide write access is required to ensure correct operation. This write access requirement is described in the Memory map and register definition section. If an 8-bit wide write access is performed to any of these registers, this access is ignored without notification.

46.5.2.2.3 Internal Register Access

The following memory-mapped registers are used to access multiple internal registers.

- Strobe Signal Control Register (FR_STBSCR)
- Slot Status Selection Register (FR_SSSR)
- Slot Status Counter Condition Register (FR_SSCCR)
- Receive Shadow Buffer Configuration Data

Each of these memory-mapped registers provides a SEL field and a WMD bit. The SEL field is used to select the internal register. The WMD bit controls the write mode. If WMD is set to 0 during the write access, all fields of the internal register are updated. If WMD is set to 1, only the SEL field is changed. All other fields of the internal register remain unchanged. This allows for reading back the values of the selected internal register in a subsequent read access.

46.5.2.3 Module Version Register (FR_MVR)

Offset: 0x0000 Access: Read-only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	CHIVER								PEVER							
W																
Reset	1	0	1	0	0	1	0	1	0	1	1	0	1	0	0	0

Figure 807. Module Version Register (FR_MVR)

This register provides the CC version number. The module version number is derived from the CHI version number and the PE version number.

Table 800. FR_MVR field descriptions

Field	Description
0:7 CHIVER	**CHI Version Number** This field provides the version number of the controller host interface.
8:15 PEVER	**PE Version Number** This field provides the version number of the protocol engine.

46.5.2.4 Module Configuration Register (FR_MCR)

Access: Read/Write
Write: SBFF, SCM, CHB, CHA, ECCE, TMODE, FUM, FAM, CLKSEL, BITRATE: Disabled mode
Offset: 0x0002
Write: MEN: Disabled or *POC:default config*
SFFE: Disabled Mode or *POC:config*

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	MEN	SBFF	SCM	CHB	CHA	SFFE	ECCE	TMODE	FUM	FAM	0	CLKSEL	BITRATE			0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 808. Module Configuration Register (FR_MCR)

This register defines the global configuration of the CC.

Table 801. FR_MCR field descriptions

Field	Description
0 MEN	**Module Enable** This bit indicates whether or not the CC is in Disabled mode. The application requests that the CC leave Disabled mode by writing 1 to this bit Before leaving Disabled mode, the application must configure the SCM, SBFF, CHB, CHA, TMODE, and BITRATE values. For details, see *Section 46.1.5: Modes of operation*. 0 Write: only during *POC:default config*, CC disable Read: CC disabled 1 Write: enable CC Read: CC enabled **Note:** If the CC is enabled, it can only be disabled during mode: POC:default config
1 SBFF	**System Bus Failure Freeze** This bit controls the behavior of the CC in case of a system bus failure. 0 Continue normal operation 1 Transition to Freeze mode
2 SCM	**Single Channel Device Mode** This control bit defines the channel device mode of the CC as described in *Section 46.6.10: Channel device modes*. 0 CC works in Dual-Channel Device mode 1 CC works in Single-Channel Device mode

Table 801. FR_MCR field descriptions (continued)

Field	Description
3 CHB	**Channel Enable** protocol related parameter: *pChannels* The semantic of this control bit depends on the channel device mode controlled by the SCM bit and is given *Table 802*.
4 CHA	**Channel Enable** protocol related parameter: *pChannels* The semantic of this control bit depends on the channel device mode controlled by the SCM bit and is given *Table 802*.
5 SFFE	**Synchronization Frame Filter Enable** This bit controls the filtering for received synchronization frames. For details see *Section 46.6.15: Sync frame filtering*. 0 Synchronization frame filtering disabled 1 Synchronization frame filtering enabled
6 ECCE	**ECC Functionality Enable** This bit controls the ECC memory error detection functionality. For details see *Section 46.6.24: Memory content error detection*. 0 ECC functionality (injection, detection, reporting, response) disabled 1 ECC functionality enabled
7 TMODE	**Test Mode** This control bit is used to enable and disable the functional test mode. For details see *Section 1.2.6.5: Functional Test Mode*. 0 Functional Test Mode disabled 1 Functional Test Mode enabled
8 FUM	**FIFO Update Mode** This bit controls the FIFO update behavior when the interrupt flags FR_GIFER[FAFAIF] and FR_GIFER[FAFBIF] are written by the application (see *Section 46.6.9.8: FIFO update*) 0 FIFOA/FIFOB is updated on writing 1 to FR_GIFER[FAFAIF] /FR_GIFER[FAFBIF] 1 FIFOA/FIFOB) is *not* updated on writing 1 to FR_GIFER[FAFAIF]/FR_GIFER[FAFBIF]
9 FAM	**FIFO Address Mode** This bit controls the location of the system memory base address for the FIFOs (see *Section 46.6.9.2: FIFO configuration*). 0 FIFO Base Address located in *Section 46.5.2.5: System Memory Base Address Register (FR_SYMBADR))* 1 FIFO Base Address located in *Section 46.5.2.70: Receive FIFO System Memory Base Address Register (FR_RFSYMBADR)*

Table 801. FR_MCR field descriptions (continued)

Field	Description
11 CLKSEL	**Protocol Engine Clock Source Select** This bit is used to select the clock source for the protocol engine. 0 PE clock source is generated by on-chip crystal oscillator. 1 PE clock source is generated by on-chip PLL.
12:14 BITRATE	**FlexRay Bus Bit Rate** This field defines the FlexRay bus bit rate. 000 10.0 Mbit/s 001 5.0 Mbit/s 010 2.5 Mbit/s 011 8.0 Mbit/s 100 Reserved 101 Reserved 110 Reserved 111 Reserved

Table 802. FlexRay channel selection

SCM	CHB	CHA	Description
Dual Channel Device modes			
0	0	0	ports FR_A_RX, FR_A_TX, and $\overline{\text{FR_A_TX_EN}}$ not driven by CC ports FR_B_RX, FR_B_TX, and $\overline{\text{FR_B_TX_EN}}$ not driven by CC
	0	1	ports FR_A_RX, FR_A_TX, and $\overline{\text{FR_A_TX_EN}}$ driven by CC—connected to FlexRay channel A ports FR_B_RX, FR_B_TX, and $\overline{\text{FR_B_TX_EN}}$ not driven by CC
	1	0	ports FR_A_RX, FR_A_TX, and $\overline{\text{FR_A_TX_EN}}$ not driven by CC ports FR_B_RX, FR_B_TX, and $\overline{\text{FR_B_TX_EN}}$ driven by CC—connected to FlexRay channel B
	1	1	ports FR_A_RX, FR_A_TX, and $\overline{\text{FR_A_TX_EN}}$ driven by CC—connected to FlexRay channel A ports FR_B_RX, FR_B_TX, and $\overline{\text{FR_B_TX_EN}}$ driven by CC—connected to FlexRay channel B
Single Channel Device mode			
1	0	0	ports FR_A_RX, FR_A_TX, and $\overline{\text{FR_A_TX_EN}}$ not driven by CC ports FR_B_RX, FR_B_TX, and $\overline{\text{FR_B_TX_EN}}$ not driven by CC
	0	1	ports FR_A_RX, FR_A_TX, and $\overline{\text{FR_A_TX_EN}}$ driven by CC—connected to FlexRay channel A ports FR_B_RX, FR_B_TX, and $\overline{\text{FR_B_TX_EN}}$ not driven by CC
	1	0	ports FR_A_RX, FR_A_TX, and $\overline{\text{FR_A_TX_EN}}$ driven by CC—connected to FlexRay channel B ports FR_B_RX, FR_B_TX, and $\overline{\text{FR_B_TX_EN}}$ not driven by CC
	1	1	Reserved

46.5.2.5 System Memory Base Address Register (FR_SYMBADR)

Offset: 0x0004

Access: Read/Write
Write: Disabled mode

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	SMBA															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 809. System Memory Base Address High Register (FR_SYMBADHR)

Offset: 0x0006

Access: Read/Write
Write: Disabled mode

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	SMBA												0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 810. System Memory Base Address Low Register (FR_SYMBADLR)

Note: The system memory base address must be set before the CC is enabled.

The system memory base address registers define the base address of the FlexRay memory area within the system memory. The base address is used by the bus master interface (BMIF) to calculate the physical memory address for system memory accesses.

Table 803. FR_SYMBADR field descriptions

Field	Description
SMBA	**System Memory Base Address** This is the value of the system memory base address for the individual message buffers and sync frame table. This is the value of the system memory base address for the receive FIFO if FR_MCR[FAM] is set to 1. It is defined as a byte address.

46.5.2.6 Strobe Signal Control Register (FR_STBSCR)

Offset: 0x0008

Access: Read/Write(1)
Write: Anytime

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	SEL				0	0	0	ENB	0	0	STBPSEL	
W	WMD															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. 16-bit write access required

Figure 811. Strobe Signal Control Register (FR_STBSCR)

This register is used to assign the individual protocol timing-related strobe signals given in *Table 805* to the external strobe ports. Each strobe signal can be assigned to at most one strobe port. Each write access to registers overwrites the previously-written ENB and STBPSEL values for the signal indicated by SEL. If more than one strobe signal is assigned to one strobe port, the current values of the strobe signals are combined with a binary OR and presented at the strobe port. If no strobe signal is assigned to a strobe port, the strobe

port carries logic 0. For more detailed and timing information refer to *Section 46.6.16: Strobe signal support*.

Note: In Single-Channel Device mode, channel B-related strobe signals are undefined and should not be assigned to the strobe ports.

Table 804. FR_STBSCR field descriptions

Field	Description
0 WMD	**Write Mode** This control bit defines the write mode of this register. 0 Write to all fields in this register on write access. 1 Write to SEL field only on write access.
4:7 SEL	**Strobe Signal Select** This control field selects one of the strobe signals given in *Table 805* to be enabled or disabled and assigned to one of the four strobe ports.
11 ENB	**Strobe Signal Enable** The control bit is used to enable and disable the strobe signal selected by STBSSEL. 0 Strobe signal is disabled and not assigned to any strobe port. 1 Strobe signal is enabled and assigned to the strobe port selected by STBPSEL.
14:15 STBPSEL	**Strobe Port Select** This field selects the strobe port that the strobe signal selected by the SEL is assigned to. All strobe signals that are enabled and assigned to the same strobe port are combined with a binary OR operation. 00 Assign selected signal to FR_DBG[0]. 01 Assign selected signal to FR_DBG[1]. 10 Assign selected signal to FR_DBG[2]. 11 Assign selected signal to FR_DBG[3].

Table 805. Strobe signal mapping

SEL		Description	Channel	Type	Offset(1)	Reference
dec	hex					
0	0x0	Arm	—	value	+1	MT start
1	0x1	mt	—	value	+1	MT start
2	0x2	Cycle start	—	pulse	0	MT start
3	0x3	Minislot start	—	pulse	0	MT start
4	0x4	Slot start	A	pulse	0	MT start
5	0x5		B			
6	0x6	Receive data after glitch filtering	A	value	+4	FR_A_RX
7	0x7		B			FR_B_RX
8	0x8	Channel idle indicator	A	level	+5	FR_A_RX
9	0x9		B			FR_B_RX
10	0xA	Syntax error detected	A	pulse	+4	FR_A_RX
11	0xB		B			FR_B_RX

Table 805. Strobe signal mapping (continued)

SEL		Description	Channel	Type	Offset[1]	Reference
dec	hex					
12	0xC	Content error detected	A	level	+4	FR_A_RX
13	0xD		B			FR_B_RX
14	0xE	Receive FIFO almost-full interrupt signals	A	value	n.a.	RX FIFO A Almost Full Interrupt
15	0xF		B			RX FIFO B Almost Full Interrupt

1. Given in PE clock cycles

46.5.2.7 Message Buffer Data Size Register (FR_MBDSR)

Offset: 0x000C

Access: Read/Write
Write: *POC:config*

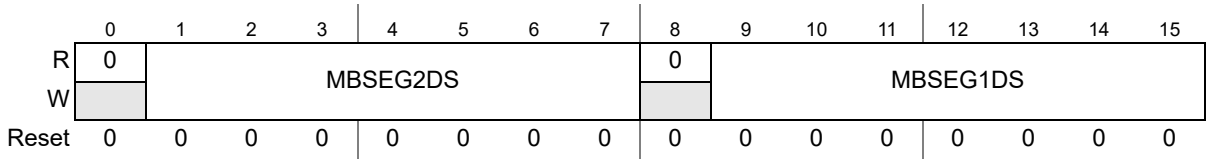


Figure 812. Message Buffer Data Size Register (FR_MBDSR)

This register defines the size of the message buffer data section for the two message buffer segments in a number of two-byte entities.

The CC provides two independent segments for the individual message buffers. All individual message buffers within one segment have to have the same size for the message buffer data section. This size can be different for the two message buffer segments.

Table 806. FR_MBDSR field descriptions

Field	Description
1:7 MBSEG2DS	**Message Buffer Segment 2 Data Size** The field defines the size of the message buffer data section in two-byte entities for message buffers within the *second* message buffer segment.
9:15 MBSEG1DS	**Message Buffer Segment 1 Data Size** The field defines the size of the message buffer data section in two-byte entities for message buffers within the *first* message buffer segment.

46.5.2.8 Message Buffer Segment Size and Utilization Register (FR_MBSSUTR)

Offset: 0x000E

Access: Read/Write
Write: *POC:config*

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	LAST_MB_SEG1						0	0	LAST_MB_UTIL					
W																
Reset	0	0	1	1	1	1	1	1	0	0	1	1	1	1	1	1

Figure 813. Message Buffer Segment Size and Utilization Register (FR_MBSSUTR)

This register defines the last individual message buffer that belongs to the first message buffer segment and the number of the last used individual message buffer.

Table 807. FR_MBSSUTR Field Descriptions

Field	Description
2:7 LAST_MB_SEG1	**Last Message Buffer In Segment 1** This field defines the message buffer number of the last individual message buffer that is assigned to the *first* message buffer segment. The individual message buffers in the *first* segment correspond to the message buffer control registers FR_MBCCSRn, FR_MBCCFRn, FR_MBFIDRn, and FR_MBIDXRn with n ≤ LAST_MB_SEG1. The first message buffer segment contains LAST_MB_SEG1 + 1 individual message buffers. **Note:** The first message buffer segment contains at least one individual message buffer. The individual message buffers in the *second* message buffer segment correspond to the message buffer control registers FR_MBCCSRn, FR_MBCCFRn, FR_MBFIDRn, FR_MBIDXRn with LAST_MB_SEG1 < n < 64. **Note:** If LAST_MB_SEG1 = 63, all individual message buffers belong to the first message buffer segment and the second message buffer segment is empty.
10:15 LAST_MB_UTIL	**Last Message Buffer Utilized** This field defines the message buffer number of last utilized individual message buffer. The message buffer search engine examines all individual message buffer with a message buffer number n ≤ LAST_MB_UTIL. **Note:** If LAST_MB_UTIL=LAST_MB_SEG1, all individual message buffers belong to the first message buffer segment and the second message buffer segment is empty.

46.5.2.9 PE DRAM Access Register (FR_PEDRAR)

Offset: 0x0010

Access: Read/Write(1)
Write: Normal mode or Functional Test mode

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	SYSMEMADDR															DAD
W																RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. 16-bit write access required

Figure 814. PE DRAM Access Register (FR_PEDRAR)

In Functional Test mode this register is used to setup read and write operations to the system memory.

The FR_PEDRAR is used to trigger write and read operations on the PE data memory (PE DRAM). These operations are used for memory error injection and memory error observation.

Each write access to this register initiates a read or write operation on the PE DRAM. The access done status bit, DAD, is cleared after the write access and is set if the PE DRAM access has been finished.

In case of an PE DRAM write access, the data provided in FR_PEDRDR is written into the PE DRAM, read back from the PE DRAM, and are stored into the FR_PEDRDR.

In case of an PE DRAM read access, the requested data is read from PE DRAM and stored into the FR_PEDRDR.

For a detailed description, refer to *Section 46.6.24: Memory content error detection*

Table 808. FR_PEDRAR field descriptions

Field	Description
15 DAD	**PE DRAM Access Done** This status bit is cleared when the application has written to this register and is set when the PE DRAM access has finished. 0 PE DRAM access running 1 PE DRAM access done
0:14 SYSMEMADDR	Functional Test Mode Only **System Memory Address** This field defines the system memory address to be written to or read from.
15 RW	Functional Test Mode Only **Debug Read Write Selector** This control bit defines the access direction for the system memory access. 0 system memory write access 1 system memory read access

46.5.2.10 PE DRAM Data Register (FR_PEDRDR)

Offset: 0x0012

Access: Read/Write(1)
Write: Normal mode or Functional Test Mode

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	DATA															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. 16-bit write access required.

Figure 815. PE DRAM Data Register (FR_PEDRDR)

This register provides the data to be written to or read from the PE DRAM by the access initiated by write access to the FR_PEDRAR.

Table 809. FR_PEDRDR field descriptions

Field	Description
0:15 DATA	Data

46.5.2.11 Protocol Operation Control Register (FR_POCR)

Offset: 0x0014

Access: Read/Write
Write: Normal mode

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	EOC_AP		ERC_AP		BSY_ WMC	0	0	0	POCCMD			
W	WME															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 816. Protocol Operation Control Register (FR_POCR)

The application uses this register to issue:

- Protocol control commands
- External clock correction commands

Protocol control commands are issued by writing to the POCCMD field. For more information on protocol control commands, see *Section 46.7.6: Protocol Control Command Execution*.

External clock correction commands are issued by writing to the EOC_AP and ERC_AP fields. For more information on external clock correction, refer to *Section 46.6.11: External clock synchronization*.

Table 810. FR_POCR Field Descriptions

Field	Description
0 WME	**Write Mode External Correction** This bit controls the write mode of the EOC_AP and ERC_AP fields. 0 Write to EOC_AP and ERC_AP fields on register write 1 No write to EOC_AP and ERC_AP fields on register write
4:5 EOC_AP	**External Offset Correction Application** This field is used to trigger the application of the external offset correction value defined in the Protocol Configuration Register 29 (FR_PCR29). 00 Do not apply external offset correction value 01 Reserved 10 Subtract external offset correction value 11 Add external offset correction value
6:7 ERC_AP	**External Rate Correction Application** This field is used to trigger application of the external rate correction value defined in the Protocol Configuration Register 21 (FR_PCR21). 00 Do not apply external rate correction value 01 Reserved 10 Subtract external rate correction value 11 Add external rate correction value

Table 810. FR_POCR Field Descriptions (continued)

Field	Description
8 BSY_WMC	**Protocol Control Command Write Busy (BSY)** This status bit indicates the acceptance of the protocol control command issued by the application via the POCCMD field. The CC sets this status bit when the application has issued a protocol control command via the POCCMD field. The CC clears this status bit when protocol control command was accepted by the PE. When the application issues a protocol control command while the BSY bit is asserted, the CC ignores this command, sets the protocol command ignored error flag PCMI_EF in the CHI Error Flag Register (FR_CHIERFR), and will not change the value of the POCCMD field. (0) Command write idle, command accepted and ready to receive new protocol command (1) Command write busy, command not yet accepted, not ready to receive new protocol command **Write Mode Command (WMC)** This bit controls the write mode of the POCCMD field. (0) Write to POCCMD field on register write (1) Do not write to POCCMD field on register write
12:15 POCCMD	**Protocol Control Command** The application writes to this field to issue a protocol control command to the PE. The CC sends the protocol command to the PE immediately. While the transfer is running, the BSY bit is set. 0000 ALLOW_COLDSTAR: Immediately activate capability of node to cold start cluster. 0001 ALL_SLOTS: Delayed(1) transition to the All Slots Transmission mode. 0010 CONFIG: Immediately transition to the *POC:config* state. 0011 FREEZE: Immediately transition to the *POC:halt* state. 0100 READY, CONFIG_COMPLETE: Immediately transition to the *POC:ready* state. 0101 RUN: Immediately transition to the *POC:startup start* state. 0110 DEFAULT_CONFIG: Immediately transition to the *POC:default config* state. 0111 HALT: Delayed transition to the *POC:halt* state 1000 WAKEUP: Immediately initiate the wakeup procedure. 1001 Reserved 1010 Reserved 1011 Reserved 1100 Reserved 1101 Reserved 1110 Reserved 1111 Reserved

1. Delayed means on completion of current communication cycle.

46.5.2.12 Global Interrupt Flag and Enable Register (FR_GIFER)

Offset: 0x0016

Access: Read/Write(1)
Write: Normal mode

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	MIF	PRIF	CHIF	WUPIF	FAFBIF	FAFAIF	RBIF	TBIF	MIE	PRIE	CHIE	WUPIE	FAFBIE	FAFAIE	RBIE	TBIE
W				w1c	w1c	w1c										
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. In Functional Test Mode writing 1 toggles the interrupt flags.

Figure 817. Global Interrupt Flag and Enable Register (FR_GIFER)

This register controls some of the interrupt request lines and provides the corresponding interrupt flags. The interrupt flags MIF, PRIF, CHIF, RBIF, and TBIF are the outcome of a binary OR of the related individual interrupt flags and interrupt enables. The generation scheme for these flags is depicted in *Figure 965*. For more details on interrupt generation, see *Section 46.6.20: Interrupt support*. These flags are cleared automatically when all of the corresponding interrupt flags or interrupt enables in the related interrupt flag and enable registers are cleared by the application.

Table 811. FR_GIFER field descriptions

Field	Description
0 MIF	**Module Interrupt Flag** This interrupt flag is set if at least one of the other interrupt flags in this register and the related interrupt enable bit are set. 0 No interrupt flag and related interrupt enable bit are set 1 At least one of the other interrupt flags in this register and the related interrupt bit are set
1 PRIF	**Protocol Interrupt Flag** This interrupt flag is set if at least one of the individual flags in the Protocol Interrupt Flag Register 0 (FR_PIFR0) and Protocol Interrupt Flag Register 1 (FR_PIFR1) and the related interrupt enable bit are set. 0 No individual protocol interrupt flag and related interrupt enable bit are set 1 At least one of the individual protocol interrupt flags and the related interrupt enable bit are set
2 CHIF	**CHI Interrupt Flag** This interrupt flag is set if at least one of the error flags in the CHI Error Flag Register (FR_CHIERFR) and the CHI error interrupt enable bit FR_GIFER[CHIE] are set. 0 All CHI error flags are equal to 0 or the CHI error interrupt is disabled 1 At least one CHI error flag and the CHI error interrupt enable are is set
3 WUPIF	**Wakeup Interrupt Flag** This interrupt flag is set when the CC has received a wakeup symbol on the FlexRay bus. The application can determine on which channel the wakeup symbol was received by reading the related wakeup flags WUB and WUA in the Protocol Status Register 3 (FR_PSR3). 0 No wakeup symbol received on FlexRay bus 1 Wakeup symbol received on FlexRay bus

Table 811. FR_GIFER field descriptions (continued)

Field	Description
4 FAFBIF	**Receive FIFO Channel B Almost Full Interrupt Flag** This interrupt flag is set when one of the following events occurs: a) The current number of FIFO B entries is equal to or greater than the watermark defined by the WM field in the Receive FIFO Watermark and Selection Register (FR_RFWMSR), and the CC writes a received message into the FIFO B, or b) The current number of FIFO B entries is at least 1 and the periodic timer as defined by Receive FIFO Periodic Timer Register (FR_RFPTR) expires. 0 No such event 1 FIFO B almost full event has occurred **Note:** If the bit MCR[FUM] is set to 0, to clear FAFAIF or FAFBIF will also remove one element from the FIFO.
5 FAFAIF	**Receive FIFO Channel A Almost Full Interrupt Flag** This interrupt flag is set when one of the following events occurs: a) The current number of FIFO A entries is equal to or greater than the watermark defined by the WM field in the Receive FIFO Watermark and Selection Register (FR_RFWMSR), and the CC writes a received message into the FIFO A, or b) The current number of FIFO A entries is at least 1 and the periodic timer as defined by Receive FIFO Periodic Timer Register (FR_RFPTR) expires. 0 No such event 1 FIFO A almost full event has occurred
6 RBIF	**Receive Message Buffer Interrupt Flag** This interrupt flag is set if for at least one of the individual receive message buffers (FR_MBCCSRn[MTD] = 0) both the MBIF interrupt flag and the MBIE interrupt enable bit in the corresponding Message Buffer Configuration, Control, Status Registers (FR_MBCCSRn) are asserted. The application can not clear this interrupt flag directly; instead, it is cleared by the CC when all of the MBIF interrupt flags of the individual receive message buffers are cleared by the application, or when the application has cleared the related MBIE interrupt enable bit. 0 None of the individual receive message buffers has the MBIF and MBIE flag set 1 At least one individual receive message buffer has the MBIF and MBIE flag set
7 TBIF	**Transmit Message Buffer Interrupt Flag** This flag is set if for at least one of the individual message buffers (FR_MBCCSRn[MTD] = 1) both the MBIF interrupt flag and the MBIE interrupt enable bit in the corresponding Message Buffer Configuration, Control, Status Registers (FR_MBCCSRn) are equal to 1. The application cannot clear this interrupt flag directly; instead, this interrupt flag is cleared by the CC when all of the individual interrupt flags MBIF of the individual transmit message buffers are cleared by the application, or when the application has cleared the related MBIE interrupt enable bit. 0 None of the individual transmit message buffers has the MBIF and MBIE flag set 1 At least one individual transmit message buffer has the MBIF and MBIE flag set
8 MIE	**Module Interrupt Enable** This bit controls whether the Module Interrupt line is asserted when the MIF flag is set. 0 Disable interrupt line 1 Enable interrupt line
9 PRIE	**Protocol Interrupt Enable** This bit controls whether the Protocol Interrupt line is asserted when the PRIF flag is set. 0 Disable interrupt line 1 Enable interrupt line

Table 811. FR_GIFER field descriptions (continued)

Field	Description
10 CHIE	**CHI Interrupt Enable** This bit controls whether the CHI Interrupt line is asserted when the CHIF flag is set. 0 Disable interrupt line 1 Enable interrupt line
11 WUPIE	**Wakeup Interrupt Enable** This bit controls whether the Wakeup Interrupt line is asserted when the WUPIF flag is set. 0 Disable interrupt line 1 Enable interrupt line
12 FAFBIE	**Receive FIFO Channel B Almost Full Interrupt Enable** This bit controls whether the RX FIFO B Almost Full Interrupt line is asserted when the FAFBIF flag is set. 0 Disable interrupt line 1 Enable interrupt line
13 FAFAIE	**Receive FIFO Channel A Almost Full Interrupt Enable** This bit controls whether the RX FIFO A Almost Full Interrupt line is asserted when the FAFAIF flag is set. 0 Disable interrupt line 1 Enable interrupt line
14 RBIE	**Receive Message Buffer Interrupt Enable** This bit controls whether the Receive Message Buffer Interrupt line is asserted when the RBIF flag is set. 0 Disable interrupt line 1 Enable interrupt line
15 TBIE	**Transmit Message Buffer Interrupt Enable** This bit controls whether the Transmit Message Buffer Interrupt line is asserted when the TBIF flag is set. 0 Disable interrupt line 1 Enable interrupt line

46.5.2.13 Protocol Interrupt Flag Register 0 (FR_PIFR0)

Offset: 0x0018

Access: Read/Write(1)
Write: Normal mode

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	FATL_IF	INTL_IF	ILCF_IF	CSA_IF	MRC_IF	MOC_IF	CCL_IF	MXS_IF	MTX_IF	LTXB_IF	LTXA_IF	TBVB_IF	TBVA_IF	TI2_IF	TI1_IF	CYS_IF
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. In Functional Test Mode writing 1 toggles the interrupt flags.

Figure 818. Protocol Interrupt Flag Register 0 (FR_PIFR0)

This register holds one set of the protocol-related individual interrupt flags.

Table 812. FR_PIFR0 field descriptions

Field	Description
0 FATL_IF	**Fatal Protocol Error Interrupt Flag** This flag is set when the protocol engine has detected a fatal protocol error. In this case, the protocol engine goes into the *POC:halt* state immediately. The fatal protocol errors are: – *pLatestTx* violation, as described in the MAC process of the FlexRay protocol – Transmission across slot boundary violation, as described in the FSP process of the FlexRay protocol 0 No such event 1 Fatal protocol error detected
1 INTL_IF	**Internal Protocol Error Interrupt Flag** This flag is set when the protocol engine has detected an internal protocol error. In this case, the protocol engine goes into the *POC:halt* state immediately. An internal protocol error occurs when the protocol engine has not finished a calculation and a new calculation is requested. This can be caused by a hardware error. 0 No such event 1 Internal protocol error detected
2 ILCF_IF	**Illegal Protocol Configuration Interrupt Flag** This flag is set when the protocol engine has detected an illegal protocol configuration parameter setting. In this case, the protocol engine goes into the *POC:halt* state immediately. The protocol engine checks the *listen_timeout* value programmed into the Protocol Configuration Register 14 (FR_PCR14) and Protocol Configuration Register 15 (FR_PCR15) when the CONFIG_COMPLETE command was sent by the application via the Protocol Operation Control Register (FR_POCR). If the value of *listen_timeout* is equal to 0, the protocol configuration setting is considered as illegal. 0 No such event 1 Illegal protocol configuration detected
3 CSA_IF	**Cold Start Abort Interrupt Flag** This flag is set when the configured number of allowed cold start attempts is reached and none of these attempts was successful. The number of allowed cold start attempts is configured by the coldstart_attempts field in the Protocol Configuration Register 3 (FR_PCR3). 0 No such event 1 Cold start aborted and no more coldstart attempts allowed
4 MRC_IF	**Missing Rate Correction Interrupt Flag** This flag is set when an insufficient number of measurements is available for rate correction at the end of the communication cycle. 0 No such event 1 Insufficient number of measurements for rate correction detected
5 MOC_IF	**Missing Offset Correction Interrupt Flag** This flag is set when an insufficient number of measurements is available for offset correction. This is related to the MISSING_TERM event in the CSP process for offset correction in the FlexRay protocol. 0 No such event 1 Insufficient number of measurements for offset correction detected

Table 812. FR_PIFR0 field descriptions (continued)

Field	Description
6 CCL_IF	**Clock Correction Limit Reached Interrupt Flag** This flag is set when the internal calculated offset or rate calculation values have reached or exceeded their configured thresholds as given by the *offset_coorection_out* field in the Protocol Configuration Register 9 (FR_PCR9) and the *rate_correction_out* field in the Protocol Configuration Register 14 (FR_PCR14). 0 No such event 1 Offset or rate correction limit reached
7 MXS_IF	**Max Sync Frames Detected Interrupt Flag** This flag is set when the number of synchronization frames detected in the current communication cycle exceeds the value of the *node_sync_max* field in the Protocol Configuration Register 30 (FR_PCR30). 0 No such event 1 More than *node_sync_max* sync frames detected **Note:** Only synchronization frames that have passed the synchronization frame acceptance and rejection filters are taken into account
8 MTX_IF	**Media Access Test Symbol Received Interrupt Flag** This flag is set when the MTS symbol was received on channel A or channel B. 0 No such event 1 MTS symbol received
9 LTXB_IF	*pLatestTx* **Violation on Channel B Interrupt Flag** This flag is set when the frame transmission on channel B in the dynamic segment exceeds the dynamic segment boundary. This is related to the *pLatestTx* violation, as described in the MAC process of the FlexRay protocol. 0 No such event 1 *pLatestTx* violation occurred on channel B
10 LTXA_IF	*pLatestTx* **Violation on Channel A Interrupt Flag** This flag is set when the frame transmission on channel A in the dynamic segment exceeds the dynamic segment boundary. This is related to the *pLatestTx* violation as described in the MAC process of the FlexRay protocol. 0 No such event 1 *pLatestTx* violation occurred on channel A
11 TBVB_IF	**Transmission across boundary on channel B Interrupt Flag** This flag is set when the frame transmission on channel B crosses the slot boundary. This is related to the transmission across slot boundary violation as described in the FSP process of the FlexRay protocol. 0 No such event 1 Transmission across boundary violation occurred on channel B
12 TBVA_IF	**Transmission across boundary on channel A Interrupt Flag** his flag is set when the frame transmission on channel A crosses the slot boundary. This is related to the transmission across slot boundary violation as described in the FSP process of the FlexRay protocol. 0 No such event 1 Transmission across boundary violation occurred on channel A
13 TI2_IF	**Timer 2 Expired Interrupt Flag** This flag is set whenever timer 2 expires. 0 No such event 1 Timer 2 has reached its time limit

Table 812. FR_PIFR0 field descriptions (continued)

Field	Description
14 TI1_IF	**Timer 1 Expired Interrupt Flag** This flag is set whenever timer 1 expires. 0 No such event 1 Timer 1 has reached its time limit
15 CYS_IF	**Cycle Start Interrupt Flag** This flag is set when a communication cycle starts. 0 No such event 1 Communication cycle started

46.5.2.14 Protocol Interrupt Flag Register 1 (FR_PIFR1)

Offset: 0x001A

Access: Read/Write(1)
Write: Normal mode

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	EMC_IF	IPC_IF	PECF_IF	PSC_IF	SSI3_IF	SSI2_IF	SSI1_IF	SSI0_IF	0	0	EVT_IF	ODT_IF	0	0	0	0
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c			w1c	w1c				
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. In Functional Test Mode writing 1 toggles the interrupt flags.

Figure 819. Protocol Interrupt Flag Register 1 (FR_PIFR1)

The register holds one set of the protocol-related individual interrupt flags.

Table 813. FR_PIFR1 field descriptions

Field	Description
0 EMC_IF	**Error Mode Changed Interrupt Flag** This flag is set when the value of the ERRMODE bit field in the Protocol Status Register 0 (FR_PSR0) is changed by the CC. 0 No such event 1 ERRMODE field changed
1 IPC_IF	**Illegal Protocol Control Command Interrupt Flag** This flag is set when the PE tries to execute a protocol control command that was issued via the POCCMD field of the Protocol Operation Control Register (FR_POCR), and detects that this protocol control command is not allowed in the current protocol state. In this case, the command is not executed. For more details, see *Section 46.7.6: Protocol Control Command Execution*. 0 No such event 1 Illegal protocol control command detected
2 PECF_IF	**Protocol Engine Communication Failure Interrupt Flag** This flag is set if the CC has detected a communication failure between the PE and the CHI. 0 No such event 1 Protocol Engine Communication Failure detected

Table 813. FR_PIFR1 field descriptions (continued)

Field	Description
3 PSC_IF	**Protocol State Changed Interrupt Flag** This flag is set when the protocol state in the PROTSTATE field in the Protocol Status Register 0 (FR_PSR0) has changed. 0 No such event 1 Protocol state changed
4 SSI3_IF	**Slot Status Counter Incremented Interrupt Flag** This flag is set when the SLOTSTATUSCNT field in the corresponding Slot Status Counter Register (FR_SSCR3) is incremented. 0 No such event 1 The corresponding slot status counter has incremented
5 SSI2_IF	**Slot Status Counter Incremented Interrupt Flag** This flag is set when the SLOTSTATUSCNT field in the corresponding Slot Status Counter Register (FR_SSCR2) is incremented. 0 No such event 1 The corresponding slot status counter has incremented
6 SSI1_IF	**Slot Status Counter Incremented Interrupt Flag** This flag is set when the SLOTSTATUSCNT field in the corresponding Slot Status Counter Register (FR_SSCR1) is incremented. 0 No such event 1 The corresponding slot status counter has incremented
7 SSI0_IF	**Slot Status Counter Incremented Interrupt Flag** This flag is set when the SLOTSTATUSCNT field in the corresponding Slot Status Counter Register (FR_SSCR0) is incremented. 0 No such event 1 The corresponding slot status counter has incremented
10 EVT_IF	**Even Cycle Table Written Interrupt Flag** This flag is set if the CC has written the sync frame measurement/ID tables into the FlexRay memory area for the even cycle. 0 No such event 1 Sync frame measurement table written
11 ODT_IF	**Odd Cycle Table Written Interrupt Flag** This flag is set if the CC has written the sync frame measurement/ID tables into the FlexRay memory area for the odd cycle. 0 No such event 1 Sync frame measurement table written

46.5.2.15 Protocol Interrupt Enable Register 0 (FR_PIER0)

Offset: 0x001C

Access: Read/Write
Write: Anytime

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	FATL_IE	INTL_IE	ILCF_IE	CSA_IE	MRC_IE	MOC_IE	CCL_IE	MXS_IE	MTX_IE	LTXB_IE	LTXA_IE	TBVB_IE	TBVA_IE	TI2_IE	TI1_IE	CYS_IE
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 820. Protocol Interrupt Enable Register 0 (FR_PIER0)

This register defines whether or not the individual interrupt flags defined in the Protocol Interrupt Flag Register 0 (FR_PIFR0) can generate a protocol interrupt request.

Table 814. FR_PIER0 field descriptions

Field	Description
0 FATL_IE	**Fatal Protocol Error Interrupt Enable** This bit controls FATL_IF interrupt request generation. 0 interrupt request generation disabled 1 interrupt request generation enabled
1 INTL_IE	**Internal Protocol Error Interrupt Enable** This bit controls INTL_IF interrupt request generation. 0 interrupt request generation disabled 1 interrupt request generation enabled
2 ILCF_IE	**Illegal Protocol Configuration Interrupt Enable** This bit controls ILCF_IF interrupt request generation. 0 interrupt request generation disabled 1 interrupt request generation enabled
3 CSA_IE	**Cold Start Abort Interrupt Enable** This bit controls CSA_IF interrupt request generation. 0 interrupt request generation disabled 1 interrupt request generation enabled
4 MRC_IE	**Missing Rate Correction Interrupt Enable** This bit controls MRC_IF interrupt request generation. 0 interrupt request generation disabled 1 interrupt request generation enabled
5 MOC_IE	**Missing Offset Correction Interrupt Enable** This bit controls MOC_IF interrupt request generation. 0 interrupt request generation disabled 1 interrupt request generation enabled
6 CCL_IE	**Clock Correction Limit Reached Interrupt Enable** This bit controls CCL_IF interrupt request generation. 0 interrupt request generation disabled 1 interrupt request generation enabled

Table 814. FR_PIER0 field descriptions (continued)

Field	Description
7 MXS_IE	**Max Sync Frames Detected Interrupt Enable** This bit controls MXS_IF interrupt request generation. 0 interrupt request generation disabled 1 interrupt request generation enabled
8 MTX_IE	**Media Access Test Symbol Received Interrupt Enable** This bit controls MTX_IF interrupt request generation. 0 interrupt request generation disabled 1 interrupt request generation enabled
9 LTXB_IE	*pLatestTx* **Violation on Channel B Interrupt Enable** This bit controls LTXB_IF interrupt request generation. 0 interrupt request generation disabled 1 interrupt request generation enabled
10 LTXA_IE	*pLatestTx* **Violation on Channel A Interrupt Enable** This bit controls LTXA_IF interrupt request generation. 0 interrupt request generation disabled 1 interrupt request generation enabled
11 TBVB_IE	**Transmission across boundary on channel B Interrupt Enable** This bit controls TBVB_IF interrupt request generation. 0 interrupt request generation disabled 1 interrupt request generation enabled
12 TBVA_IE	**Transmission across boundary on channel A Interrupt Enable** This bit controls TBVA_IF interrupt request generation. 0 interrupt request generation disabled 1 interrupt request generation enabled
13 TI2_IE	**Timer 2 Expired Interrupt Enable** This bit controls TI1_IF interrupt request generation. 0 interrupt request generation disabled 1 interrupt request generation enabled
14 TI1_IE	**Timer 1 Expired Interrupt Enable** This bit controls TI1_IF interrupt request generation. 0 interrupt request generation disabled 1 interrupt request generation enabled
15 CYS_IE	**Cycle Start Interrupt Enable** This bit controls CYC_IF interrupt request generation. 0 interrupt request generation disabled 1 interrupt request generation enabled

46.5.2.16 Protocol Interrupt Enable Register 1 (FR_PIER1)

Offset: 0x001E

Access: Read/Write
Write: Anytime

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	EMC_IE	IPC_IE	PECF_IE	PSC_IE	SSI3_IE	SSI2_IE	SSI1_IE	SSI0_IE	0	0	EVT_IE	ODT_IE	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 821. Protocol Interrupt Enable Register 1 (FR_PIER1)

This register defines whether or not the individual interrupt flags defined in Protocol Interrupt Flag Register 1 (FR_PIFR1) can generate a protocol interrupt request.

Table 815. FR_PIER1 field descriptions

Field	Description
0 EMC_IE	**Error Mode Changed Interrupt Enable** This bit controls EMC_IF interrupt request generation. 0 interrupt request generation disabled 1 interrupt request generation enabled
1 IPC_IE	**Illegal Protocol Control Command Interrupt Enable** This bit controls IPC_IF interrupt request generation. 0 interrupt request generation disabled 1 interrupt request generation enabled
2 PECF_IE	**Protocol Engine Communication Failure Interrupt Enable** This bit controls PECF_IF interrupt request generation. 0 interrupt request generation disabled 1 interrupt request generation enabled
3 PSC_IE	**Protocol State Changed Interrupt Enable** This bit controls PSC_IF interrupt request generation. 0 interrupt request generation disabled 1 interrupt request generation enabled
4 SSI3_IE	**Slot Status Counter Incremented Interrupt Enable** This bit controls SSI3_IF interrupt request generation. 0 interrupt request generation disabled 1 interrupt request generation enabled
5 SSI2_IE	**Slot Status Counter Incremented Interrupt Enable** This bit controls SSI2_IF interrupt request generation. 0 interrupt request generation disabled 1 interrupt request generation enabled
6 SSI1_IE	**Slot Status Counter Incremented Interrupt Enable** This bit controls SSI1_IF interrupt request generation. 0 interrupt request generation disabled 1 interrupt request generation enabled

Table 815. FR_PIER1 field descriptions (continued)

Field	Description
7 SSI0_IE	**Slot Status Counter Incremented Interrupt Enable** This bit controls SSI0_IF interrupt request generation. 0 interrupt request generation disabled 1 interrupt request generation enabled
10 EVT_IE	**Even Cycle Table Written Interrupt Enable** This bit controls EVT_IF interrupt request generation. 0 interrupt request generation disabled 1 interrupt request generation enabled
11 ODT_IE	**Odd Cycle Table Written Interrupt Enable** This bit controls ODT_IF interrupt request generation. 0 interrupt request generation disabled 1 interrupt request generation enabled

46.5.2.17 CHI Error Flag Register (FR_CHIERFR)

Offset: 0x0020

Access: Read/Write(1)
Write: Normal mode

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	FRLB_EF	FRLA_EF	PCMI_EF	FOVB_EF	FOVA_EF	MBS_EF	MBU_EF	LCK_EF	0	SBCF_EF	FID_EF	DPL_EF	SPL_EF	NML_EF	NMF_EF	ILSA_EF
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c		w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. In Functional Test Mode writing 1 toggles the interrupt flags.

Figure 822. CHI Error Flag Register (FR_CHIERFR)

This register holds the CHI-related error flags. The interrupt generation for each of these error flags is controlled by the CHI interrupt enable bit, CHIE, in the Global Interrupt Flag and Enable Register (FR_GIFER).

Table 816. FR_CHIERFR field descriptions

Field	Description
0 FRLB_EF	**Frame Lost Channel B Error Flag** This flag is set if a complete frame was received on channel B but it could not be stored in the selected individual message buffer because the message buffer is currently locked by the application. In this case, the frame and the related slot status information are lost. 0 No such event 1 Frame lost on channel B detected
1 FRLA_EF	**Frame Lost Channel A Error Flag** This flag is set if a complete frame was received on channel A but it could not be stored in the selected individual message buffer because the message buffer is currently locked by the application. In this case, the frame and the related slot status information are lost. 0 No such error 1 Frame lost on channel A detected

Table 816. FR_CHIERFR field descriptions (continued)

Field	Description
2 PCMI_EF	**Protocol Command Ignored Error Flag** This flag is set if the application has issued a POC command by writing to the POCCMD field in the Protocol Operation Control Register (FR_POCR) while the BSY flag is equal to 1. In this case, the command is ignored by the CC and is lost. 0 No such error 1 POC command ignored
3 FOVB_EF	**Receive FIFO Overrun Channel B Error Flag** This flag is set when an overrun of the FIFO for channel B has occurred. This error occurs if a semantically valid frame was received on channel B and matches all criteria to be appended to the FIFO for channel B but the FIFO is full. In this case, the received frame and its related slot status information is lost. 0 No such error 1 FIFO overrun on channel B has been detected
4 FOVA_EF	**Receive FIFO Overrun Channel A Error Flag** This flag is set when an overrun of the FIFO for channel A has occurred. This error occurs if a semantically valid frame was received on channel A and matches the all criteria to be appended to the FIFO for channel A but the FIFO is full. In this case, the received frame and its related slot status information is lost. 0 No such error 1 FIFO overrun on channel B has been detected
5 MBS_EF	**Message Buffer Search Error Flag** This flag is set if at least one of the following events occurs: a) The message buffer search engine is still running while the next search must be started due to the FlexRay protocol timing. b) A message buffer index greater than 67 is detected in the FR_MBIDXR[MBIDX] field of a found message buffer or in one of the FR_RSBIR[RSBIDX] fields. Refer to *Section 46.6.7.4: Message buffer search error* for details. 0 No such event 1 Search engine active while search start appears or illegal message buffer index detected
6 MBU_EF	**Message Buffer Utilization Error Flag** This flag is asserted if the application writes to a message buffer control field that is beyond the number of utilized message buffers programmed in the Message Buffer Segment Size and Utilization Register (FR_MBSSUTR). If the application writes to a FR_MBCCSRn register with n > LAST_MB_UTIL, the CC ignores the write attempt and asserts the message buffer utilization error flag MBU_EF in the CHI Error Flag Register (FR_CHIERFR). 0 No such event 1 Non-utilized message buffer enabled
7 LCK_EF	**Lock Error Flag** This flag is set if the application tries to lock a message buffer that is already locked by the CC due to internal operations. In that case, the CC does not grant the lock to the application. The application must issue the lock request again. 0 No such error 1 Lock error detected

Table 816. FR_CHIERFR field descriptions (continued)

Field	Description
9 SBCF_EF	**System Bus Communication Failure Error Flag** This flag is set if a system bus access was not finished within the required amount of time (see *Section 46.6.19.1.2: System bus access timeout*). 0 No such event 1 System bus access not finished in time
10 FID_EF	**Frame ID Error Flag** This flag is set if the frame ID stored in the message buffer header area differs from the frame ID stored in the message buffer control register. 0 No such error occurred 1 Frame ID error occurred
11 DPL_EF	**Dynamic Payload Length Error Flag** This flag is set if the payload length written into the message buffer header field of a transmit message buffer assigned to the dynamic segment is greater than the maximum payload length for the dynamic segment as it is configured in the corresponding max_payload_length_dynamic field in the Protocol Configuration Register 24 (FR_PCR24). 0 No such error occurred 1 Dynamic payload length error occurred
12 SPL_EF	**Static Payload Length Error Flag** This flag is set if the payload length written into the message buffer header field of a transmit message buffer assigned to the static segment is different from the payload length for the static segment as it is configured in the corresponding payload_length_static field in the Protocol Configuration Register 19 (FR_PCR19). 0 No such error occurred 1 Static payload length error occurred
13 NML_EF	**Network Management Length Error Flag** This flag is set if the payload length written into the header structure of a receive message buffer assigned to the static segment is less than the configured length of the Network Management Vector as configured in the Network Management Vector Length Register (FR_NMVLR). In this case, the received part of the Network Management Vector will be used to update the Network Management Vector. 0 No such error occurred 1 Network management length error occurred
14 NMF_EF	**Network Management Frame Error Flag** This flag is set if a received message in the static segment with a Preamble Indicator flag (PP) asserted has its Null Frame indicator flag (NF) asserted as well. In this case, the Global Network Management Registers (see *Section 46.5.2.39: Network Management Vector Registers (FR_NMVRn)*) are not updated. 0 No such error occurred 1 Network management frame error occurred
15 ILSA_EF	**Illegal System Bus Address Error Flag** This flag is set if the external system bus subsystem has detected an access to an illegal system bus address from the CC (see *Section 46.6.19.1.1: System bus illegal address access*). 0 No such event 1 Illegal system bus address accessed

46.5.2.18 Message Buffer Interrupt Vector Register (FR_MBIVEC)

Offset: 0x0022 Access: Read-only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	TBIVEC						0	0	RBIVEC					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 823. Message Buffer Interrupt Vector Register (FR_MBIVEC)

This register indicates the lowest numbered receive message buffer and the lowest numbered transmit message buffer that have their MBIF interrupt status flag and MBIE interrupt enable bits asserted. This means that message buffers with lower message buffer numbers have higher priority.

Table 817. FR_MBIVEC field descriptions

Field	Description
2:7 TBIVEC	**Transmit Buffer Interrupt Vector** This field provides the number of the lowest numbered enabled transmit message buffer that has its MBIF interrupt status flag and its MBIE interrupt enable bit set. If there is no transmit message buffer with the MBIF interrupt status flag and the MBIE interrupt enable bits asserted, the value in this field is set to 0.
10:15 RBIVEC	**Receive Buffer Interrupt Vector** This field provides the message buffer number of the lowest numbered receive message buffer that has its MBIF interrupt flag and its MBIE interrupt enable bit asserted. If there is no receive message buffer with the MBIF interrupt status flag and the MBIE interrupt enable bits asserted, the value in this field is set to 0.

46.5.2.19 Channel A Status Error Counter Register (FR_CASERCR)

Offset: 0x0024 Access: Read-only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	CHAERSCNT															
W																
Reset(1)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. **A**dditional Reset: RUN Command

Figure 824. Channel A Status Error Counter Register (FR_CASERCR)

This register provides the channel status error counter for channel A. The protocol engine generates a slot status vector for each static slot, each dynamic slot, the symbol window, and the NIT. The slot status vector contains the four protocol related error indicator bits *vSS!SyntaxError, vSS!ContentError, vSS!BViolation*, and *vSS!TxConflict*. The CC increments the status error counter by 1 if, for a slot or segment, at least one error indicator bit is set to 1. The counter wraps around after it has reached the maximum value. For more information on slot status monitoring, see *Section 46.6.18: Slot status monitoring*.

Table 818. FR_CASERCR Field Descriptions

Field	Description
0:15 CHAERSCNT	**Channel Status Error Counter** This field provides the current value channel status error counter. The counter value is updated within the first macrotick of the following slot or segment.

46.5.2.20 Channel B Status Error Counter Register (FR_CBSERCR)

Offset: 0x0026 Access: Read-only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	CHBERSCNT															
W																
Reset(1)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. **A**dditional Reset: RUN Command

Figure 825. Channel B Status Error Counter Register (FR_CBSERCR)

This register provides the channel status error counter for channel B. The protocol engine generates a slot status vector for each static slot, each dynamic slot, the symbol window, and the network idle time (NIT). The slot status vector contains the four protocol-related error indicator bits *vSS!SyntaxError*, *vSS!ContentError*, *vSS!BViolation*, and *vSS!TxConflict*. The CC increments the status error counter by 1 if, for a slot or segment, at least one error indicator bit is set to 1. The counter wraps around after it has reached the maximum value. For more information on slot status monitoring, see *Section 46.6.18: Slot status monitoring*.

Table 819. FR_CBSERCR field descriptions

Field	Description
0:15 CHBERSCNT	**Channel Status Error Counter** This field provides the current channel status error count. The counter value is updated within the first macrotick of the following slot or segment.

46.5.2.21 Protocol Status Register 0 (FR_PSR0)

Offset: 0x0028 Access: Read-only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	ERRMODE		SLOTMODE		0	PROTSTATE			STARTUPSTATE				0	WAKEUPSTATUS		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 826. .Protocol Status Register 0 (FR_PSR0)

This register provides information about the current protocol status.

Table 820. FR_PSR0 field descriptions

Field	Description
0:1 ERRMODE	**Error Mode** protocol-related variable: *vPOC!ErrorMode*. This field indicates the error mode of the protocol. 00 ACTIVE 01 PASSIVE 10 COMM_HALT 11 Reserved
2:3 SLOTMODE	**Slot Mode** protocol-related variable: *vPOC!SlotMode*. This field indicates the slot mode of the protocol. 00 SINGLE 01 ALL_PENDING 10 ALL 11 Reserved
5:7 PROTSTATE	**Protocol State** protocol-related variable: *vPOC!State.* This field indicates the state of the protocol. 000 *POC:default config* 001 *POC:config* 010 *POC:wakeup* 011 *POC:ready* 100 *POC:normal passive* 101 *POC:normal active* 110 *POC:halt* 111 *POC:startup*

Table 820. FR_PSR0 field descriptions (continued)

Field	Description
8:11 STARTUP STATE	**Startup State** protocol-related variable: *vPOC!StartupState*. This field indicates the current sub-state of the startup procedure. 0000 Reserved 0001 Reserved 0010 *POC:coldstart collision resolution* 0011 *POC:coldstart listen* 0100 *POC:integration consistency check* 0101 *POC:integration listen* 0110 Reserved 0111 *POC:initialize schedule* 1000 Reserved 1001 Reserved 1010 *POC:coldstart consistency check* 1011 Reserved 1100 Reserved 1101 *POC:integration coldstart check* 1110 *POC:coldstart gap* 1111 *POC:coldstart join*
13:15 WAKEUP STATUS	**Wakeup Status** protocol-related variable: *vPOC!WakeupStatus*. This field provides the outcome of the execution of the wakeup mechanism. 000 UNDEFINED 001 RECEIVED_HEADER 010 RECEIVED_WUP 011 COLLISION_HEADER 100 COLLISION_WUP 101 COLLISION_UNKNOWN 110 TRANSMITTED 111 Reserved

46.5.2.22 Protocol Status Register 1 (FR_PSR1)

Offset: 0x002A

Access: Read/Write
Write: Normal mode

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	CSAA	CSP	0	REMCSAT					CPN	HHR	FRZ	APTAC				
W	w1c															
Reset(1)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. Additional Reset: CSAA, CSP, CPN: RUN Command

Figure 827. Protocol Status Register 1 (FR_PSR1)

Table 821. FR_PSR1 field descriptions

Field	Description
0 CSAA	**Cold Start Attempt Aborted Flag** protocol-related event: 'set coldstart abort indicator in CHI.' This flag is set when the CC has aborted a cold start attempt. 0 No such event 1 Cold start attempt aborted
1 CSP	**Leading Cold Start Path** This status bit is set when the CC has reached the *POC:normal active* state via the leading cold start path. This indicates that this node has started the network 0 No such event 1 *POC:normal active* reached from *POC:startup* state via leading cold start path
3:7 REMCSAT	**Remaining Coldstart Attempts** protocol-related variable: *vRemainingColdstartAttempts.* This field provides the number of remaining cold start attempts that the CC will execute.
8 CPN	**Leading Cold Start Path Noise**—protocol-related variable: *vPOC!ColdstartNoise.* This status bit is set if the CC has reached the *POC:normal active* state via the leading cold start path under noise conditions. This indicates there was some activity on the FlexRay bus while the CC was starting up the cluster. 0 No such event 1 *POC:normal active* state was reached from *POC:startup* state via noisy leading cold start path
9 HHR	**Host Halt Request Pending** protocol related variable: *vPOC!CHIHaltRequest.* This status bit is set when CC receives the HALT command from the application via the Protocol Operation Control Register (FR_POCR). The CC clears this status bit after a hard reset condition or when the protocol is in the *POC:default config* state. 0 No such event 1 HALT command received
10 FRZ	**Freeze Occurred** protocol related variable: *vPOC!Freeze.* This status bit is set when the CC has reached the *POC:halt* state due to the host FREEZE command or due to an internal error condition requiring immediate halt. The CC clears this status bit after a hard reset condition or when the protocol is in the *POC:default config* state. 0 No such event 1 Immediate halt due to FREEZE or internal error condition
11:15 APTAC	**Allow Passive to Active Counter** protocol-related variable: *vPOC!vAllowPassivetoActive.* This field provides the number of consecutive even/odd communication cycle pairs that have passed with valid rate and offset correction terms, but the protocol is still in the *POC:normal passive* state due to an application-configured delay to enter *POC:normal active* state. This delay is defined by the allow_passive_to_active field in Protocol Configuration Register 12 (FR_PCR12).

46.5.2.23 Protocol Status Register 2 (FR_PSR2)

Offset: 0x002C Access: Read-only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	NBVB	NSEB	STCB	SBVB	SSEB	MTB	NBVA	NSEA	STCA	SBVA	SSEA	MTA	CKCORFCNT			
W																
Reset(1)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. Additional Reset: RUN Command

Figure 828. Protocol Status Register 2 (FR_PSR2)

This register provides a snapshot of status information about the network idle time (NIT), the symbol window, and the clock synchronization. The NIT-related status bits NBVB, NSEB, NBVA, and NSEA are updated by the CC after the end of the NIT and before the end of the first slot of the next communication cycle. The symbol-window-related status bits STCB, SBVB, SSEB, MTB, STCA, SBVA, SSEB, and MTA are updated by the CC after the end of the symbol window and before the end of the current communication cycle. If no symbol window is configured, the symbol-window-related status bits remain in their reset state. The clock synchronization-related CKCORFCNT is updated by the CC after the end of the static segment and before the end of the current communication cycle.

Table 822. FR_PSR2 field descriptions

Field	Description
0 NBVB	**NIT Boundary Violation on Channel B** protocol-related variable: *vSS!BViolation* for NIT on channel B. This status bit is set when there was some media activity on the FlexRay bus channel B at the end of the NIT. 0 No such event 1 Media activity at boundaries detected
1 NSEB	**NIT Syntax Error on Channel B** protocol-related variable: *vSS!SyntaxError* for NIT on channel B. This status bit is set when a syntax error was detected during NIT on channel B. 0 No such event 1 Syntax error detected
2 STCB	**Symbol Window Transmit Conflict on Channel B** protocol-related variable: *vSS!TxConflict* for symbol window on channel B. This status bit is set if there was a transmission conflict during the symbol window on channel B. 0 No such event 1 Transmission conflict detected
3 SBVB	**Symbol Window Boundary Violation on Channel B** protocol-related variable: *vSS!BViolation* for symbol window on channel B. This status bit is set if there was some media activity on the FlexRay bus channel B at the start or at the end of the symbol window. 0 No such event 1 Media activity at boundaries detected

Table 822. FR_PSR2 field descriptions (continued)

Field	Description
4 SSEB	**Symbol Window Syntax Error on Channel B** protocol related variable: *vSS!SyntaxError* for symbol window on channel B. This status bit is set when a syntax error was detected during the symbol window on channel B. 0 No such event 1 Syntax error detected
5 MTB	**Media Access Test Symbol MTS Received on Channel B** protocol-related variable: *vSS!ValidMTS* for Symbol Window on channel B. This status bit is set if the Media Access Test Symbol MTS was received in the symbol window on channel B. 0 No such event 1 MTS symbol received
6 NBVA	**NIT Boundary Violation on Channel A** protocol-related variable: *vSS!BViolation* for NIT on channel A. This status bit is set when there was some media activity on the FlexRay bus channel A at the end of the NIT. 0 No such event 1 Media activity at boundaries detected
7 NSEA	**NIT Syntax Error on Channel A** protocol-related variable: *vSS!SyntaxError* for NIT on channel A This status bit is set when a syntax error was detected during NIT on channel A. 0 No such event 1 Syntax error detected
8 STCA	**Symbol Window Transmit Conflict on Channel A** protocol related variable: *vSS!TxConflict* for symbol window on channel A. This status bit is set if there was a transmission conflicts during the symbol window on channel A. 0 No such event 1 Transmission conflict detected
9 SBVA	**Symbol Window Boundary Violation on Channel A** protocol-related variable: *vSS!BViolation* for symbol window on channel A. This status bit is set if there was some media activity on the FlexRay bus channel A at the start or at the end of the symbol window. 0 No such event 1 Media activity at boundaries detected
10 SSEA	**Symbol Window Syntax Error on Channel A** protocol-related variable: *vSS!SyntaxError* for symbol window on channel A. This status bit is set when a syntax error was detected during the symbol window on channel A. 0 No such event 1 Syntax error detected

Table 822. FR_PSR2 field descriptions (continued)

Field	Description
11 MTA	**Media Access Test Symbol MTS Received on Channel A** protocol-related variable: *vSS!ValidMTS* for symbol window on channel A. This status bit is set if the Media Access Test Symbol (MTS) was received in the symbol window on channel A. 0 No such event 1 MTS symbol received
12:15 CKCORFCNT	**Clock Correction Failed Counter** protocol -elated variable: *vClockCorrectionFailed.* This field provides the number of consecutive even/odd communication cycle pairs that have passed without a clock synchronization offset or rate correction due to lack of synchronization frames. It is not incremented when it has reached the configured value of either max_without_clock_correction_fatal or max_without_clock_correction_passive as defined in Protocol Configuration Register 8 (FR_PCR8). The CC resets this counter on a hard reset condition, when the protocol enters the *POC:normal active* state, or when both the rate and offset correction terms have been calculated successfully.

46.5.2.24 Protocol Status Register 3 (FR_PSR3)

Offset: 0x002E

Access: Read/Write
Write: Normal mode

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	WUB	ABVB	AACB	ACEB	ASEB	AVFB	0	0	WUA	ABVA	AACA	ACEA	ASEA	AVFA
W			w1c	w1c	w1c	w1c	w1c	w1c			w1c	w1c	w1c	w1c	w1c	w1c
Reset(1)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. Additional Reset: RUN Command

Figure 829. Protocol Status Register 3 (FR_PSR3)

This register provides aggregated channel status information as an accrued status of channel activity for all communication slots, regardless of whether they are assigned for transmission or subscribed for reception. It provides accrued information for the symbol window, the NIT, and the wakeup status.

Table 823. FR_PSR3 field descriptions

Field	Description
2 WUB	**Wakeup Symbol Received on Channel B** This flag is set when a wakeup symbol was received on channel B. 0 No wakeup symbol received 1 Wakeup symbol received
3 ABVB	**Aggregated Boundary Violation on Channel B** This flag is set when a boundary violation was detected on channel B. Boundary violations are detected in the communication slots, the symbol window, and the NIT. 0 No boundary violation detected 1 Boundary violation detected

Table 823. FR_PSR3 field descriptions (continued)

Field	Description
4 AACB	**Aggregated Additional Communication on Channel B** This flag is set when at least one valid frame was received on channel B in a slot that also contained an additional communication with either syntax error, content error, or boundary violations. 0 No additional communication detected 1 Additional communication detected
5 ACEB	**Aggregated Content Error on Channel B** This flag is set when a content error was detected on channel B. Content errors are detected in the communication slots, the symbol window, and the NIT. 0 No content error detected 1 Content error detected
6 ASEB	**Aggregated Syntax Error on Channel B** This flag is set when a syntax error was detected on channel B. Syntax errors are detected in the communication slots, the symbol window, and the NIT. 0 No syntax error detected 1 Syntax errors detected
7 AVFB	**Aggregated Valid Frame on Channel B** This flag is set when a syntactically correct valid frame has been received in any static or dynamic slot through channel B. 0 No syntactically valid frames received 1 At least one syntactically valid frame received
10 WUA	**Wakeup Symbol Received on Channel A** This flag is set when a wakeup symbol was received on channel A. 0 No wakeup symbol received 1 Wakeup symbol received
11 ABVA	**Aggregated Boundary Violation on Channel A** This flag is set when a boundary violation was detected on channel A. Boundary violations are detected in the communication slots, the symbol window, and the NIT. 0 No boundary violation detected 1 Boundary violation detected
12 AACA	**Aggregated Additional Communication on Channel A** This flag is set when a valid frame was received in a slot on channel A that also contained an additional communication with either syntax error, content error, or boundary violations. 0 No additional communication detected 1 Additional communication detected
13 ACEA	**Aggregated Content Error on Channel A** This flag is set when a content error has been detected on channel A. Content errors are detected in the communication slots, the symbol window, and the NIT. 0 No content error detected 1 Content error detected

Table 823. FR_PSR3 field descriptions (continued)

Field	Description
14 ASEA	**Aggregated Syntax Error on Channel A** This flag is set when a syntax error has been detected on channel A. Syntax errors are detected in the communication slots, the symbol window, and the NIT. 0 No syntax error detected 1 Syntax errors detected
15 AVFA	**Aggregated Valid Frame on Channel A** This flag is set when a syntactically correct valid frame has been received in any static or dynamic slot through channel A. 0 No syntactically valid frames received 1 At least one syntactically valid frame received

46.5.2.25 Macrotick Counter Register (FR_MTCTR)

Offset: 0x0030 Access: Read-only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	MTCT													
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 830. Macrotick Counter Register (FR_MTCTR)

This register provides the macrotick count of the current communication cycle.

Table 824. FR_MTCTR field descriptions

Field	Description
2:15 MTCT	**Macrotick Counter** protocol-related variable: *vMacrotick.* This field provides the macrotick count of the current communication cycle.

46.5.2.26 Cycle Counter Register (FR_CYCTR)

Offset: 0x0032 Access: Read-only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	CYCCNT					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 831. Cycle Counter Register (FR_CYCTR)

This register provides the number of the current communication cycle.

Table 825. FR_CYCTR field descriptions

Field	Description
10:15 CYCCNT	**Cycle Counter** protocol-related variable: *vCycleCounter.* This field provides the number of the current communication cycle. If the counter reaches the maximum value of 63, the counter wraps and starts from 0 again.

46.5.2.27 Slot Counter Channel A Register (FR_SLTCTAR)

Offset: 0x0034 Access: Read-only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	SLOTCNTA										
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 832. Slot Counter Channel A Register (FR_SLTCTAR)

This register provides the number of the current slot in the current communication cycle for channel A.

Table 826. FR_SLTCTAR field descriptions

Field	Description
5:15 SLOTCNTA	**Slot Counter Value for Channel A** protocol-related variable: *vSlotCounter* for channel A. This field provides the number of the current slot in the current communication cycle.

46.5.2.28 Slot Counter Channel B Register (FR_SLTCTBR)

Offset: 0x0036 Access: Read-only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	SLOTCNTB										
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 833. Slot Counter Channel B Register (FR_SLTCTBR)

This register provides the number of the current slot in the current communication cycle for channel B.

Table 827. FR_SLTCTBR field descriptions

Field	Description
5:15 SLOTCNTB	**Slot Counter Value for Channel B** protocol-related variable: *vSlotCounter* for channel B. This field provides the number of the current slot in the current communication cycle.

46.5.2.29 Rate Correction Value Register (FR_RTCORVR)

Offset: 0x0038 Access: Read-only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	RATECORR															
W																
Reset(1)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. Additional Reset: RUN Command

Figure 834. Rate Correction Value Register (FR_RTCORVR)

This register provides the sign extended rate correction value in microticks as it was calculated by the clock synchronization algorithm. The CC updates this register during the NIT of each odd numbered communication cycle.

Table 828. FR_RTCORVR Field Descriptions

Field	Description
0:15 RATECORR	**Rate Correction Value** protocol-related variable: *vRateCorrection* (before value limitation and external rate correction). This field provides the sign extended rate correction value in microticks as it was calculated by the clock synchronization algorithm. The value is represented in 2's complement format. This value does not include the value limitation and the application of the external rate correction. If the magnitude of the internally calculated rate correction value exceeds the limit given by rate_correction_out in the Protocol Configuration Register 13 (FR_PCR13), the clock correction reached limit interrupt flag (CCL_IF) is set in the Protocol Interrupt Flag Register 0 (FR_PIFR0). **Note:** If the CC was not able to calculate a new rate correction term due to a lack of synchronization frames, the RATECORR value is not updated.

46.5.2.30 Offset Correction Value Register (FR_OFCORVR)

Offset: 0x003A Access: Read-only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	OFFSETCORR															
W																
Reset(1)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. Additional Reset: RUN Command

Figure 835. Offset Correction Value Register (FR_OFCORVR)

This register provides the sign extended offset correction value in microticks as it was calculated by the clock synchronization algorithm. The CC updates this register during the NIT.

Table 829. FR_OFCORVR Field Descriptions

Field	Description
0:15 OFFSETCORR	**Offset Correction Value** protocol-related variable: *vOffsetCorrection* (before value limitation and external offset correction). This field provides the sign extended offset correction value in microticks as it was calculated by the clock synchronization algorithm. The value is represented in 2's complement format. This value does not include the value limitation and the application of the external offset correction. If the magnitude of the internally calculated rate correction value exceeds the limit given by offset_correction_out field in the Protocol Configuration Register 29 (FR_PCR29), the clock correction reached limit interrupt flag CCL_IF is set in the Protocol Interrupt Flag Register 0 (FR_PIFR0). **Note:** If the CC was not able to calculate an new offset correction term due to a lack of synchronization frames, the OFFSETCORR value is not updated.

46.5.2.31 Combined Interrupt Flag Register (FR_CIFR)

Offset: 0x003C Access: Read-only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	MIF	PRIF	CHIF	WUPIF	FAFBIF	FAFAIF	RBIF	TBIF
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 836. Combined Interrupt Flag Register (FR_CIFR)

This register provides five combined interrupt flags and a copy of three individual interrupt flags. The combined interrupt flags are the result of a binary OR of the values of other interrupt flags regardless of the state of the interrupt enable bits. The generation scheme for the combined interrupt flags is depicted in *Figure 967*. The individual interrupt flags WUPIF, FAFBIF, and FAFAIF are copies of corresponding flags in the Global Interrupt Flag and Enable Register (FR_GIFER) and are provided here to simplify the application interrupt flag check. To clear the individual interrupt flags, the application must use the Global Interrupt Flag and Enable Register (FR_GIFER).

Note: The meanings of the combined status bits MIF, PRIF, CHIF, RBIF, and TBIF are different from those mentioned in the Global Interrupt Flag and Enable Register (FR_GIFER).

Table 830. FR_CIFR field descriptions

Field	Description
8 MIF	**Module Interrupt Flag** This flag is set if there is at least one interrupt source that has its interrupt flag asserted. 0 No interrupt source has its interrupt flag asserted 1 At least one interrupt source has its interrupt flag asserted
9 PRIF	**Protocol Interrupt Flag** This flag is set if at least one of the individual protocol interrupt flags in the Protocol Interrupt Flag Register 0 (FR_PIFR0) or Protocol Interrupt Flag Register 1 (FR_PIFR1) is equal to 1. 0 All individual protocol interrupt flags are equal to 0 1 At least one of the individual protocol interrupt flags is equal to 1
10 CHIF	**CHI Interrupt Flag** This flag is set if at least one of the individual CHI error flags in the CHI Error Flag Register (FR_CHIERFR) is equal to 1. 0 All CHI error flags are equal to 0 1 At least one CHI error flag is equal to 1
11 WUPIF	**Wakeup Interrupt Flag** Provides the same value as FR_GIFER[WUPIF].
12 FAFBIF	**Receive FIFO Channel B Almost Full Interrupt Flag** Provides the same value as FR_GIFER[FAFBIF].
13 FAFAIF	**Receive FIFO Channel A Almost Full Interrupt Flag** Provides the same value as FR_GIFER[FAFAIF].
14 RBIF	**Receive Message Buffer Interrupt Flag** This flag is set if for at least one of the individual receive message buffers (FR_MBCCSRn[MTD] = 0) the interrupt flag MBIF in the corresponding Message Buffer Configuration, Control, Status Registers (FR_MBCCSRn) is equal to 1. 0 None of the individual receive message buffers has the MBIF flag asserted 1 At least one individual receive message buffers has the MBIF flag asserted
15 TBIF	**Transmit Message Buffer Interrupt Flag** This flag is set if for at least one of the individual transmit message buffers (FR_MBCCSRn[MTD] = 1) the interrupt flag MBIF in the corresponding Message Buffer Configuration, Control, Status Registers (FR_MBCCSRn) is equal to 1. 0 None of the individual transmit message buffers has the MBIF flag asserted 1 At least one individual transmit message buffers has the MBIF flag asserted

46.5.2.32 System Memory Access Timeout Register (FR_SYMATOR)

Offset: 0x003E

Access: Read/Write
Write: Disabled mode

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	TIMEOUT							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0

Figure 837. System Memory Access Time-Out Register (FR_SYMATOR)

Table 831. FR_SYMATOR field descriptions

Field	Description
8:15 TIMEOUT	**System Memory Access Timeout** This value defines when a system bus access timeout is detected. For a detailed description see *Section 46.7.1.1: Configure System Memory Access Timeout Register (FR_SYMATOR)* and *Section 46.6.19.1.2: System bus access timeout*.

46.5.2.33 Sync Frame Counter Register (FR_SFCNTR)

Offset: 0x0040 Access: Read-only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	SFEVB				SFEVA				SFODB				SFODA			
W																
Reset(1)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. Additional Reset: RUN Command

Figure 838. Sync Frame Counter Register (FR_SFCNTR)

This register provides the number of synchronization frames that are used for clock synchronization in the last even and in the last odd numbered communication cycle. This register is updated after the start of the NIT and before 10 macroticks after offset correction start.

Note: If the application has locked the even synchronization table at the end of the static segment of an even communication cycle, the CC will not update the fields SFEVB and SFEVA.

If the application has locked the odd synchronization table at the end of the static segment of an odd communication cycle, the CC will not update the SFODB and SFODA values.

Table 832. FR_SFCNTR field descriptions

Field	Description
0:3 SFEVB	Sync Frames Channel B, even cycle protocol-related variable: size of (*vsSyncIdListB* for even cycle). This field provides the size of the internal list of frame IDs of received synchronization frames used for clock synchronization.
4:7 SFEVA	Sync Frames Channel A, even cycle protocol related variable: size of (*vsSyncIdListA* for even cycle). This field provides the size of the internal list of frame IDs of received synchronization frames used for clock synchronization.
8:11 SFODB	Sync Frames Channel B, odd cycle protocol-related variable: size of (*vsSyncIdListB* for odd cycle). This field provides the size of the internal list of frame IDs of received synchronization frames used for clock synchronization.
12:15 SFODA	Sync Frames Channel A, odd cycle protocol-related variable: size of (*vsSyncIdListA* for odd cycle). This field provides the size of the internal list of frame IDs of received synchronization frames used for clock synchronization.

46.5.2.34 Sync Frame Table Offset Register (FR_SFTOR)

Offset: 0x0042

Access: Read/Write
Write: *POC:config*

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	SFT_OFFSET															0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 839. Sync Frame Table Offset Register (FR_SFTOR)

This register defines the FlexRay memory area related offset for sync frame tables. For more details, see *Section 46.6.12: Sync frame ID and sync frame deviation tables*.

Table 833. FR_SFTOR field descriptions

Field	Description
0:14 SFT_OFFSET	**Sync Frame Table Offset** The offset of the sync frame tables in the FlexRay memory area. This offset is required to be 16-bit aligned. Thus, STF_OFFSET[0] is always 0.

46.5.2.35 Sync Frame Table Configuration, Control, Status Register (FR_SFTCCSR)

Offset: 0x0044

Access: Read/Write
Write: Normal mode

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	CYCNUM						ELKS	OLKS	EVAL	OVAL	0	0	SDVEN	SIDEN
W	ELKT	OLKT												OPT		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 840. Sync Frame Table Configuration, Control, Status Register (FR_SFTCCSR)

This register provides configuration, control, and status information related to the generation and access of the clock sync ID tables and clock sync measurement tables. For a detailed description, see *Section 46.6.12: Sync frame ID and sync frame deviation tables*.

Table 834. FR_SFTCCSR field descriptions

Field	Description
0 ELKT	**Even Cycle Tables Lock/Unlock Trigger** This trigger bit is used to lock and unlock the even cycle tables. 0 No effect 1 Triggers lock/unlock of the even cycle tables
1 OLKT	**Odd Cycle Tables Lock/Unlock Trigger** This trigger bit is used to lock and unlock the odd cycle tables. 0 No effect 1 Triggers lock/unlock of the odd cycle tables

Table 834. FR_SFTCCSR field descriptions (continued)

Field	Description
2:7 CYCNUM	**Cycle Number** This field provides the number of the cycle in which the currently locked table was recorded. If none or both tables are locked, this value is related to the even cycle table.
8 ELKS	**Even Cycle Tables Lock Status** This status bit indicates whether the application has locked the even cycle tables. 0 Application has not locked the even cycle tables 1 Application has locked the even cycle tables
9 OLKS	**Odd Cycle Tables Lock Status** This status bit indicates whether the application has locked the odd cycle tables. 0 Application has not locked the odd cycle tables 1 Application has locked the odd cycle tables
10 EVAL	**Even Cycle Tables Valid** This status bit indicates whether the Sync Frame ID and Sync Frame Deviation Tables for the even cycle are valid. The CC clears this status bit when it starts updating the tables, and sets this bit when it has finished the table update. 0 Tables are not valid (update is ongoing) 1 Tables are valid (consistent)
11 OVAL	**Odd Cycle Tables Valid** This status bit indicates whether the Sync Frame ID and Sync Frame Deviation Tables for the odd cycle are valid. The CC clears this status bit when it starts updating the tables, and sets this bit when it has finished the table update. 0 Tables are not valid (update is ongoing) 1 Tables are valid (consistent)
13 OPT	**One Pair Trigger** This trigger bit controls whether the CC writes continuously or only one pair of Sync Frame Tables into the FlexRay memory area. If this trigger is set to 1 while SDVEN or SIDEN is set to 1, the CC writes only one pair of the enabled Sync Frame Tables corresponding to the next even-odd-cycle pair into the FlexRay memory area. In this case, the CC clears the SDVEN or SIDEN bits immediately. If this trigger is set to 0 while SDVEN or SIDEN is set to 1, the CC writes continuously the enabled Sync Frame Tables into the FlexRay memory area. 0 Write continuously pairs of enabled Sync Frame Tables into FlexRay memory area. 1 Write only one pair of enabled Sync Frame Tables into FlexRay memory area.

Table 834. FR_SFTCCSR field descriptions (continued)

Field	Description
14 SDVEN	**Sync Frame Deviation Table Enable** This bit controls the generation of the Sync Frame Deviation Tables. The application must set this bit to request the CC to write the Sync Frame Deviation Tables into the FlexRay memory area. 0 Do not write Sync Frame Deviation Tables 1 Write Sync Frame Deviation Tables into FlexRay memory area **Note:** If SDVEN is set to 1, then SIDEN must also be set to 1.
15 SIDEN	**Sync Frame ID Table Enable** This bit controls the generation of the Sync Frame ID Tables. The application must set this bit to 1 to request the CC to write the Sync Frame ID Tables into the FlexRay memory area. 0 Do not write Sync Frame ID Tables 1 Write Sync Frame ID Tables into FlexRay memory area

46.5.2.36 Sync Frame ID Rejection Filter Register (FR_SFIDRFR)

Offset: 0x0046

Access: Read/Write(1)
Write: Normal mode

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	SYNFRID									
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. 16-bit write access required

Figure 841. Sync Frame ID Rejection Filter Register (FR_SFIDRFR)

This register defines the Sync Frame Rejection Filter ID. The application must update this register outside of the static segment. If the application updates this register in the static segment, it can appear that the CC accepts the sync frame in the current cycle.

Table 835. FR_SFIDRFR field descriptions

Field	Description
6:15 SYNFRID	**Sync Frame Rejection ID** This field defines the frame ID of a frame that must not be used for clock synchronization. For details see *Section 46.6.15.2: Sync Frame Rejection Filtering*.

46.5.2.37 Sync Frame ID Acceptance Filter Value Register (FR_SFIDAFVR)

Offset: 0x0048

Access: Read/Write
Write: *POC:config*

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	FVAL									
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 842. Sync Frame ID Acceptance Filter Value Register (FR_SFIDAFVR)

This register defines the sync frame acceptance filter value. For details on filtering, see *Section 46.6.15: Sync frame filtering*.

Table 836. FR_SFIDAFVR field descriptions

Field	Description
6:15 FVAL	**Filter Value** This field defines the value for the sync frame acceptance filtering.

46.5.2.38 Sync Frame ID Acceptance Filter Mask Register (FR_SFIDAFMR)

Offset: 0x004A

Access: Read/Write
Write: *POC:config*

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	FMSK									
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 843. Sync Frame ID Acceptance Filter Mask Register (FR_SFIDAFMR)

This register defines the sync frame acceptance filter mask. For details on filtering see *Section 46.6.15.1: Sync frame acceptance filtering*.

Table 837. FR_SFIDAFMR field descriptions

Field	Description
6:15 FMSK	**Filter Mask** This field defines the mask for the sync frame acceptance filtering.

46.5.2.39 Network Management Vector Registers (FR_NMVR*n*)

Offset: 0x004C +*n**0x2 (*n*=0 to 5)

Access: Read-only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	NMVP															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 844. Network Management Vector Registers (FR_NMVR*n*)

Each of these six registers holds one part of the Network Management Vector. The length of the Network Management Vector is configured in the Network Management Vector Length Register (FR_NMVLR). If FR_NMVLR is programmed with a value that is less than 12 bytes, the remaining bytes of the Network Management Vector Registers (FR_NMVR0–FR_NMVR5), which are not used for the Network Management Vector accumulating, will remain 0.

The NMVR provides accrued information over all received NMVs in the last communication cycle. All NMVs received in one cycle are ORed into the NMVR. The NMVR is updated at the end of the communication cycle.

Table 838. NMVR*n* field descriptions

Field	Description
0:15 NMVP	**Network Management Vector Part** The mapping between the Network Management Vector Registers (FR_NMVR0–FR_NMVR5) and the receive message buffer payload bytes in NMV[0:11] is depicted in *Table 839*.

Table 839. Mapping of NMVRn to the Received Payload Bytes NMV*x*

NMVR*n* Register	NMV*x* Received Payload
FR_NMVR0.NMVP[0:7]	NMV0
FR_NMVR0.NMVP[8:15]	NMV1
FR_NMVR1.NMVP[0:7]	NMV2
FR_NMVR1.NMVP[[8:15]	NMV3
FR_NMVR2.NMVP[0:7]	NMV4
FR_NMVR2.NMVP[8:15]	NMV5
FR_NMVR3.NMVP[0:7]	NMV6
FR_NMVR3.NMVP[8:15]	NMV7
FR_NMVR4.NMVP[0:7]	NMV8
FR_NMVR4.NMVP[8:15]	NMV9
FR_NMVR5.NMVP[0:7]	NMV10
FR_NMVR5.NMVP[8:15]	NMV11

46.5.2.40 Network Management Vector Length Register (FR_NMVLR)

Offset: 0x0058

Access: Read/Write
Write: *POC:config*

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	NMVL			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 845. Network Management Vector Length Register (FR_NMVLR)

This register defines the length of the network management vector in bytes.

Table 840. FR_NMVLR field descriptions

Field	Description
12:15 NMVL	**Network Management Vector Length** Protocol-related variable: *gNetworkManagementVectorLength.* This field defines the length of the Network Management Vector in bytes. Legal values are between 0 and 12.

46.5.2.41 Timer Configuration and Control Register (FR_TICCR)

Access: Read/Write
Offset: 0x005A Write: T2_CFG: *POC:config*
T2_REP, T1_REP, T1SP, T2SP, T1TR, T2TR: Normal mode

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	T2_CFG	T2_REP	0	0	0	T2ST	0	0	0	T1_REP	0	0	0	T1ST
W						T2SP	T2TR							T1SP	T1TR	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 846. Timer Configuration and Control Register (FR_TICCR)

This register is used to configure and control the T1 and T2 timers. For timer details, see *Section 46.6.17: Timer support*. The T1 timer is an absolute timer. The T2 timer can be configured as an absolute or relative timer.

Table 841. FR_TICCR field descriptions

Field	Description
2 T2_CFG	**Timer T2 Configuration** This bit configures the timebase mode of timer T2. 0 T2 is absolute timer. 1 T2 is relative timer.
3 T2_REP	**Timer T2 Repetitive Mode** This bit configures the repetition mode of timer T2. 0 T2 is non repetitive. 1 T2 is repetitive.
5 T2SP	**Timer T2 Stop** This trigger bit is used to stop timer T2. 0 No effect 1 Stop timer T2
6 T2TR	**Timer T2 Trigger** This trigger bit is used to start timer T2. 0 No effect 1 Start timer T2

Table 841. FR_TICCR field descriptions (continued)

Field	Description
7 T2ST	**Timer T2 State** This status bit provides the current state of timer T2. 0 Timer T2 is idle. 1 Timer T2 is running.
11 T1_REP	**Timer T1 Repetitive Mode** This bit configures the repetition mode of timer T1. 0 T1 is non repetitive. 1 T1 is repetitive.
13 T1SP	**Timer T1 Stop** This trigger bit is used to stop timer T1. 0 No effect 1 Stop timer T1
14 T1TR	**Timer T1 Trigger** This trigger bit is used to start timer T1. 0 No effect 1 Start timer T1
15 T1ST	**Timer T1 State** This status bit provides the current state of timer T1. 0 Timer T1 is idle. 1 Timer T1 is running.

Note: Both timers are deactivated immediately when the protocol enters a state different from POC:normal active or POC:normal passive.

46.5.2.42 Timer 1 Cycle Set Register (FR_TI1CYSR)

Offset: 0x005C

Access: Read/Write
Write: Anytime

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	T1_CYC_VAL						0	0	T1_CYC_MSK					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 847. Timer 1 Cycle Set Register (FR_TI1CYSR)

This register defines the cycle filter value and the cycle filter mask for timer T1. For a detailed description of timer T1, refer to *Section 46.6.17.1: Absolute timer T1*.

Table 842. FR_TI1CYSR field descriptions

Field	Description
2:7 T1_CYC_VAL	**Timer T1 Cycle Filter Value** This field defines the cycle filter value for timer T1.
10:15 T1_CYC_MSK	**Timer T1 Cycle Filter Mask** This field defines the cycle filter mask for timer T1.

Note: If the application modifies the value in this register while the timer is running, the change becomes effective immediately and timer T1 will expire according to the changed value.

46.5.2.43 Timer 1 Macrotick Offset Register (FR_TI1MTOR)

Offset: 0x005E

Access: Read/Write
Write: Anytime

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	T1_MTOFFSET													
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 848. Timer 1 Macrotick Offset Register (FR_TI1MTOR)

This register holds the macrotick offset value for timer T1. For a detailed description of timer T1, refer to *Section 46.6.17.1: Absolute timer T1*.

Table 843. FR_TI1MTOR field descriptions

Field	Description
2:15 T1_MTOFFSET	**Timer 1 Macrotick Offset** This field defines the macrotick offset value for timer 1.

Note: If the application modifies the value in this register while the timer is running, the change becomes effective immediately and timer T1 will expire according to the changed value.

46.5.2.44 Timer 2 Configuration Register 0 (FR_TI2CR0)

Offset: 0x0060

Access: Read/Write
Write: Anytime

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Reserved(1)		T2_CYC_VAL						Reserved(1)		T2_CYC_MSK					
W																
R	T2_MTCNT[31:16]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. It is read as 0. Application must not write 1 to this bit.

Figure 849. Timer 2 Configuration Register 0 (FR_TI2CR0)

The content of this register depends on the value of the T2_CFG bit in the Timer Configuration and Control Register (FR_TICCR). For a detailed description of timer T2, refer to *Section 46.6.17.2: Absolute/relative timer T2*.

Table 844. FR_TI2CR0 field descriptions

Field	Description
Fields for absolute timer T2 (FR_TICCR[T2_CFG] = 0)	
2:7 T2_CYC_VAL	**Timer T2 Cycle Filter Value** This field defines the cycle filter value for timer T2.
10:15 T2_CYC_MSK	**Timer T2 Cycle Filter Mask** This field defines the cycle filter mask for timer T2.
Fields for relative timer T2 (FR_TICCR[T2_CFG] = 1)	
0:15 T2_MTCNT[31:16]	**Timer T2 Macrotick High Word** This field defines the high word of the macrotick count for timer T2.

Note: If timer T2 is configured as an absolute timer and the application modifies the values in this register while the timer is running, the change becomes effective immediately and timer T2 will expire according to the changed values.

If timer T2 is configured as a relative timer and the application changes the values in this register while the timer is running, the change becomes effective when the timer has expired according to the old values.

46.5.2.45 Timer 2 Configuration Register 1 (FR_TI2CR1)

Offset: 0x0062

Access: Read/Write
Write: Anytime

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	Reserved(1)				T2_MTOFFSET											
R W	T2_MTCNT[15:0]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. It is read as 0. Application must not write 1 to this bit.

Figure 850. Timer 2 Configuration Register 1 (FR_TI2CR1)

The content of this register depends on the value of the T2_CFG bit in the Timer Configuration and Control Register (FR_TICCR). For a detailed description of timer T2, refer to *Section 46.6.17.2: Absolute/relative timer T2*.

Table 845. FR_TI2CR1 field descriptions

Field	Description
Fields for absolute timer T2 (FR_TICCR[T2_CFG] = 0)	
2:15 T2_MTOFFSET	**Timer T2 Macrotick Offset** This field holds the macrotick offset value for timer T2.

Table 845. FR_TI2CR1 field descriptions

Field	Description
Fields for relative timer T2 (FR_TICCR[T2_CFG] = 1)	
0:15 T2_MTCNT[15:0]	**Timer T2 Macrotick Low Word** This field defines the low word of the macrotick value for timer T2.

Note: If timer T2 is configured as an absolute timer and the application modifies the values in this register while the timer is running, the change becomes effective immediately and the timer T2 will expire according to the changed values.

If timer T2 is configured as a relative timer and the application changes the values in this register while the timer is running, the change becomes effective when the timer has expired according to the old values.

46.5.2.46 Slot Status Selection Register (FR_SSSR)

Offset: 0x0064

Access: Read/Write(1)
Write: Anytime

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	SEL		0	SLOTNUMBER										
W	WMD															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. 16-bit write access required

Figure 851. Slot Status Selection Register (FR_SSSR)

This register is used to access the four internal non-memory-mapped Slot Status Selection Registers FR_SSSR0 to FR_SSSR3. Each internal register selects a slot, or symbol window/NIT, whose status vector will be saved in the corresponding Slot Status Registers (FR_SSR0–FR_SSR7) according to *Table 847*. For a detailed description of slot status monitoring, refer to *Section 46.6.18: Slot status monitoring*.

Table 846. FR_SSSR field descriptions

Field	Description
0 WMD	**Write Mode** This control bit defines the write mode of this register. 0 Write to all fields in this register on write access 1 Write to SEL field only on write access

Table 846. FR_SSSR field descriptions

Field	Description
2:3 SEL	**Selector** This field selects one of the four internal slot status selection registers for access. 00 Select FR_SSSR0 01 Select FR_SSSR1 10 Select FR_SSSR2 11 Select FR_SSSR3
5:15 SLOTNUMBER	**Slot Number** This field specifies the number of the slot whose status will be saved in the corresponding slot status registers. **Note:** If this value is set to 0, the related slot status register provides the status of the symbol window after the NIT start, and provides the status of the NIT after the cycle start.

Table 847. Mapping between FR_SSSRn and FR_SSRn

Internal Slot Status Selection Register	Write the Slot Status of the Slot Selected by FR_SSSRn for each			
	Even Communication Cycle		Odd Communication Cycle	
	For Channel B to	For Channel A to	For Channel B to	For Channel A to
FR_SSSR0	FR_SSR0[0:7]	FR_SSR0[8:15]	FR_SSR1[0:7]	FR_SSR1[8:15]
FR_SSSR1	FR_SSR2[0:7]	FR_SSR2[8:15]	FR_SSR3[0:7]	FR_SSR3[8:15]
FR_SSSR2	FR_SSR4[0:7]	FR_SSR4[8:15]	FR_SSR5[0:7]	FR_SSR5[8:15]
FR_SSSR3	FR_SSR6[0:7]	FR_SSR6[8:15]	FR_SSR7[0:7]	FR_SSR7[8:15]

46.5.2.47 Slot Status Counter Condition Register (FR_SSCCR)

Offset: 0x0066

Access: Read/Write(1)
Write: Anytime

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	SEL		0	CNTCFG		MCY	VFR	SYF	NUF	SUF	STATUSMASK			
W	WMD															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. 16-bit write access required

Figure 852. Slot Status Counter Condition Register (FR_SSCCR)

This register is used to access and program the four internal non-memory-mapped Slot Status Counter Condition Registers FR_SSCCR0 to FR_SSCCR3. Each of these four internal Slot Status Counter Condition Registers defines the mode and the conditions for incrementing the counter in the corresponding Slot Status Counter Registers (FR_SSCR0–FR_SSCR3). The correspondence is given in *Table 849*. For a detailed description of slot status counters, refer to *Section 46.6.18.4: Slot Status Counter Registers*.

Table 848. FR_SSCCR field descriptions

Field	Description
0 WMD	**Write Mode** This control bit defines the write mode of this register. 0 Write to all fields in this register on write access. 1 Write to SEL field only on write access.
2:3 SEL	**Selector** This field selects one of the four internal slot counter condition registers for access. 00 select FR_SSCCR0 01 select FR_SSCCR1 10 select FR_SSCCR2 11 select FR_SSCCR3
5:6 CNTCFG	**Counter Configuration** These bit field controls the channel related incrementing of the slot status counter. 00 Increment by 1 if condition is fulfilled on channel A 01 Increment by 1 if condition is fulfilled on channel B 10 Increment by 1 if condition is fulfilled on at least one channel 11 Increment by 2 if condition is fulfilled on both channels Increment by 1 if condition is fulfilled on only one channel.
7 MCY	**Multi Cycle Selection** This bit defines whether the slot status counter accumulates over multiple communication cycles or provides information for the previous communication cycle only. 0 The Slot Status Counter provides information for the previous communication cycle only. 1 The Slot Status Counter accumulates over multiple communication cycles.
8 VFR	**Valid Frame Restriction** This bit is used to restrict the counter to received valid frames. 0 The counter is not restricted to valid frames only. 1 The counter is restricted to valid frames only.
9 SYF	**Sync Frame Restriction** This bit is used to restrict the counter to received frames with the sync frame indicator bit set to 1. 0 The counter is not restricted with respect to the sync frame indicator bit. 1 The counter is restricted to frames with the sync frame indicator bit set to 1.
10 NUF	**Null Frame Restriction** This bit is used to restrict the counter to received frames with the null frame indicator bit set to 0. 0 The counter is not restricted with respect to the null frame indicator bit. 1 The counter is restricted to frames with the null frame indicator bit set to 0.

Table 848. FR_SSCCR field descriptions (continued)

Field	Description
11 SUF	**Startup Frame Restriction** This bit is used to restrict the counter to received frames with the startup frame indicator bit set to 1. 0 The counter is not restricted with respect to the startup frame indicator bit. 1 The counter is restricted to received frames with the startup frame indicator bit set to 1.
12:15 STATUSMASK	**Slot Status Mask** This bit field is used to enable the counter with respect to the four slot status error indicator bits. STATUSMASK[3]: This bit enables the counting for slots with the syntax error indicator bit set to 1. STATUSMASK[2]: This bit enables the counting for slots with the content error indicator bit set to 1. STATUSMASK[1]: This bit enables the counting for slots with the boundary violation indicator bit set to 1. STATUSMASK[0]: This bit enables the counting for slots with the transmission conflict indicator bit set to 1.

Table 849. Mapping between internal FR_SSCCR*n* and FR_SSCR*n*

Condition Register	Condition Defined for Register
FR_SSCCR0	FR_SSCR0
FR_SSCCR1	FR_SSCR1
FR_SSCCR2	FR_SSCR2
FR_SSCCR3	FR_SSCR3

46.5.2.48 Slot Status Registers (FR_SSR*n*)

Offset: 0x0068 +*n**0x2 (*n* = 0 to 7) Access: Read-only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	VFB	SYB	NFB	SUB	SEB	CEB	BVB	TCB	VFA	SYA	NFA	SUA	SEA	CEA	BVA	TCA
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 853. Slot Status Registers (FR_SSR*n*)

Each of these eight registers holds the status vector of the slot specified in the corresponding internal Slot Status Selection Register, which can be programmed using the Slot Status Selection Register (FR_SSSR). Each register is updated after the end of the corresponding slot as shown in *Figure 963*. The register bits are directly related to the protocol variables and described in more detail in *Section 46.6.18: Slot status monitoring*.

Table 850. FR_SSR*n* field descriptions

Field	Description
0 VFB	**Valid Frame on Channel B** protocol-related variable: *vSS!ValidFrame* channel B. 0 *vSS!ValidFrame* = 0 1 *vSS!ValidFrame* = 1
1 SYB	**Sync Frame Indicator Channel B** protocol-related variable: *vRF!Header!SyFIndicator* channel B. 0 *vRF!Header!SyFIndicator* = 0 1 *vRF!Header!SyFIndicator* = 1
2 NFB	**Null Frame Indicator Channel B** protocol-related variable: *vRF!Header!NFIndicator* channel B. 0 *vRF!Header!NFIndicator* = 0 1 *vRF!Header!NFIndicator* = 1
3 SUB	**Startup Frame Indicator Channel B** protocol-related variable: *vRF!Header!SuFIndicator* channel B. 0 *vRF!Header!SuFIndicator* = 0 1 *vRF!Header!SuFIndicator* = 1
4 SEB	**Syntax Error on Channel B** protocol-related variable: *vSS!SyntaxError* channel B. 0 *vSS!SyntaxError* = 0 1 *vSS!SyntaxError* = 1
5 CEB	**Content Error on Channel B** protocol-related variable: *vSS!ContentError* channel B. 0 *vSS!ContentError* = 0 1 *vSS!ContentError* = 1
6 BVB	**Boundary Violation on Channel B** protocol-related variable: *vSS!BViolation* channel B. 0 *vSS!BViolation* = 0 1 *vSS!BViolation* = 1
7 TCB	**Transmission Conflict on Channel B** protocol-related variable: *vSS!TxConflict* channel B. 0 *vSS!TxConflict* = 0 1 *vSS!TxConflict* = 1
8 VFA	**Valid Frame on Channel A** protocol-related variable: *vSS!ValidFrame* channel A. 0 *vSS!ValidFrame* = 0 1 *vSS!ValidFrame* = 1
9 SYA	**Sync Frame Indicator Channel A** protocol-related variable: *vRF!Header!SyFIndicator* channel A. 0 *vRF!Header!SyFIndicator* = 0 1 *vRF!Header!SyFIndicator* = 1
10 NFA	**Null Frame Indicator Channel A** protocol-related variable: *vRF!Header!NFIndicator* channel A. 0 *vRF!Header!NFIndicator* = 0 1 *vRF!Header!NFIndicator* = 1

Table 850. FR_SSR*n* field descriptions (continued)

Field	Description
11SUA	**Startup Frame Indicator Channel A** protocol-related variable: *vRF!Header!SuFIndicator* channel A. 0 *vRF!Header!SuFIndicator* = 0 1 *vRF!Header!SuFIndicator* = 1
12SEA	**Syntax Error on Channel A** protocol-related variable: *vSS!SyntaxError* channel A. 0 *vSS!SyntaxError* = 0 1 *vSS!SyntaxError* = 1
13CEA	**Content Error on Channel A** protocol-related variable: *vSS!ContentError* channel A. 0 *vSS!ContentError* = 0 1 *vSS!ContentError* = 1
14BVA	**Boundary Violation on Channel A** protocol-related variable: *vSS!BViolation* channel A. 0 *vSS!BViolation* = 0 1 *vSS!BViolation* = 1
15TCA	**Transmission Conflict on Channel A** protocol-related variable: *vSS!TxConflict* channel A. 0 *vSS!TxConflict* = 0 1 *vSS!TxConflict* = 1

46.5.2.49 Slot Status Counter Registers (FR_SSCR*n*)

Offset: 0x0078 +*n**0x2 (*n* = 0 to 3) Access: Read-only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	SLOTSTATUSCNT															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 854. Slot Status Counter Registers (FR_SSCR*n*)

Each of these four registers provides the slot status counter value for the previous communication cycle(s) and is updated at the cycle start. The provided value depends on the control bits and fields in the related internal Slot Status Counter Condition Register (FR_SSCCRn), which can be programmed by using the Slot Status Counter Condition Register (FR_SSCCR). For more details, see *Section 46.6.18.4: Slot Status Counter Registers*.

Note: If the counter has reached its maximum value 0xFFFF and is in Multicycle mode (that is, FR_SSCCRn[MCY] = 1), the counter is not reset to 0x0000. The application can reset the counter by clearing the FR_SSCCRn[MCY] bit and waiting for the next cycle start, when the CC clears the counter. Subsequently, the counter can be put into Multicycle mode again.

Table 851. FR_SSCR*n* field descriptions

Field	Description
0:15 SLOTSTATUSCNT	**Slot Status Counter** This field provides the current value of the Slot Status Counter.

46.5.2.50 MTS A Configuration Register (FR_MTSACFR)

Offset: 0x0080

Access: Read/Write
Write: MTE: Anytime
CYCCNTMSK,CYCCNTVAL:*POC:config*

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	MTE	0	CYCCNTMSK						0	0	CYCCNTVAL					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 855. MTS A Configuration Register (FR_MTSACFR)

This register controls the transmission of the Media Access Test Symbol (MTS) on channel A. For more details, see *Section 46.6.13: MTS generation*.

Table 852. FR_MTSACFR field descriptions

Field	Description
0 MTE	**Media Access Test Symbol Transmission Enable** This control bit is used to enable and disable the transmission of the MTS in the selected set of cycles. 0 MTS transmission disabled 1 MTS transmission enabled
2:7 CYCCNTMSK	**Cycle Counter Mask** This field provides the filter mask for the MTS cycle count filter.
10:15 CYCCNTVAL	**Cycle Counter Value** This field provides the filter value for the MTS cycle count filter.

46.5.2.51 MTS B Configuration Register (FR_MTSBCFR)

Offset: 0x0082

Access: Read/Write
Write: MTE: Anytime
CYCCNTMSK,CYCCNTVAL:*POC:config*

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	MTE	0	CYCCNTMSK						0	0	CYCCNTVAL					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 856. MTS B Configuration Register (FR_MTSBCFR)

This register controls the transmission of the Media Access Test Symbol (MTS) on channel B. For more details, see *Section 46.6.13: MTS generation*.

Table 853. FR_MTSBCFR field descriptions

Field	Description
0 MTE	**Media Access Test Symbol Transmission Enable** This control bit is used to enable and disable the transmission of the MTS in the selected set of cycles. 0 MTS transmission disabled 1 MTS transmission enabled
2:7 CYCCNTMSK	**Cycle Counter Mask** This field provides the filter mask for the MTS cycle count filter.
10:15 CYCCNTVAL	**Cycle Counter Value** This field provides the filter value for the MTS cycle count filter.

46.5.2.52 Receive Shadow Buffer Index Register (FR_RSBIR)

Access: Read/Write(1)
Write: WMD, SEL: Any Time
RSBIDX: *POC:config*

Offset: 0x0084

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	SEL		0	0	0	0	0	$RSBIDX_{A1}$/$RSBIDX_{A2}$/ $RSBIDX_{B1}$/$RSBIDX_{B2}$						
W	WMD															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. 16-bit write access required

Figure 857. Receive Shadow Buffer Index Register (FR_RSBIR)

This register is used to provide and retrieve the indices of the message buffer header fields currently associated with the receive shadow buffers. For more details on the receive shadow buffer concept, refer to *Section 46.6.6.3.5: Receive shadow buffer concept*.

Table 854. FR_RSBIR field descriptions

Field	Description
0 WMD	**Write Mode** This bit controls the write mode for this register. 0 Update SEL and RSBIDX field on register write 1 Update only SEL field on register write

Table 854. FR_RSBIR field descriptions (continued)

Field	Description
2:3 SEL	**Selector** This field is used to select the internal receive shadow buffer index register for access. 00 FR_RSBIR_A1: receive shadow buffer index register for channel A, segment 1 01 FR_RSBIR_A2: receive shadow buffer index register for channel A, segment 2 10 FR_RSBIR_B1: receive shadow buffer index register for channel B, segment 1 11 FR_RSBIR_B2: receive shadow buffer index register for channel B, segment 2
9:15 $RSBIDX_{A1}$/ $RSBIDX_{A2}$/ $RSBIDX_{B1}$/ $RSBIDX_{B2}$	**Receive Shadow Buffer Index** This field contains the current index of the message buffer header field of the receive shadow message buffer selected by the SEL field. The CC uses this index to determine the physical location of the shadow buffer header field in the FlexRay memory area. The CC will update this field during receive operation.The application provides initial message buffer header index value in the configuration phase. - CC: Updates the message buffer header index after successful reception. Application: Provides initial message buffer header index. Legal Values are $0 \le i \le 67$. Illegal values will be detected during the message buffer search.

46.5.2.53 Receive FIFO Watermark and Selection Register (FR_RFWMSR)

Offset: 0x0086

Access: Read/Write

Write: *WM_A/WM_B: POC:config, SEL: Anytime*

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	WM_A/WM_B								0	0	0	0	0	0	0	SEL
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 858. Receive FIFO Watermark and Selection Register (FR_RFWMSR)

Table 855. FR_RFWMSR field descriptions

Field	Description
0:7 WM_A/WM_B	**Watermark** This field defines the watermark value for the selected FIFO. This value is used to control the generation of the almost full interrupt flags.
15 SEL	**Select** This control bit selects the receiver FIFO for subsequent programming. 0 Receiver FIFO for channel A selected 1 Receiver FIFO for channel B selected

This register is used to:

- Select a receiver FIFO for subsequent programming access through the receiver FIFO configuration registers summarized in *Table 856*.
- Define the watermark for the selected FIFO.

Table 856. SEL-controlled receiver FIFO registers

Register
Receive FIFO Start Index Register (FR_RFSIR)
Receive FIFO Depth and Size Register (FR_RFDSR)
Receive FIFO Message ID Acceptance Filter Value Register (FR_RFMIDAFVR)
Receive FIFO Message ID Acceptance Filter Mask Register (FR_RFMIDAFMR)
Receive FIFO Frame ID Rejection Filter Value Register (FR_RFFIDRFVR)
Receive FIFO Frame ID Rejection Filter Mask Register (FR_RFFIDRFMR)
Receive FIFO Range Filter Configuration Register (FR_RFRFCFR)
Receive FIFO Range Filter Control Register (FR_RFRFCTR)

46.5.2.54 Receive FIFO Start Index Register (FR_RFSIR)

Offset: 0x0088

Access: Read/Write
Write: *POC:config*

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	$SIDX_A/SIDX_B$									
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 859. Receive FIFO Start Index Register (FR_RFSIR)

This register defines the message buffer header index of the first message buffer of the selected FIFO.

Table 857. FR_RFSIR field descriptions

Field	Description
6:15 $SIDX_A/SIDX_B$	**Start Index** This field defines the number of the message buffer header field of the selected FIFO's first message buffer. The CC uses the value of the SIDX field to determine the physical location of the receiver FIFO's first message buffer header field.

46.5.2.55 Receive FIFO Depth and Size Register (FR_RFDSR)

Offset: 0x008A

Access: Read/Write
Write: *POC:config*

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	$FIFO_DEPTH_A/FIFO_DEPTH_B$								0	$ENTRY_SIZE_A/ENTRY_SIZE_B$						
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 860. Receive FIFO Depth and Size Register (FR_RFDSR)

This register defines the structure of the selected FIFO, that is the number of entries and the size of each entry.

Table 858. RFDSR field descriptions

Field	Description
0:7 $FIFO_DEPTH_A$/ $FIFO_DEPTH_B$	**FIFO Depth** This field defines the depth (the number of entries) of the selected FIFO. **Note:** If the FIFO_DEPTH is configured to 0, FR_RFFIDRFMR[FIDRFMSK] must be configured to 0 as well to ensure that no frames are received into the FIFO.
9:15 $ENTRY_SIZE_A$/ $ENTRY_SIZE_B$	**Entry Size** This field defines the size of the frame data sections for the selected FIFO in two-byte entities. =

46.5.2.56 Receive FIFO A Read Index Register (FR_RFARIR)

Offset: 0x008C Access: Read-only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	RDIDX									
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 861. Receive FIFO A Read Index Register (FR_RFARIR)

This register provides the message buffer header index of the next available FIFO A entry that the application can read.

Table 859. FR_RFARIR field descriptions

Field	Description
6:15 RDIDX	**Read Index** This field provides the message buffer header index of the next available FIFO message buffer that the application can read.

Note: If the FIFO is empty, the RDIDX field points to an physical message buffer with invalid content.

46.5.2.57 Receive FIFO B Read Index Register (FR_RFBRIR)

Offset: 0x008E Access: Read-only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	RDIDX									
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 862. Receive FIFO B Read Index Register (FR_RFBRIR)

This register provides the message buffer header index of the next available FIFO B entry that the application can read.

Table 860. FR_RFBRIR field descriptions

Field	Description
6:15 RDIDX	**Read Index** This field provides the message buffer header index of the next available FIFO message buffer that the application can read.

Note: If the FIFO is empty, the RDIDX field points to an physical message buffer with invalid content.

46.5.2.58 Receive FIFO Message ID Acceptance Filter Value Register (FR_RFMIDAFVR)

Offset: 0x0090

Access: Read/Write
Write: *POC:config*

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	$MIDAFVAL_A$/$MIDAFVAL_B$															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 863. Receive FIFO Message ID Acceptance Filter Value Register (FR_RFMIDAFVR)

This register defines the filter value for the message ID acceptance filter of the selected FIFO. For details on message ID filtering, see *Section 46.6.9.9: FIFO filtering*.

Table 861. FR_RFMIDAFVR field descriptions

Field	Description
0:15 $MIDAFVAL_A$/ $MIDAFVAL_B$	**Message ID Acceptance Filter Value** Filter value for the message ID acceptance filter.

46.5.2.59 Receive FIFO Message ID Acceptance Filter Mask Register (FR_RFMIDAFMR)

Offset: 0x0092

Access: Read/Write
Write: *POC:config*

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	$MIDAFMSK_A$/$MIDAFMSK_B$															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 864. Receive FIFO Message ID Acceptance Filter Mask Register (FR_RFMIDAFMR)

This register defines the filter mask for the message ID acceptance filter of the selected FIFO. For details on message ID filtering, see *Section 46.6.9.9: FIFO filtering*.

Table 862. FR_RFMIDAFMR field descriptions

Field	Description
0:15 $MIDAFMSK_A$/ $MIDAFMSK_B$	**Message ID Acceptance Filter Mask** Filter mask for the message ID acceptance filter.

46.5.2.60 Receive FIFO Frame ID Rejection Filter Value Register (FR_RFFIDRFVR)

Offset: 0x0094

Access: Read/Write
Write: *POC:config*

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	$FIDRFVAL_A$/$FIDRFVAL_B$										
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 865. Receive FIFO Frame ID Rejection Filter Value Register (FR_RFFIDRFVR)

This register defines the filter value for the frame ID rejection filter of the selected FIFO. For details on frame ID filtering, see *Section 46.6.9.9: FIFO filtering*.

Table 863. FR_RFFIDRFVR field descriptions

Field	Description
5:15 $FIDRFVAL_A$/ $FIDRFVAL_B$	**Frame ID Rejection Filter Value** Filter value for the frame ID rejection filter.

46.5.2.61 Receive FIFO Frame ID Rejection Filter Mask Register (FR_RFFIDRFMR)

Offset: 0x0096

Access: Read/Write
Write: *POC:config*

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	$FIDRFMSK_A$/$FIDRFMSK_B$										
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 866. Receive FIFO Frame ID Rejection Filter Mask Register (FR_RFFIDRFMR)

This register defines the filter mask for the frame ID rejection filter of the selected FIFO. For details on frame ID filtering see *Section 46.6.9.9: FIFO filtering*.

Table 864. FR_RFFIDRFMR field descriptions

Field	Description
5:15 $FIDRFMSK_A$/ $FIDRFMSK_B$	**Frame ID Rejection Filter Mask** Filter mask for the frame ID rejection filter.

46.5.2.62 Receive FIFO Range Filter Configuration Register (FR_RFRFCFR)

Offset: 0x0098

Access: Read/Write(1)
Write: WMD, IBD, SEL: Any Time
SID: *POC:config*

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	IBD	SEL		0	SID_A/SID_B										
W	WMD															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. 16-bit write access required

Figure 867. Receive FIFO Range Filter Configuration Register (FR_RFRFCFR)

This register provides access to the four internal frame ID range filter boundary registers of the selected FIFO. For details on frame ID range filter, see *Section 46.6.9.9: FIFO filtering*.

Table 865. FR_RFRFCFR field descriptions

Field	Description
0 WMD	**Write Mode** This control bit defines the write mode of this register. 0 Write to all fields in this register on write access 1 Write to SEL and IBD field only on write access
1 IBD	**Interval Boundary** This control bit selects the interval boundary to be programmed with the SID value. 0 Program lower interval boundary 1 Program upper interval boundary
2:3 SEL	**Filter Selector** This control field selects the frame ID range filter to be accessed. 00 Select frame ID range filter 0 01 Select frame ID range filter 1 10 Select frame ID range filter 2 11 Select frame ID range filter 3
5:15 SID_A/SID_B	**Slot ID** Defines the IBD-selected frame ID boundary value for the SEL-selected range filter.

46.5.2.63 Receive FIFO Range Filter Control Register (FR_RFRFCTR)

Offset: 0x009A

Access: Read/Write
Write: Anytime

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	F3MD	F2MD	F1MD	F0MD	0	0	0	0	F3EN	F2EN	F1EN	F0EN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 868. Receive FIFO Range Filter Control Register (FR_RFRFCTR)

This register is used to enable and disable each frame ID range filter and to define whether it is running as an acceptance or rejection filter.

Table 866. FR_RFRFCTR field descriptions

Field	Description
4 F3MD	**Range Filter 3 Mode** This control bit defines the filter mode of the frame ID range filter 3. 0 Range filter 3 runs as acceptance filter 1 Range filter 3 runs as rejection filter
5 F2MD	**Range Filter 2 Mode** This control bit defines the filter mode of the frame ID range filter 2. 0 Range filter 2 runs as acceptance filter 1 Range filter 2 runs as rejection filter
6 F1MD	**Range Filter 1 Mode** This control bit defines the filter mode of the frame ID range filter 1. 0 Range filter 1 runs as acceptance filter 1 Range filter 1 runs as rejection filter
7 F0MD	**Range Filter 0 Mode** This control bit defines the filter mode of the frame ID range filter 0. 0 Range filter 0 runs as acceptance filter 1 Range filter 0 runs as rejection filter
12 F3EN	**Range Filter 3 Enable** This control bit is used to enable and disable the frame ID range filter 3. 0 Range filter 3 disabled 1 Range filter 3 enabled
13 F2EN	**Range Filter 2 Enable** This control bit is used to enable and disable the frame ID range filter 2. 0 Range filter 2 disabled 1 Range filter 2 enabled
14 F1EN	**Range Filter 1 Enable** This control bit is used to enable and disable the frame ID range filter 1. 0 Range filter 1 disabled 1 Range filter 1 enabled
15 F0EN	**Range Filter 0 Enable** This control bit is used to enable and disable the frame ID range filter 0. 0 Range filter 0 disabled 1 Range filter 0 enabled

46.5.2.64 Last Dynamic Transmit Slot Channel A Register (FR_LDTXSLAR)

Offset: 0x009C Access: Read-only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	LDYNTXSLOTA										
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 869. Last Dynamic Transmit Slot Channel A Register (FR_LDTXSLAR)

This register provides the number of the last transmission slot in the dynamic segment for channel A. This register is updated after the end of the dynamic segment and before the start of the next communication cycle.

Table 867. FR_LDTXSLAR field descriptions

Field	Description
5:15 LDYNTXSLOTA	**Last Dynamic Transmission Slot Channel A** Protocol-related variable: *zLastDynTxSlot* channel A. Number of the last transmission slot in the dynamic segment for channel A. If no frame was transmitted during the dynamic segment on channel A, the value of this field is set to 0.

46.5.2.65 Last Dynamic Transmit Slot Channel B Register (FR_LDTXSLBR)

Offset: 0x009E Access: Read-only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	LDYNTXSLOTB										
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 870. Last Dynamic Transmit Slot Channel B Register (FR_LDTXSLBR)

This register provides the number of the last transmission slot in the dynamic segment for channel B. This register is updated after the end of the dynamic segment and before the start of the next communication cycle.

Table 868. FR_LDTXSLBR field descriptions

Field	Description
5:15 LDYNTXSLOTB	**Last Dynamic Transmission Slot Channel B** Protocol-related variable: *zLastDynTxSlot* channel B. Number of the last transmission slot in the dynamic segment for channel B. If no frame was transmitted during the dynamic segment on channel B the value of this field is set to 0.

46.5.2.66 Protocol Configuration Registers

The following configuration registers provide the necessary configuration information to the protocol engine. The individual values in the registers are described in *Table 869*. For more details about the FlexRay related configuration parameters and the allowed parameter ranges, see *FlexRay Communications System Protocol Specification, Version 2.1 Rev A*.

Table 869. Protocol Configuration Register fields

Name	Description[1]	Min	Max	Unit	FR_PCR
coldstart_attempts	gColdstartAttempts	—	—	number	3
action_point_offset	gdActionPointOffset - 1	—	—	MT	0
cas_rx_low_max	gdCASRxLowMax - 1	—	—	gdBit	4

Table 869. Protocol Configuration Register fields (continued)

Name	Description[1]	Min	Max	Unit	FR_PCR
dynamic_slot_idle_phase	gdDynamicSlotIdlePhase	—	—	minislot	28
minislot_action_point_offset	gdMinislotActionPointOffset - 1	—	—	MT	3
minislot_after_action_point	gdMinislot - gdMinislotActionPointOffset - 1	—	—	MT	2
static_slot_length	gdStaticSlot	—	—	MT	0
static_slot_after_action_point	gdStaticSlot - gdActionPointOffset - 1	—	—	MT	13
symbol_window_exists	gdSymbolWindow!=0	0	1	bool	9
symbol_window_after_action_point	gdSymbolWindow - gdActionPointOffset - 1	—	—	MT	6
tss_transmitter	gdTSSTransmitter	—	—	gdBit	5
wakeup_symbol_rx_idle	gdWakeupSymbolRxIdle	—	—	gdBit	5
wakeup_symbol_rx_low	gdWakeupSymbolRxLow	—	—	gdBit	3
wakeup_symbol_rx_window	gdWakeupSymbolRxWindow	—	—	gdBit	4
wakeup_symbol_tx_idle	gdWakeupSymbolTxIdle	—	—	gdBit	8
wakeup_symbol_tx_low	gdWakeupSymbolTxLow	—	—	gdBit	5
noise_listen_timeout	(gListenNoise * pdListenTimeout) - 1	—	—	µT	16/17
macro_initial_offset_a	pMacroInitialOffset[A]	—	—	MT	6
macro_initial_offset_b	pMacroInitialOffset[B]	—	—	MT	16
macro_per_cycle	gMacroPerCycle	—	—	MT	10
macro_after_first_static_slot	gMacroPerCycle - gdStaticSlot	—	—	MT	1
macro_after_offset_correction	gMacroPerCycle - gOffsetCorrectionStart	—	—	MT	28
max_without_clock_correction_fatal	gMaxWithoutClockCorrectionFatal	—	—	cyclepairs	8
max_without_clock_correction_passive	gMaxWithoutClockCorrectionPassive	—	—	cyclepairs	8
minislot_exists	gNumberOfMinislots!=0	0	1	bool	9
minislots_max	gNumberOfMinislots - 1	—	—	minislot	29
number_of_static_slots	gNumberOfStaticSlots	—	—	static slot	2
offset_correction_start	gOffsetCorrectionStart	—	—	MT	11
payload_length_static	gPayloadLengthStatic	—	—	2 bytes	19
max_payload_length_dynamic	pPayloadLengthDynMax	—	—	2 bytes	24
first_minislot_action_point_offset	max(gdActionPointOffset, gdMinislotActionPointOffset) - 1	—	—	MT	13
allow_halt_due_to_clock	pAllowHaltDueToClock	—	—	bool	26
allow_passive_to_active	pAllowPassiveToActive	—	—	cyclepairs	12

Table 869. Protocol Configuration Register fields (continued)

Name	Description[1]	Min	Max	Unit	FR_PCR
cluster_drift_damping	pClusterDriftDamping	—	—	µT	24
comp_accepted_startup_range_a	pdAcceptedStartupRange - pDelayCompensation[A]	—	—	µT	22
comp_accepted_startup_range_b	pdAcceptedStartupRange - pDelayCompensation[B]	—	—	µT	26
listen_timeout	pdListenTimeout - 1	—	—	µT	14/15
key_slot_id	pKeySlotId	—	—	number	18
key_slot_used_for_startup	pKeySlotUsedForStartup	—	—	bool	11
key_slot_used_for_sync	pKeySlotUsedForSync	—	—	bool	11
latest_tx	gNumberOfMinislots - pLatestTx	—	—	minislot	21
sync_node_max	gSyncNodeMax	—	—	number	30
micro_initial_offset_a	pMicroInitialOffset[A]	—	—	µT	20
micro_initial_offset_b	pMicroInitialOffset[B]	—	—	µT	20
micro_per_cycle	pMicroPerCycle	—	—	µT	22/23
micro_per_cycle_min	pMicroPerCycle - pdMaxDrift	—	—	µT	24/25
micro_per_cycle_max	pMicroPerCycle + pdMaxDrift	—	—	µT	26/27
micro_per_macro_nom_half	round(pMicroPerMacroNom / 2)	—	—	µT	7
offset_correction_out	pOffsetCorrectionOut	—	—	µT	9
rate_correction_out	pRateCorrectionOut	—	—	µT	14
single_slot_enabled	pSingleSlotEnabled	—	—	bool	10
wakeup_channel	pWakeupChannel	see *Table 870*			10
wakeup_pattern	pWakeupPattern	—	—	number	18
decoding_correction_a	pDecodingCorrection + pDelayCompensation[A] + 2	—	—	µT	19
decoding_correction_b	pDecodingCorrection + pDelayCompensation[B] + 2	—	—	µT	7
key_slot_header_crc	header CRC for key slot	0x000	0x7FF	number	12
extern_offset_correction	pExternOffsetCorrection	—	—	µT	29
extern_rate_correction	pExternRateCorrection	—	—	µT	21

1. See *FlexRay Communications System Protocol Specification, Version 2.1 Rev A* for detailed protocol parameter definitions

Table 870. Wake up channel selection

wakeup_channel	Wake up Channel
0	A
1	B

46.5.2.66.1 Protocol Configuration Register 0 (FR_PCR0)

Offset: 0x00A0

Access: Read/Write
Write: *POC:config*

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	action_point_offset								static_slot_length							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 871. Protocol Configuration Register 0 (FR_PCR0)

46.5.2.66.2 Protocol Configuration Register 1 (FR_PCR1)

Offset: 0x00A2

Access: Read/Write
Write: *POC:config*

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	macro_after_first_static_slot													
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 872. Protocol Configuration Register 1 (FR_PCR1)

46.5.2.66.3 Protocol Configuration Register 2 (FR_PCR2)

Offset: 0x00A4

Access: Read/Write
Write: *POC:config*

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	minislot_after_action_point								number_of_static_slots							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 873. Protocol Configuration Register 2 (FR_PCR2)

46.5.2.66.4 Protocol Configuration Register 3 (FR_PCR3)

Offset: 0x00A6

Access: Read/Write
Write: *POC:config*

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	wakeup_symbol_rx_low						minislot_action_point_offset[4:0]						coldstart_attempts			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 874. Protocol Configuration Register 3 (FR_PCR3)

46.5.2.66.5 Protocol Configuration Register 4 (FR_PCR4)

Offset: 0x00A8

Access: Read/Write
Write: *POC:config*

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	cas_rx_low_max								wakeup_symbol_rx_window							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 875. Protocol Configuration Register 4 (FR_PCR4)

46.5.2.66.6 Protocol Configuration Register 5 (FR_PCR5)

Offset: 0x00AA

Access: Read/Write
Write: *POC:config*

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	tss_transmitter				wakeup_symbol_tx_low								wakeup_symbol_rx_idle			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 876. Protocol Configuration Register 5 (FR_PCR5)

46.5.2.66.7 Protocol Configuration Register 6 (FR_PCR6)

Offset: 0x00AC

Access: Read/Write
Write: *POC:config*

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	symbol_window_after_action_point							macro_initial_offset_a							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 877. Protocol Configuration Register 6 (FR_PCR6)

46.5.2.66.8 Protocol Configuration Register 7 (FR_PCR7)

Offset: 0x00AE

Access: Read/Write
Write: *POC:config*

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	decoding_correction_b								micro_per_macro_nom_half							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 878. Protocol Configuration Register 7 (FR_PCR7)

46.5.2.66.9 Protocol Configuration Register 8 (FR_PCR8)

Offset: 0x00B0

Access: Read/Write
Write: *POC:config*

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	max_without_clock_correction_fatal				max_without_clock_correction_passive				wakeup_symbol_tx_idle							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 879. Protocol Configuration Register 8 (FR_PCR8)

46.5.2.66.10 Protocol Configuration Register 9 (FR_PCR9)

Offset: 0x00B2

Access: Read/Write
Write: *POC:config*

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	mini slot_exists	sym bol_win	offset_correction_out													
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 880. Protocol Configuration Register 9 (FR_PCR9)

46.5.2.66.11 Protocol Configuration Register 10 (FR_PCR10)

Offset: 0x00B4

Access: Read/Write
Write: *POC:config*

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	single_slot_enabled	wake up_channel	macro_per_cycle													
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 881. Protocol Configuration Register 10 (FR_PCR10)

46.5.2.66.12 Protocol Configuration Register 11 (FR_PCR11)

Offset: 0x00B6

Access: Read/Write
Write: *POC:config*

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	key_slot_used_for_start up	key_slot_used_for_sync	offset_correction_start													
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 882. Protocol Configuration Register 11 (FR_PCR11)

46.5.2.66.13 Protocol Configuration Register 12 (FR_PCR12)

Offset: 0x00B8

Access: Read/Write
Write: *POC:config*

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	allow_passive_to_active				key_slot_header_crc											
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 883. Protocol Configuration Register 12 (FR_PCR12)

46.5.2.66.14 Protocol Configuration Register 13 (FR_PCR13)

Offset: 0x00BA

Access: Read/Write
Write: *POC:config*

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	first_minislot_action_point_offset								static_slot_after_action_point							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 884. Protocol Configuration Register 13 (FR_PCR13)

46.5.2.66.15 Protocol Configuration Register 14 (FR_PCR14)

Offset: 0x00BC

Access: Read/Write
Write: *POC:config*

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	rate_correction_out												listen_timeout[20:16]			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 885. Protocol Configuration Register 14 (FR_PCR14)

46.5.2.66.16 Protocol Configuration Register 15 (FR_PCR15)

Offset: 0x00BE

Access: Read/Write
Write: *POC:config*

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	listen_timeout[15:0]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 886. Protocol Configuration Register 15 (FR_PCR15)

46.5.2.66.17 Protocol Configuration Register 16 (FR_PCR16)

Offset: 0x00C0

Access: Read/Write
Write: *POC:config*

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	macro_initial_offset_b								noise_listen_timeout[24:16]							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 887. Protocol Configuration Register 16 (FR_PCR16)

46.5.2.66.18 Protocol Configuration Register 17 (FR_PCR17)

Offset: 0x00C2

Access: Read/Write
Write: *POC:config*

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	noise_listen_timeout[15:0]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 888. Protocol Configuration Register 17 (FR_PCR17)

46.5.2.66.19 Protocol Configuration Register 18 (FR_PCR18)

Offset: 0x00C4

Access: Read/Write
Write: *POC:config*

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	wakeup_pattern								key_slot_id							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 889. Protocol Configuration Register 18 (FR_PCR18)

46.5.2.66.20 Protocol Configuration Register 19 (FR_PCR19)

Offset: 0x00C6

Access: Read/Write
Write: *POC:config*

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	decoding_correction_a								payload_length_static							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 890. Protocol Configuration Register 19 (FR_PCR19)

46.5.2.66.21 Protocol Configuration Register 20 (FR_PCR20)

Offset: 0x00C8

Access: Read/Write
Write: *POC:config*

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	micro_initial_offset_b								micro_initial_offset_a							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 891. Protocol Configuration Register 20 (FR_PCR20)

46.5.2.66.22 Protocol Configuration Register 21 (FR_PCR21)

Offset: 0x00CA

Access: Read/Write
Write: *POC:config*

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	extern_rate_ correction				latest_tx											
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 892. Protocol Configuration Register 21 (FR_PCR21)

46.5.2.66.23 Protocol Configuration Register 22 (FR_PCR22)

Offset: 0x00CC

Access: Read/Write
Write: *POC:config*

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	R(1)				comp_accepted_startup_range_a								micro_per_cycle[19:16]			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 893. Protocol Configuration Register 22 (FR_PCR22)

1. Reserved bit or field. Application must not write any value different from the reset value.

46.5.2.66.24 Protocol Configuration Register 23 (FR_PCR23)

Offset: 0x00CE

Access: Read/Write
Write: *POC:config*

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	micro_per_cycle[15:0]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 894. Protocol Configuration Register 23 (FR_PCR23)

46.5.2.66.25 Protocol Configuration Register 24 (FR_PCR24)

Offset: 0x00D0

Access: Read/Write
Write: *POC:config*

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	cluster_drift_damping				max_payload_length_dynamic								micro_per_cycle_min [19:16]			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 895. Protocol Configuration Register 24 (FR_PCR24)

46.5.2.66.26 Protocol Configuration Register 25 (FR_PCR25)

Offset: 0x00D2

Access: Read/Write
Write: *POC:config*

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	micro_per_cycle_min[15:0]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 896. Protocol Configuration Register 25 (FR_PCR25)

46.5.2.66.27 Protocol Configuration Register 26 (FR_PCR26)

Offset: 0x00D4

Access: Read/Write
Write: *POC:config*

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	allow_halt_ due_to_clock	comp_accepted_startup_range_b											micro_per_cycle_max [19:16]			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 897. Protocol Configuration Register 26 (FR_PCR26)

46.5.2.66.28 Protocol Configuration Register 27 (FR_PCR27)

Offset: 0x00D6

Access: Read/Write
Write: *POC:config*

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	micro_per_cycle_max[15:0]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 898. Protocol Configuration Register 27 (FR_PCR27)

46.5.2.66.29 Protocol Configuration Register 28 (FR_PCR28)

Offset: 0x00D8

Access: Read/Write
Write: *POC:config*

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	dynamic_slot_idle_phase				macro_after_offset_correction											
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 899. Protocol Configuration Register 28 (FR_PCR28)

46.5.2.66.30 Protocol Configuration Register 29 (FR_PCR29)

Offset: 0x00DA

Access: Read/Write
Write: *POC:config*

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	extern_offset_correction				minislots_max											
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 900. Protocol Configuration Register 29 (FR_PCR29)

46.5.2.66.31 Protocol Configuration Register 30 (FR_PCR30)

Offset: 0x00DC

Access: Read/Write
Write: *POC:config*

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	sync_node_max			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 901. Protocol Configuration Register 30 (FR_PCR30)

46.5.2.67 Stop Watch Count Register (FR_STPWR)

Offset: 0x000DE

Access: Read/Write
Read Reset Only: Anytime

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	STPW															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 902. Stop Watch Count High Register (FR_STPWHR)

Offset: 0x000E0

Access: Read/Write
Read Reset Only: Anytime

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	STPW															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 903. Stop Watch Count Low Register (FR_STPWLR)

Table 871. FR_STPWR field descriptions

Field	Description
0:15 STPW	Stop Watch Count Register This is the value of the stop watch counter running on system clock. This starts on cycle start even and stops on an external event “OS_tick”. For details see *Section 46.6.26*. This is a read reset type register, which means that when all the 32 bits are read, it automatically clears.

46.5.2.68 Protocol Event Output Enable and StopWatch Control Register (FR_PEOER)

Offset: 0x00E2

Access: Read/Write
Write: Anytime

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	STPW_EN	0	0	0	0	0	TIM2_EE	TIM1_EE	CYS_EE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 904. Protocol Event Output Enable and StopWatch Control Register (FR_PEOER)

This register defines whether or not the event output ports are enabled based on the events such as Stop watch count Cycle start and Timer1 and Timer2 expiry (see *Section 46.6.17: Timer support* for details).

Table 872. FR_PEOER field descriptions

Field	Description
7 STPW_EN	Stop watch count Enable This bit controls the stopwatch counter. 0 Stopwatch counter disabled 1 Stopwatch counter enabled
13 TIM2_EE	Timer 2 expired Event Output Enable This bit controls the event on port “fr_evt_tim2”. 0 Timer 2 expired event out disabled 1 Timer 2 expired event out enabled

Table 872. FR_PEOER field descriptions (continued)

Field	Description
14 TIM1_EE	Timer 1 expired Event Output Enable This bit controls the event on port “fr_evt_tim1”. 0 Timer 1 expired event out disabled 1 Timer 1 expired event out enabled
15 CYS_EE	Cycle Start Event Output Enable This bit controls the event on port “fr_evt_cyc”. 0 Cycle start event out disabled 1 Cycle start event out enabled

46.5.2.69 Receive FIFO Start Data Offset Register (FR_RFSDOR)

Offset: 0x00E6

Access: Read/Write
Write: *POC:config*

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	SDO_A/SDO_B															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 905. Receive FIFO Start Data Offset Register (FR_RFSDOR)

Table 873. FR_RFSDOR field descriptions

Field	Description
0:15 SDO_A/SDO_B	Start Data Field Offset This field defines the data field offset of the header field of the first message buffer of the selected FIFO. The CC uses the value of the SDO field to determine the physical location of the receiver FIFO’s first message buffer header field. For configuration constraints, see *Section 46.7.1.2: Configure data field offsets*.

Note: Since all data fields of the FIFO are of equal length and are located at subsequent system memory addresses, the content of the FR_RFSDOR register corresponds to the start address of the payload area of the selected FIFO.

46.5.2.70 Receive FIFO System Memory Base Address Register (FR_RFSYMBADR)

Offset: 0x00E8

Access: Read/Write
Write: Disabled mode

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	SMBA															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 906. Receive FIFO System Memory Base Address High Register (FR_RFSYMBADHR)

Offset: 0x00EA

Access: Read/Write
Write: Disabled mode

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	SMBA												0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 907. Receive FIFO System Memory Base Address Low Register (FR_RFSYMBADLR)

These registers define the system memory base address for the receive FIFO if the FIFO address mode bit FR_MCR[FAM] is set to 1. The system memory base address is used by the BMIF to calculate the physical memory address for system memory accesses for the FIFOs.

Table 874. FR_RFSYMBADR field descriptions

Field	Description
SMBA	System Memory Base Address This is the value of the system memory base address for the receive FIFO if the FIFO address mode bit FR_MCR[FAM] is set to 1. It is defined as a byte address.

46.5.2.71 Receive FIFO Periodic Timer Register (FR_RFPTR)

Offset: 0x00EC

Access: Read/Write
Write: *POC:config*

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	PTD													
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 908. Receive FIFO Periodic Timer Register (FR_RFPTR)

This register holds the periodic timer duration for the periodic FIFO timer. The periodic timer applies to both FIFOs (see *Section 46.6.9.3: FIFO periodic timer*).

Table 875. FR_RFPTR field descriptions

Field	Description
2:15 PTD	Periodic Timer Duration This value defines the periodic timer duration in terms of macroticks. 0000 Timer stays expired 3FFF Timer never expires other Timer expires after specified number of macroticks, expires and is restarted at each cycle start

46.5.2.72 Receive FIFO Fill Level and POP Count Register (FR_RFFLPCR)

Offset: 0x00EE Access: Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	FLB_PCB								FLA_PCA							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 909. Receive FIFO Fill Level and POP Count Register (FR_RFFLPCR)

This register provides the current fill level of the two receiver FIFOs and is used to pop a number of entries from the FIFOs.

Table 876. FR_RFFLPCR field descriptions

Field	Description
0:7 FLB_PCB	Fill Level FIFO B (FLB) This field provides the current number of entries in the FIFO B. Pop Count FIFO B (PCB) This field defines the number of entries to be removed from FIFO B.
8:15 FLA_PCA	Fill Level FIFO A (FLA) This field provides the current number of entries in the FIFO A. Pop Count FIFO A (PCA) This field defines the number of entries to be removed from FIFO A.

Note: If the pop count value PCA/PCB is greater than the current FIFO fill level FLB/FLA, then the FIFO is empty after the update. No notification is given that not the required number of entries was removed.

46.5.2.73 ECC Error Interrupt Flag and Enable Register (FR_EEIFER)

Offset: 0x00F0 Access: Read/Write(1)
Write: Normal Mode

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	LRNE_OF	LRCE_OF	DRNE_OF	DRCE_OF	LRNE_IF	LRCE_IF	DRNE_IF	DRCE_IF	0	0	0	0	LRNE_IE	LRCE_IE	DRNE_IE	DRCE_IE
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c								
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. In Functional Test Mode writing 1 toggles the _OF and _IF flags.

Figure 910. ECC Error Interrupt Flag and Enable Register (FR_EEIFER)

This register provides the means to control the ECC-related interrupt request lines and provides the corresponding interrupt flags. The interrupt flags are cleared by writing 1, which resets the corresponding report registers. For a detailed description, see *Section 46.6.24.2*.

Table 877. FR_EEIFER field descriptions

Field	Description
Error Overflow Flags	
0 LRNE_OF	LRAM Non-Corrected Error Overflow Flag This flag is set to 1 when at least one of the following events occurs: a) Memory errors are detected *but not corrected* on CHI LRAM and interrupt flag LRNE_IF is already 1. b) Memory errors are detected *but not corrected* on at least two banks of CHI LRAM. 0 No such event 1 Non-Corrected Error overflow detected on CHI LRAM
1 LRCE_OF	LRAM Corrected Error Overflow Flag This flag is set to 1 when at least one of the following events occurs: a) Memory errors are detected *and corrected* on CHI LRAM and interrupt flag LRCE_IF is already 1. b) Memory errors are detected *and corrected* on at least two banks of CHI LRAM. 0 No such event 1 Corrected Error overflow detected on CHI LRAM **Note:** Error Correction is not implemented on CHI LRAM, so the flag will never be asserted.
2 DRNE_OF	DRAM Non-Corrected Error Overflow Flag This flag is set to 1 when at least one of the following events appears: a) Memory errors are detected *but not corrected* on PE DRAM and interrupt flag DRNE_IF is already 1. b) Memory errors are detected *but not corrected* on at least two banks of the PE DRAM. 0 No such event 1 Non-Corrected Error overflow detected on PE DRAM
3 DRCE_OF	DRAM Corrected Error Overflow Flag This flag is set to 1 when at least one of the following events appears: a) Memory errors are detected *and corrected* on PE DRAM and interrupt flag DRCE_IF is already 1. b) Memory errors are detected *and corrected* on at least two banks of PE DRAM. 0 No such event 1 Corrected Error overflow detected on PE DRAM

Table 877. FR_EEIFER field descriptions (continued)

Field	Description
Error Interrupt Flags	
4 LRNE_IF	LRAM Non-Corrected Error Interrupt Flag This interrupt flag is set to 1 when a memory error is *detected but not corrected* on the CHI LRAM. 0 No such event 1 Non-Corrected Error detected on CHI LRAM
5 LRCE_IF	LRAM Corrected Error Interrupt Flag This interrupt flag is set to 1 when a memory error is *detected and corrected* on the CHI LRAM. 0 No such event 1 Corrected Error detected on CHI LRAM **Note:** Error Correction is not implemented on CHI LRAM, so the flag will never be asserted.
6 DRNE_IF	DRAM Non-Corrected Error Interrupt Flag This interrupt flag is set to 1 when a memory error is *detected but not corrected* on PE DRAM. 0 No such event 1 Non-Corrected Error detected on PE DRAM
7 DRCE_IF	DRAM Corrected Error Interrupt Flag This interrupt flag is set to 1 when a memory error is *detected and corrected* on PE DRAM. 0 No such event 1 Corrected Error detected on PE DRAM
Error Interrupt Enables	
12 LRNE_IE	**LRAM Non-Corrected Error Interrupt Enable** This flag controls if the LRAM Non-Corrected Error Interrupt line is asserted when the LRNE_IF flag is set. 0 Disable interrupt line 1 Enable interrupt line
13 LRCE_IE	**LRAM Corrected Error Interrupt Enable** This flag controls if the LRAM Corrected Error Interrupt line is asserted when the LRCE_IF flag is set. 0 Disable interrupt line 1 Enable interrupt line
14 DRNE_IE	**DRAM Non-Corrected Error Interrupt Enable** This flag controls if the DRAM Non-Corrected Error Interrupt line is asserted when the DRNE_IF flag is set. 0 Disable interrupt line 1 Enable interrupt line
15 DRCE_IE	**DRAM Corrected Error Interrupt Enable** This flag controls if the DRAM Corrected Error Interrupt line is asserted when the DRCE_IF flag is set. 0 Disable interrupt line 1 Enable interrupt line

46.5.2.74 ECC Error Report and Injection Control Register (FR_EERICR)

Offset: 0x00F2

Access: Read/Write
Write: ERS: Anytime
ERM, EIM, EIE: IDL

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	BSY	0	0	0	0	0	ERS		0	0	0	ERM	0	0	EIM	EIE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 911. ECC Error Report and Injection Control Register (FR_EERICR)

This register configures the error injection and error reporting and provides the selector for the content of the report registers.

Table 878. FR_EERICR field descriptions

Field	Description
0 BSY	Register Update Busy This field indicates the current state of the ECC configuration update and controls the register write access condition IDL specified in *Section 46.5.2.2*. 0 ECC configuration is idle 1 ECC configuration is running
6:7 ERS	Error Report Select This field selects the content of the ECC Error reporting registers. 00 Show PE DRAM non-corrected error information 01 Show PE DRAM corrected error information 10 Show CHI LRAM non-corrected error information 11 Show CHI LRAM corrected error information
11 ERM	Error Report Mode This bit configures the type of data written into the internal error report registers on the detection of a memory error. 0 Store data and code as delivered by ECC decoding logic 1 Store data and code as read from the memory
14 EIM	Error Injection Mode This bit configures the ECC error injection mode. 0 Use FR_EEIDR[DATA] and FR_EEICR[CODE] as XOR distortion pattern for error injection 1 Use FR_EEIDR[DATA] and FR_EEICR[CODE] as write value for error injection
15 EIE	Error Injection Enable This bit configures the ECC error injection on the memories. 0 Error injection disabled 1 Error injection enabled

46.5.2.75 ECC Error Report Address Register (FR_EERAR)

Offset: 0x00F4 Access: Read-only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	MID	BANK			ADDR											
W																
Reset	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0

Figure 912. ECC Error Report Address Register (FR_EERAR)

This register provides the memory identifier, bank, and address for which the memory error is reported.

Table 879. FR_EERAR field descriptions

Field	Description
0 MID	Memory Identifier This flag provides the memory instance for which the memory error is reported. 0 PE DRAM 1 CHI LRAM
1:3 BANK	Memory Bank This field provides the BANK for which the memory error is reported. 111 Reset value, indicates no error found after reset. For MID = 0: 000 PE DRAM [7:0] 001 PE DRAM [15:8] Others are not used. For MID = 1: Refer to *Table 880* for the assignment of the LRAM banks.
4:15 ADDR	Memory Address This field provides the address of the failing memory location.

Table 880. LRAM bank value for MID = 1

BANK	Register		
000	FR_MBCCFR(2n)	FR_MBDOR(6n)	FR_LEETR0
001	FR_MBFIDR(2n)	FR_MBDOR(6n + 1)	FR_LEETR1
010	FR_MBIDXR(2n)	FR_MBDOR(6n + 2)	FR_LEETR2
011	FR_MBCCFR(2n+1)	FR_MBDOR(6n + 3)	FR_LEETR3
100	FR_MBFIDR(2n+1)	FR_MBDOR(6n + 4)	FR_LEETR4
101	FR_MBIDXR(2n+1)	FR_MBDOR(6n + 5)	FR_LEETR5
110	Not Used	Not Used	Not Used
111			

46.5.2.76 ECC Error Report Data Register (FR_EERDR)

Offset: 0x00F6 Access: Read-only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DATA															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 913. ECC Error Report Data Register (FR_EERDR)

This register provides the data related information of the reported memory read access. The assignment of the bits depends on the selected memory and memory bank as shown in *Table 882*.

Table 881. FR_EERDR field descriptions

Field	Description
0:15 DATA	Data The content of this field depends on the report mode selected by FR_EERICR[ERM]. ERM = 0: ECC Data, shows data as generated by the ECC decoding logic ERM = 1: Memory Data, shows data as read from the memory

Table 882. Valid bits in FR_EERDR[DATA] / FR_EEIDR[DATA] field

MEM	BANK	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PE DRAM	0									PE DRAM[7:0]							
PE DRAM	1									PE DRAM[15:8]							
CHI LRAM	0	FR_MBCCFR(2n)															
CHI LRAM	0	FR_MBDOR(6n)															
CHI LRAM	0	FR_LEETR0															
CHI LRAM	1						FR_MBFIDR(2n)[FID]										
CHI LRAM	1	FR_MBDOR(6n+1)															
CHI LRAM	1	FR_LEETR1															
CHI LRAM	2										FR_MBIDXR(2n)[MBIDX]						
CHI LRAM	2	FR_MBDOR(6n+2)															
CHI LRAM	2	FR_LEETR2															
CHI LRAM	3	FR_MBCCFR(2n+1)															
CHI LRAM	3	FR_MBDOR(6n+3)															
CHI LRAM	3	FR_LEETR3															
CHI LRAM	4						FR_MBFIDR(2n+1)[FID]										
CHI LRAM	4	FR_MBDOR(6n+4)															
CHI LRAM	4	FR_LEETR4															
CHI LRAM	5										FR_MBIDXR(2n+1)[MBIDX]						

Table 882. Valid bits in FR_EERDR[DATA] / FR_EEIDR[DATA] field (continued)

MEM	BANK	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CHI LRAM	5	FR_MBDOR(6n+5)															
CHI LRAM	5	FR_LEETR5															

46.5.2.77 ECC Error Report Code Register (FR_EERCR)

Offset: 0x00F8 Access: Read-only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	CODE				
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 914. ECC Error Report Code Register (FR_EERCR)

This register provides the ECC-related information of the reported memory read access.

Table 883. FR_EERSR field descriptions

Field	Description
11:15 CODE	Code The content of this field depends on the report mode selected by FR_EERICR[ERM]. ERM = 0: Syndrome. Shows the ECC syndrome generated by the ECC decoding logic. The coding of the PE DRAM syndrome is shown in *Section 46.6.24.2.2*. The coding of the CHI LRAM syndrome is shown in *Section 46.6.24.2.4*. ERM = 1: Checkbits. Shows the ECC checkbits read from the memory.

46.5.2.78 ECC Error Injection Address Register (FR_EEIAR)

Offset: 0x00FA Access: Read/Write
Write: IDL

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	MID		BANK		ADDR											
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 915. ECC Error Injection Address Register (FR_EEIAR)

This register defines the memory module, bank, and address where the ECC error has to be injected.

Table 884. FR_EEIAR field descriptions

Field	Description
0 MID	Memory Identifier This flag defines the memory instance for ECC error injection. 0 PE DRAM 1 CHI LRAM
1:3 BANK	Memory Bank This field defines the memory bank for ECC error injection. For MID = 0: 000 BANK0: PE DRAM [7:0] 001 BANK1: PE DRAM [15:8] Others reserved For MID = 1: Refer to *Table 880* for the assignment of the LRAM banks.
4:15 ADDR	Memory Address This flag defines the memory address for ECC error injection.

46.5.2.79 ECC Error Injection Data Register (FR_EEIDR)

Offset: 0x00FC

Access: Read/Write
Write: IDL

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	DATA															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 916. ECC Error Injection Data Register (FR_EEIDR)

This register defines the data distortion pattern for the error injection write. The number of valid bits depends on the selected memory and memory bank as shown in *Table 882*.

Table 885. FR_EEIDR field descriptions

Field	Description
0:15 DATA	Data The content of this field depends on the error injection mode selected by FR_EERICR[EIM]. EIM = 0: This field defines the XOR distortion pattern for the data written into the memory. EIM = 1: This field defines the data to be written into the memory.

Note: The effect of the error injected depends from the LRAM content at the address accessed and from the module internal usage of the data. Refer to Section 46.6.24.3: Memory error response for details.

46.5.2.80 ECC Error Injection Code Register (FR_EEICR)

Offset: 0x00FE

Access: Read/Write
Write: IDL

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	CODE				
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 917. ECC Error Injection Code Register (FR_EEICR)

This register defines the ECC code distortion pattern for the error injection write.

Table 886. FR_EEICR field descriptions

Field	Description
11:15 CODE	Code The content of this field depends on the error injection mode selected by FR_EERICR[EIM]. EIM = 0: This field defines the XOR distortion pattern for the ECC checkbits written into the memory. EIM = 1: This field defines the ECC checkbits written into the memory.

46.5.2.81 Message Buffer Configuration, Control and Status Registers (FR_MBCCSR*n*)

Offset: 0x0800 + *n**0x8 (*n* = 0 to 63)

Access: Read/Write(1)
Write: MTD: *POC:config* or MB_DIS
CMT: MB_LCK or MB_DIS
EDT, LCKT, MBIE, MBIF: Normal mode

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	MTD	CMT	0	0	MBIE	0	0	0	DUP	DVAL	EDS	LCKS	MBIF
W					rwm	EDT	LCKT									w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. Additional Reset: CMT, DUP, DVAL, MBIF: Message Buffer Disable

Figure 918. Message Buffer Configuration, Control and Status Registers (FR_MBCCSR*n*)

The content of these registers comprises message buffer configuration data, message buffer control data, message buffer status information, and message buffer interrupt flags. A detailed description of all flags can be found in *Section 46.6.6: Individual message buffer functional description*.

If the application writes 1 to the EDT bit, no write access to the other register bits is performed.

If the application writes 0 to the EDT bit and 1 to the LCKT bit, no write access to the other bits is performed.

Table 887. FR_MBCCSR*n* field descriptions

Field	Description
Message Buffer Configuration	
3 MTD	Message Buffer Transfer Direction This bit configures the transfer direction of a message buffer. 0 Receive message buffer 1 Transmit message buffer
Message Buffer Control	
4 CMT	Commit for Transmission This bit indicates if the transmit message buffer data are ready for transmission. 0 Message buffer data not ready for transmission 1 Message buffer data ready for transmission
5 EDT	Enable/Disable Trigger If the application writes 1 to this bit, a message buffer enable or disable is triggered, depending on the current value of the EDS status bit. 0 No effect 1 Message buffer enable or disable is triggered
6 LCKT	Lock/Unlock Trigger If the application writes 1 to this bit and writes 0 to the EDT bit, a message buffer lock or unlock is triggered, depending on the current value of the LCKS status bit. 0 No effect 1 Message buffer lock or unlock is triggered
7 MBIE	Message Buffer Interrupt Enable This control bit defines whether the message buffer will generate an interrupt request when its MBIF flag is set. 0 Interrupt request generation disabled 1 Interrupt request generation enabled
Message Buffer Status	
11 DUP	Data Updated This status bit indicates whether the frame header in the message buffer header field and the data in the message buffer data field were updated after a frame reception. 0 Frame Header and message buffer data field not updated 1 Frame Header and message buffer data field updated
12 DVAL	Data Valid For receive message buffers this status bit indicates whether the message buffer data field contains valid frame data. For transmit message buffers, the status bit indicates if a message is transferred again due to the state transmission mode of the message buffer. 0 Receive message buffer contains no valid frame data / message is transmitted for the first time. 1 Receive message buffer contains valid frame data / message will be transferred again.
13 EDS	Enable/Disable Status This status bit indicates whether the message buffer is enabled or disabled. 0 Message buffer is disabled 1 Message buffer is enabled

Table 887. FR_MBCCSR*n* field descriptions (continued)

Field	Description
14 LCKS	Lock Status This status bit indicates the current lock status of the message buffer. 0 Message buffer is not locked by the application 1 Message buffer is locked by the application
15 MBIF	Message Buffer Interrupt Flag This flag is set when the slot status field of the message buffer was updated after frame transmission or reception, or when a transmit message buffer was just enabled by the application. 0 No such event 1 Slot status field updated or transmit message buffer just enabled In Functional Test Mode writing a 1 toggles the interrupt flag

46.5.2.82 Message Buffer Cycle Counter Filter Registers (FR_MBCCFR*n*)

Offset: 0x0802 +*n**0x8 (*n* = 0 to 63)

Access: Read/Write(1)
Write: *POC:config* or MB_DIS

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	MTM	CHA	CHB	CCFE	CCFMSK								CCFVAL			
Reset	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

1. 16-bit write access required

Figure 919. Message Buffer Cycle Counter Filter Registers (FR_MBCCFR*n*)

This register contains message buffer configuration data for the transmission mode, the channel assignment, and for the cycle counter filtering. For detailed information on cycle counter filtering, refer to *Section 46.6.7.1: Message buffer cycle counter filtering*.

Table 888. FR_MBCCFR*n* field descriptions

Field	Description
0 MTM	Message Buffer Transmission Mode This control bit applies only to transmit message buffers and defines the transmission mode. 0 Event Transmission mode 1 State Transmission mode
1 CHA	Channel Assignment This control bits defines the channel assignment and control the receive and transmit behavior of the message buffer according to *Table 889*.
2 CHB	Channel Assignment This control bits defines the channel assignment and control the receive and transmit behavior of the message buffer according to *Table 889*.
3 CCFE	Cycle Counter Filtering Enable This control bit is used to enable and disable the cycle counter filtering. 0 Cycle counter filtering disabled 1 Cycle counter filtering enabled

Table 888. FR_MBCCFR*n* field descriptions (continued)

Field	Description
4:9 CCFMSK	Cycle Counter Filtering Mask This field defines the filter mask for the cycle counter filtering.
10:15 CCFVAL	Cycle Counter Filtering Value This field defines the filter value for the cycle counter filtering.

Table 889. Channel assignment description

CHA	CHB	Transmit message buffer		Receive message buffer	
		Static segment	Dynamic segment	Static segment	Dynamic segment
1	1	transmit on both channel A and channel B	transmit on channel A only	store first valid frame received on either channel A or channel B	store first valid frame received on channel A, ignore channel B
0	1	transmit on channel B	transmit on channel B	store first valid frame received on channel B	store first valid frame received on channel B
1	0	transmit on channel A	transmit on channel A	store first valid frame received on channel A	store first valid frame received on channel A
0	0	no frame transmission	no frame transmission	no frame stored	no frame stored

Note: If at least one message buffer assigned to a certain slot is assigned to both channels, then all message buffers assigned to this slot have to be assigned to both channels. Otherwise, the message buffer configuration is illegal and the result of the message buffer search is not defined.

46.5.2.83 Message Buffer Frame ID Registers (FR_MBFIDR*n*)

Offset: 0x0804 +*n**0x8 (*n* = 0 to 63)

Access: Read/Write(1)
Write: *POC:config* or MB_DIS

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

1. 16-bit write access required

Figure 920. Message Buffer Frame ID Registers (FR_MBFIDR*n*)

Table 890. FR_MBFIDR*n* field descriptions

Field	Description
5:15 FID	Frame ID The semantic of this field depends on the message buffer transfer type. – *Receive Message Buffer:* This field is used as a filter value to determine if the message buffer is used for reception of a message received in a slot with the slot ID equal to FID. – *Transmit Message Buffer:* This field is used to determine the slot in which the message in this message buffer should be transmitted.

46.5.2.84 Message Buffer Index Registers (FR_MBIDXR*n*)

Offset: 0x0806 +*n**0x8 (*n* = 0 to 63)

Access: Read/Write(1)
Write: *POC:config* or MB_DIS

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

1. 16-bit write access required

Figure 921. Message Buffer Index Registers (FR_MBIDXR*n*)

Table 891. FR_MBIDXR*n* field descriptions

Field	Description
9:15 MBIDX	Message Buffer Index This field provides the index of the message buffer header field of the physical message buffer that is currently associated with this message buffer. The application writes the index of the initially associated message buffer header field into this register. The CC updates this register after frame reception or transmission. Legal Values are 0 ≤ i ≤ 67. Illegal values will be detected during the message buffer search.

46.5.2.85 Message Buffer Data Field Offset Registers (FR_MBDORn)

Offset: 0x1000 + *n**0x2 (n = 0 to 67)

Access: Read/Write(1)
Write: *POC:config*

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

1. 16-bit write access required

Figure 922. Message Buffer Data Field Offset Registers (FR_MBDORn)

Table 892. FR_MBDORn field descriptions

Field	Description
0:15 MBDO	Message Buffer Data Field Offset This field provides the data field offset belonging to a particular Message Buffer Index. For configuration constraints see *Section 46.7.1.2: Configure data field offsets*.

46.5.2.86 LRAM ECC Error Test Registers (FR_LEETR*n*)

Offset: 0x1090 + *n**0x2 (n = 0 to 5)

Access: Read/Write(1)
Write: Anytime

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

1. 16-bit write access required

Figure 923. LRAM ECC Error Test Registers (FR_LEETR*n*)

Table 893. FR_LEETR*n* field descriptions

Field	Description
0:15 LEETD	LRAM ECC Error Test Data This field contains the LRAM data belonging to the test register located in LRAM Bank n.

46.6 Functional description

This section provides a detailed description of the functionality implemented in the CC.

46.6.1 Message buffer concept

The CC uses a data structure called a *message buffer* to store frame data, configuration, control, and status data. Each message buffer consists of two parts: the *message buffer control data* and the *physical message buffer.* The message buffer control data is located in dedicated registers. The structure of the message buffer control data depends on the message buffer type, described in *Section 46.6.3: Message Buffer Types*. The physical message buffer is located in the FlexRay memory area and is described in *Section 46.6.2: Physical message buffer*.

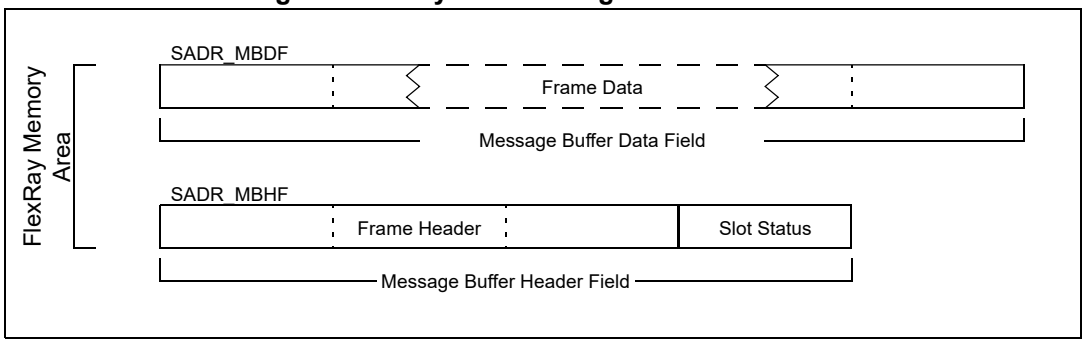
46.6.2 Physical message buffer

All FlexRay messages and related frame and slot status information of received frames and of frames to be transmitted to the FlexRay bus are stored in data structures called *physical message buffers*. The physical message buffers are located in the FlexRay memory area.The structure of a physical message buffer is depicted in *Figure 923*.

A physical message buffer consists of two fields, the *message buffer header field* and the *message buffer data field*. The message buffer header field contains the *frame header* and the *slot status*. The message buffer data field contains the *frame data*.

The connection between the two fields is established by the *data field offset*.

Figure 924. Physical message buffer structure



46.6.2.1 Message buffer header field

The message buffer header field is a contiguous region in the FlexRay memory area that occupies eight bytes. It contains the frame header and the slot status. Its structure is shown in *Figure 924*. The physical start address SADR_MBHF of the message buffer header field must be 16-bit aligned.

46.6.2.1.1 Frame header

The frame header occupies the first six bytes in the message buffer header field. It contains all FlexRay frame header-related information according to the *FlexRay Communications System Protocol Specification, Version 2.1 Rev A*. A detailed description of the usage and the content of the frame header is provided in *Section 46.6.5.2.1: Frame header description*.

46.6.2.1.2 Slot status

The slot status occupies the last 2 bytes of the message buffer header field. It provides the slot and frame status-related information according to the *FlexRay Communications System Protocol Specification, Version 2.1 Rev A*. A detailed description of the content and usage of the slot status is provided in *Section 46.6.5.2.2: Slot status description*.

46.6.2.2 Message buffer data field

The message buffer data field is a contiguous area of 2-byte entities. This field contains the frame payload data, or a part of it, of the frame to be transmitted to or received from the FlexRay bus. The minimum length of this field depends on the specific message buffer configuration and is specified in the message buffer descriptions given in *Section 46.6.3: Message Buffer Types*.

46.6.3 Message Buffer Types

The CC provides three different types of message buffers.

- Individual Message Buffers
- Receive Shadow Buffers
- Receive FIFO Buffers

For each message buffer type, the structure of the physical message buffer is identical. The message buffer types differ only in the structure and content of message buffer control data, which control the related physical message buffer. The message buffer control data are described in the following sections.

46.6.3.1 Individual Message Buffers

The individual message buffers are used for all types of frame transmission and for dedicated frame reception based on individual filter settings for each message buffer. The CC supports three types of individual message buffers, which are described in *Section 46.6.6: Individual message buffer functional description*.

Each individual message buffer consists of two parts, the physical message buffer, which is located in the FlexRay memory area, and the message buffer control data, which are located in dedicated registers. The structure of an individual message buffer is given in *Figure 925*.

Each individual message buffer has a message buffer number *n* assigned, which determines the set of message buffer control registers associated to this individual message buffer. The individual message buffer with message buffer number *n* is controlled by the registers FR_MBCCSR*n*, FR_MBCCFR*n*, FR_MBFIDR*n* and FR_MBIDXR*n*.

The connection between the message buffer control registers and the physical message buffer is established by the message buffer index field, MBIDX, in the Message Buffer Index Registers (FR_MBIDXRn). The start address SADR_MBHF of the related message buffer header field in the FlexRay memory area is determined according to *Equation 41*.

Equation 41 **SADR_MBHF = (FR_MBIDXR*n*[MBIDX] × 8) + SMBA**

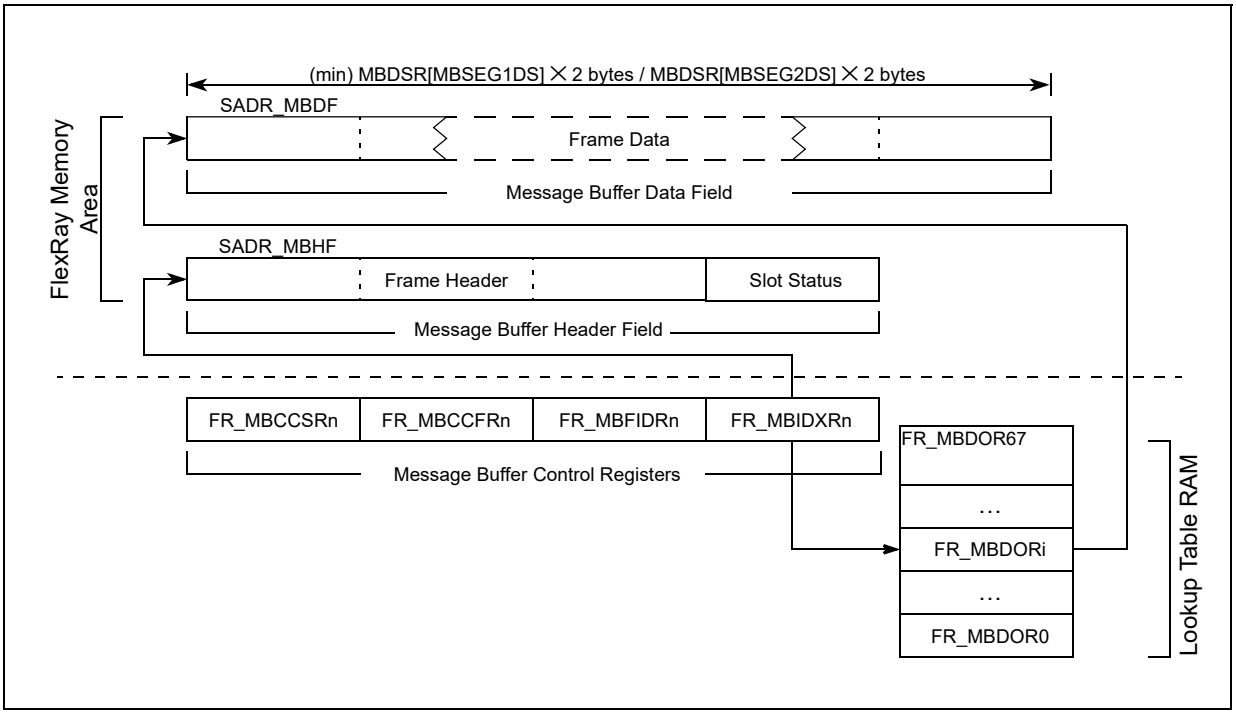
The data field belonging to a particular physical message buffer is characterized by the data field offset. For each physical message buffer with MBIDX i, the FR_MBDORi contains the offset of the corresponding message buffer data field with respect to the CC FlexRay memory area base address as provided by SMBA field in the System Memory Base Address Register (FR_SYMBADR).

The data field offset is used to determine the start address SADR_MBDF of the corresponding message buffer data field in the FlexRay memory area according to *Equation 42*.

Equation 42 **SADR_MBDF = [Data Field Offset] + SMBA**

The FR_MBDOR*n* are stored in the module internal memory LRAM. Refer to *Section 46.7.2.3: CHI LRAM initialization* for the setup of the data field offset values.

Figure 925. Individual message buffer structure



46.6.3.1.1 Individual message buffer segments

The set of the individual message buffers can be split up into two message buffer segments using the Message Buffer Segment Size and Utilization Register (FR_MBSSUTR). All individual message buffers with a message buffer number n ≤ FR_MBSSUTR[LAST_MB_SEG1] belong to the first message buffer segment. All individual message buffers with a message buffer number n > FR_MBSSUTR[LAST_MB_SEG1] belong to the second message buffer segment. The following rules apply to the length of the message buffer data field:

- All physical message buffers associated to individual message buffers that belong to the same message buffer segment must have message buffer data fields of the same length.
- The minimum length of the message buffer data field for individual message buffers in the first message buffer segment is 2 × FR_MBDSR[MBSEG1DS] bytes.
- The minimum length of the message buffer data field for individual message buffers assigned to the second segment is 2 × FR_MBDSR[MBSEG2DS] bytes.

46.6.3.2 Receive shadow buffers

The receive shadow buffers are required for the individual message buffers' frame reception process. The CC provides four receive shadow buffers: one receive shadow buffer per channel and per message buffer segment.

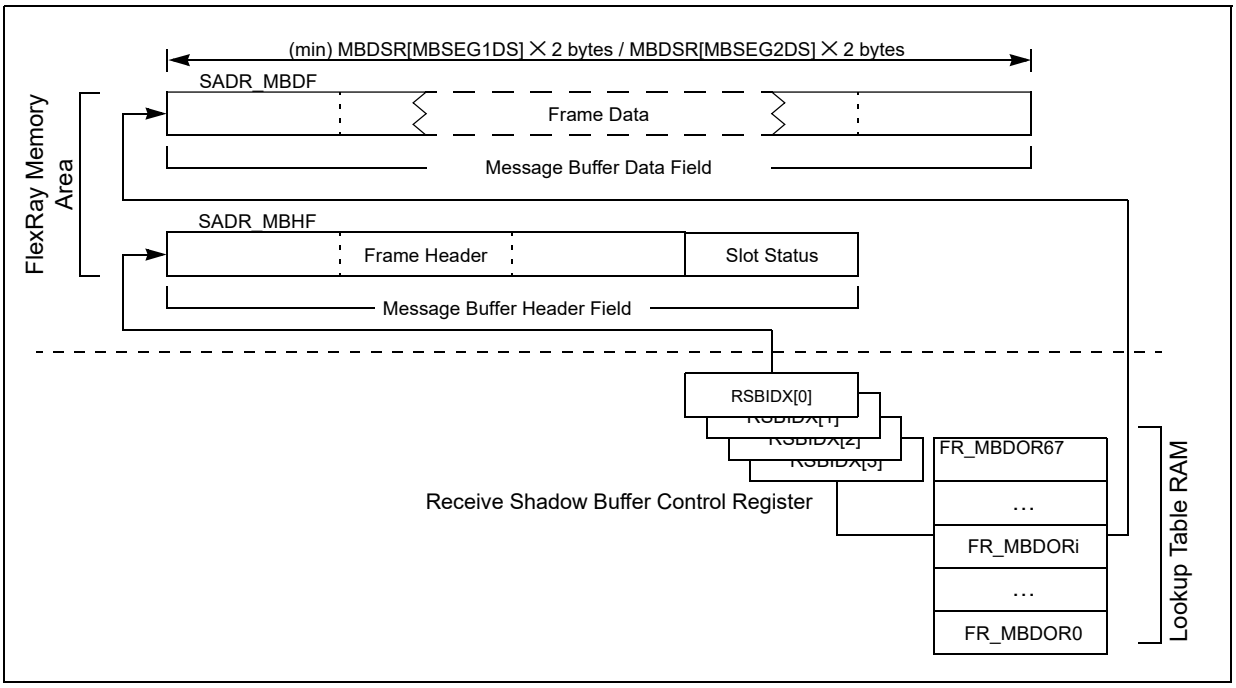
Each receive shadow buffer consists of two parts: the physical message buffer located in the FlexRay memory area and the receive shadow buffer control registers located in dedicated registers. The structure of a receive shadow buffer is shown in *Figure 926*. The four internal shadow buffer control registers can be accessed by the Receive Shadow Buffer Index Register (FR_RSBIR).

The connection between the receive shadow buffer control register and the physical message buffer for the selected receive shadow buffer is established by the receive shadow buffer index field RSBIDX in the Receive Shadow Buffer Index Register (FR_RSBIR). The start address SADR_MBHF of the related message buffer header field in the FlexRay memory area is determined according to *Equation 43*.

Equation 43 $$SADR_MBHF = (FR_RSBIR[RSBIDX] \times 8) + SMBA$$

The length required for the message buffer data field depends on the message buffer segment that the receive shadow buffer is assigned to. For the receive shadow buffers assigned to the first message buffer segment, the length must be the same as for the individual message buffers assigned to the first message buffer segment. For the receive shadow buffers assigned to the second message buffer segment, the length must be the same as for the individual message buffers assigned to the second message buffer segment. The receive shadow buffer assignment is described in Receive Shadow Buffer Index Register (FR_RSBIR).

Figure 926. Receive shadow buffer structure



46.6.3.3 Receive FIFO

The receive FIFO implements a frame reception system based on the FIFO concept. The CC provides two independent receive FIFOs, one per channel.

A receive FIFO consists of a set of physical message buffers in the FlexRay memory area and a set of receive FIFO control registers located in dedicated registers. The structure of a receive FIFO is given in *Figure 927*.

The connection between the receive FIFO control registers and the set of physical message buffers is established by the Receive FIFO Start Index Register (FR_RFSIR), the Receive FIFO Depth and Size Register (FR_RFDSR), and the Receive FIFO A Read Index Register (FR_RFARIR)/Receive FIFO B Read Index Register (FR_RFBRIR).

The system memory base address SMBA valid for the receive FIFOs is defined by the system memory base address register selected by the FIFO address mode bit FR_MCR[FAM], refer to *Section 46.5.2.4: Module Configuration Register (FR_MCR)*.

The start byte address SADR_MBHF[1] of the first message buffer header field that belongs to the receive FIFO is determined according to *Equation 44*.

Equation 44 SADR_MBHF[1] = (8 × FR_RFSIR[SIDX]) + SMBA

The start byte address SADR_MBHF[n] of the last message buffer header field that belongs to the receive FIFO in the FlexRay memory area is determined according to *Equation 45*.

Equation 45 SADR_MBHF[n] =
(8 × (FR_RFSIR[SIDX] + FR_RFDSR[FIFO_DEPTH])) + SMBA

The required information to access the current entry of the FIFO is given in the following registers:

- The registers Receive FIFO A Read Index Register (FR_RFARIR) and Receive FIFO B Read Index Register (FR_RFBRIR) provide the index of the physical message buffer belonging to the current entry.
- The data field offset belonging to the current FIFO entry RF_DFO[X] must be calculated using the current read index i according to the following formula:

Equation 46 RF_DFO[X] =
FR_RFSDOR[X] + (FR_RFDSR[X][ENTRY_SIZE] × 2) × (i - R_RFSIR[SIDX])

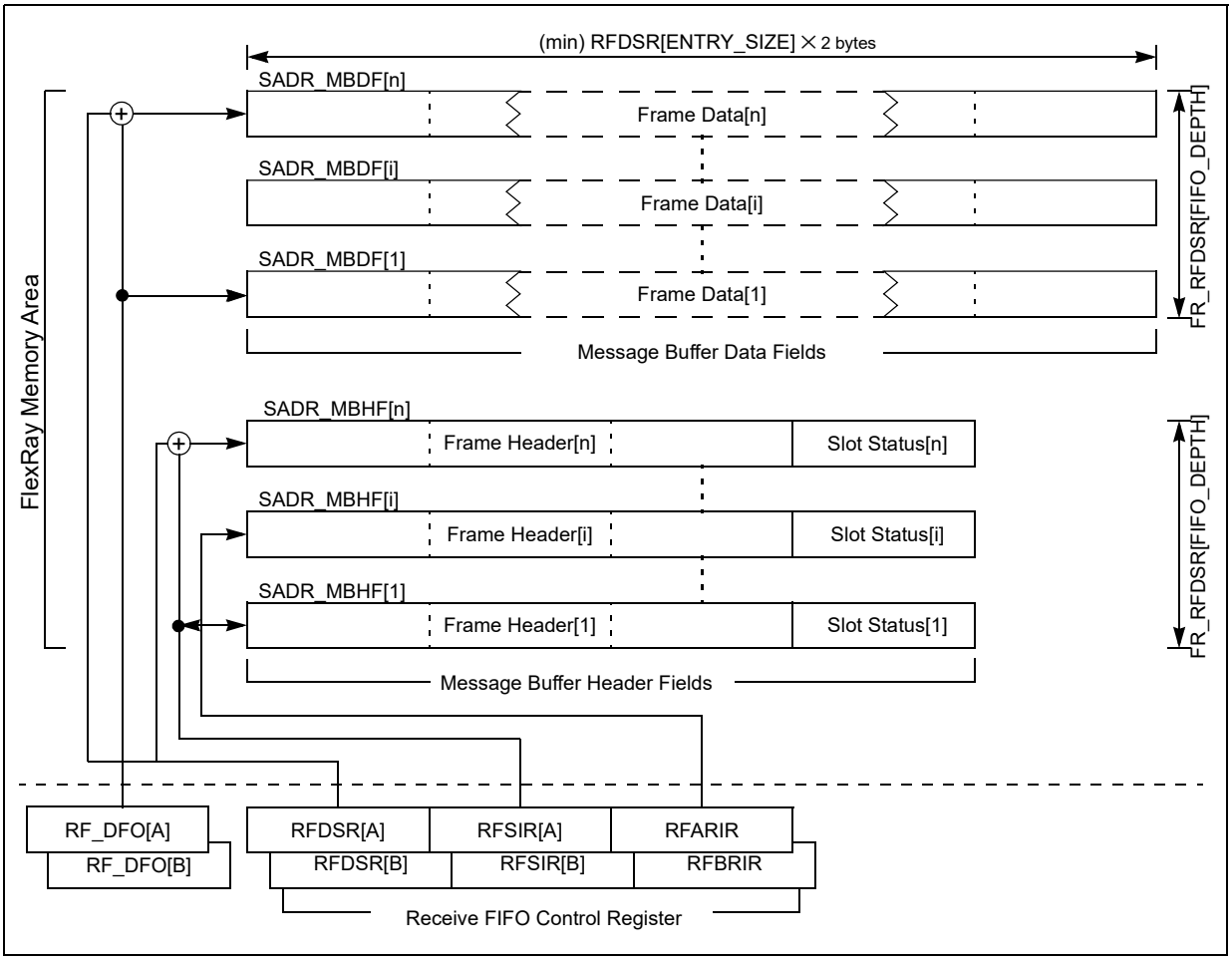
Note: The current read index loops up starting at the number given in the FR_RD[A/B]RDIDX register for the required number of entries.

Refer to Section 46.6.9.8: FIFO update for details about updating the FIFO read pointer.

All message buffer header fields assigned to a receive FIFO are within a contiguous region defined by FR_RFSIR[SIDX] and FR_RFDSR[FIFO_DEPTH].

The data sections of all FIFO entries within on receive FIFO are of the same length defined by FR_RFDSR[FIFO_SIZE].

Figure 927. Receive FIFO structure



Note: The actual values of the data field offsets RF_DFO[A/B] need to be calculated according to Equation 46. They are not stored in a register.

46.6.3.4 Message buffer configuration and control data

This section describes the configuration and control data for each message buffer type.

46.6.3.4.1 Individual message buffer configuration data

Before an individual message buffer can be used for transmission or reception, it must be configured. There is a set of common configuration parameters that applies to all individual message buffers and a set of configuration parameters that applies to each message buffer individually.

46.6.3.4.2 Common configuration data

The set of common configuration data for individual message buffers is located in the following registers.

- Message Buffer Data Size Register (FR_MBDSR)
  The MBSEG2DS and MBSEG1DS fields define the minimum length of the message buffer data field with respect to the message buffer segment.
- Message Buffer Segment Size and Utilization Register (FR_MBSSUTR)
  The LAST_MB_SEG1 and LAST_MB_UTIL fields define the segmentation of the individual message buffers and the number of individual message buffers that are used. For more details, see *Section 46.6.3.1.1: Individual message buffer segments*.

46.6.3.4.3 Specific configuration data

The set of message buffer specific configuration data for individual message buffers is located in the following registers.

- Message Buffer Configuration, Control, Status Registers (FR_MBCCSR*n*)
  The MTD bit configures the message buffer type.
- Message Buffer Cycle Counter Filter Registers (FR_MBCCFR*n*)
  The MTM, CHA, CHB bits configure the transmission mode and the channel assignment. The CCFE, CCFMSK, and CCFVAL bits and fields configure the cycle counter filter.
- Message Buffer Frame ID Registers (FR_MBFIDR*n*)
  For a transmit message buffer, the FID field is used to determine the slot in which the message in this message buffer will be transmitted.
- Message Buffer Index Registers (FR_MBIDXR*n*)
  This MBIDX field provides the index of the message buffer header field of the physical message buffer that is currently associated with this message buffer.

46.6.3.5 Individual message buffer control data

During normal operation, each individual message buffer can be controlled by the CMT, LCKT, EDT, and MBIE control and trigger bits in the Message Buffer Configuration, Control, Status Registers (FR_MBCCSR*n*).

46.6.3.6 Receive shadow buffer configuration data

Before the individual message buffers can receive frames, the receive shadow buffers must be configured. The configuration data are provided by the Receive Shadow Buffer Index Register (FR_RSBIR). For each receive shadow buffer, the application provides the message buffer header index. When the protocol is in the *POC:normal active* or *POC:normal passive* state, the receive shadow buffers are under full CC control.

46.6.3.7 Receive FIFO control and configuration data

This section describes the configuration and control data for the two receive FIFOs.

46.6.3.7.1 Receive FIFO configuration data

The CC provides two functional independent receive FIFOs, one per channel. The FIFOs have a common subset of configuration data:

- Receive FIFO Periodic Timer Register (FR_RFPTR)

Each FIFO has its own set of configuration data. The configuration data are located in the following registers:

- Receive FIFO Watermark and Selection Register (FR_RFWMSR)
- Receive FIFO Start Index Register (FR_RFSIR)
- Receive FIFO Start Data Offset Register (FR_RFSDOR)
- Receive FIFO Depth and Size Register (FR_RFDSR)
- Receive FIFO Message ID Acceptance Filter Value Register (FR_RFMIDAFVR)
- Receive FIFO Message ID Acceptance Filter Mask Register (FR_RFMIDAFMR)
- Receive FIFO Frame ID Rejection Filter Value Register (FR_RFFIDRFVR)
- Receive FIFO Frame ID Rejection Filter Mask Register (FR_RFFIDRFMR)
- Receive FIFO Range Filter Configuration Register (FR_RFRFCFR)

46.6.3.7.2 Receive FIFO control data

The application can access the FIFOs at any time using the control bits in the following registers:

- Global Interrupt Flag and Enable Register (FR_GIFER)
- Receive FIFO Fill Level and POP Count Register (FR_RFFLPCR)

46.6.3.7.3 Receive FIFO status data

The current status of the receive FIFO is provided in the following registers:

- Global Interrupt Flag and Enable Register (FR_GIFER)
- Receive FIFO A Read Index Register (FR_RFARIR)
- Receive FIFO B Read Index Register (FR_RFBRIR)
- Receive FIFO Fill Level and POP Count Register (FR_RFFLPCR)

46.6.4 FlexRay memory area layout

The CC supports a wide range of possible layouts for the FlexRay memory area. Two basic layout modes can be selected by the FIFO address mode bit FR_MCR[FAM].

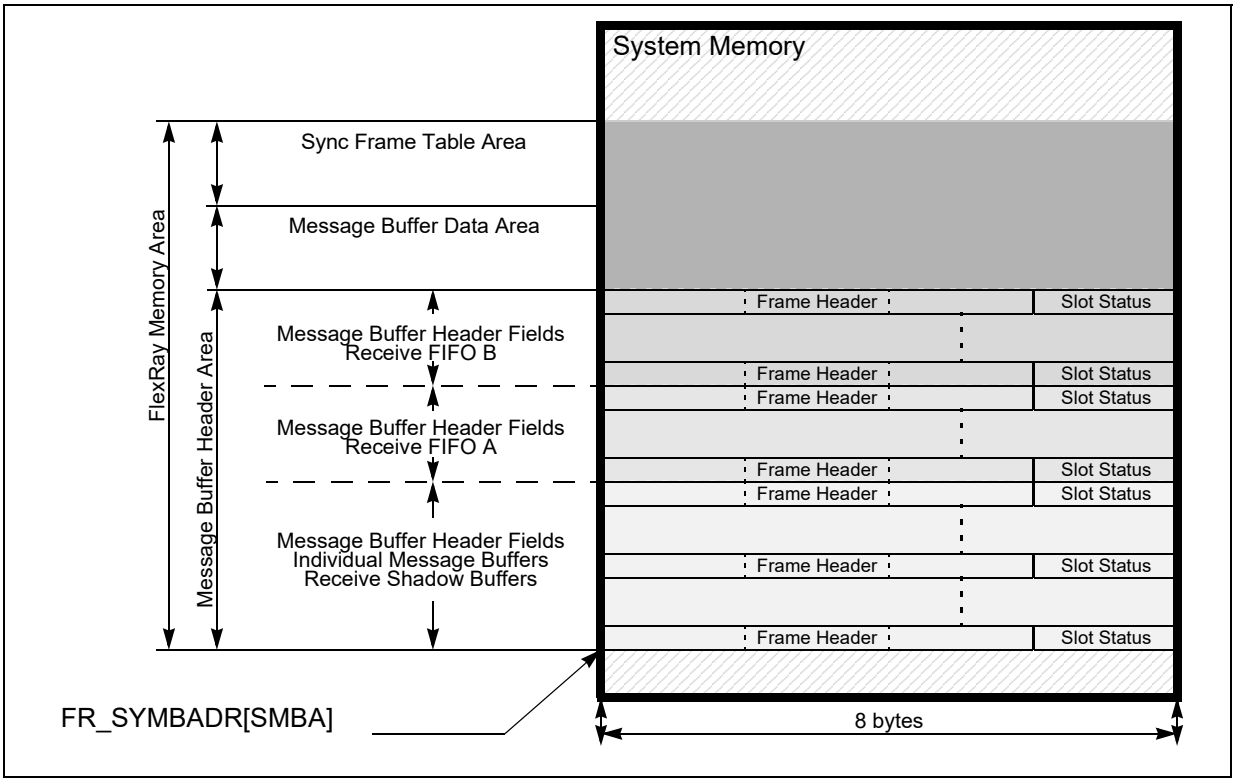
46.6.4.1 FlexRay memory area layout (FR_MCR[FAM] = 0)

Figure 928 shows an example layout for the FIFO address mode FR_MCR[FAM] = 0. In this mode, the following set of rules applies to the layout of the FlexRay memory area:

- The FlexRay memory area is one contiguous region.
- The FlexRay memory area size is a maximum of 64 KB.
- The FlexRay memory area starts at a 16 byte boundary.

The FlexRay memory area contains three areas: the *message buffer header area*, the *message buffer data area*, and the *sync frame table area*.

Figure 928. Example of FlexRay memory area layout (FR_MCR[FAM] = 0)

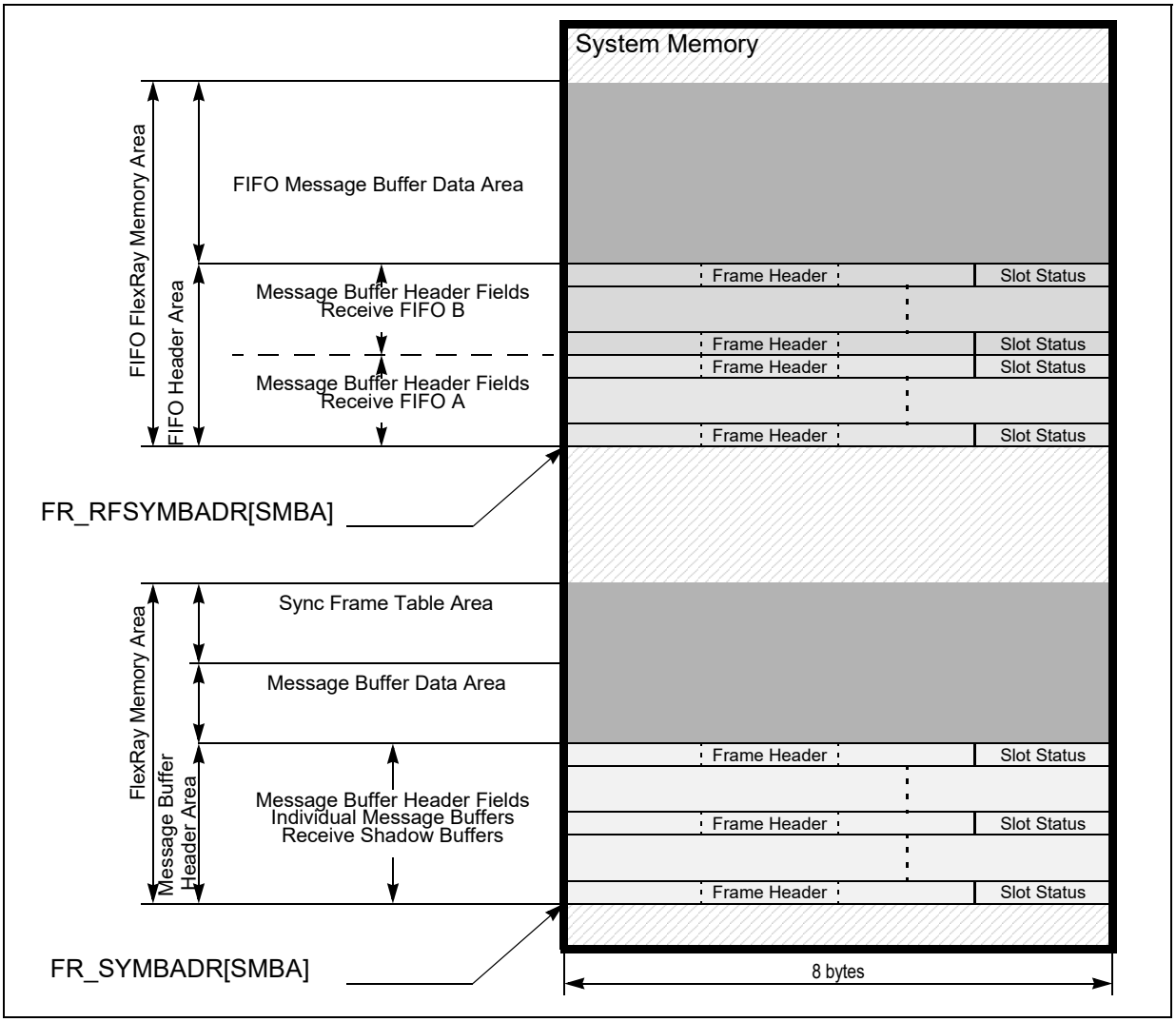


46.6.4.2 FlexRay memory area layout (FR_MCR[FAM] = 1)

Figure 929 shows an example layout for the FIFO address mode FR_MCR[FAM] = 1. The following set of rules applies to the layout of the FlexRay memory area:

- The FlexRay memory area consists of two contiguous regions.
- The size of each region is maximum 64 KB.
- Each region start at a 16 byte boundary.

Figure 929. AreaExample of FlexRay memory area layout (FR_MCR[FAM] = 1)



46.6.4.3 Message buffer header area (FR_MCR[FAM] = 0)

The message buffer header area contains all message buffer header fields of the physical message buffers for all message buffer types. The following rules apply to the message buffer header fields for the three type of message buffers.

1. The start byte address SADR_MBHF of each message buffer header field for individual message buffers and receive shadow buffers must fulfill *Equation 47*.

Equation 47 $\quad SADR_MBHF = (i \times 8) + FR_SYMBADR[SMBA];\ (0 \leq i \leq 67)$

2. The start byte address SADR_MBHF of each message buffer header field for the FIFO must fulfill *Equation 48*.

Equation 48 $\quad SADR_MBHF = (i \times 8) + FR_SYMBADR[SMBA];\ (0 \leq i \leq 1023)$

3. The message buffer header fields for each FIFO must be a contiguous area.

46.6.4.4 Message buffer header area (FR_MCR[FAM] = 1)

The message buffer header area contains all message buffer header fields of the physical message buffers for the individual message buffers and receiver shadow buffers. The following rules apply to the message buffer header fields for the two type of message buffers.

1. The start address SADR_MBHF of each message buffer header field for individual message buffers and receive shadow buffers must fulfill *Equation 49*.

Equation 49 $$SADR_MBHF = (i \times 8) + FR_SYMBADR[SMBA];\ (0 \leq i \leq 67)$$

46.6.4.5 FIFO message buffer header area (FR_MCR[FAM] = 1)

The FIFO message buffer header area contains all message buffer header fields of the physical message buffers for the FIFO. The following rules apply to the FIFO message buffer header fields.

1. The start byte address SADR_MBHF of each message buffer header field for the FIFO must fulfill .

$$SADR_MBHF = (i \times 8) + FR_RFSYMBADR[SMBA];\ (0 \leq i \leq 1023)$$

2. The message buffer header fields for each FIFO have to be a contiguous area.

46.6.4.6 Message buffer data area

The message buffer data area contains all the message buffer data fields of the physical message buffers. Each message buffer data field must start at a 16-bit boundary.

46.6.4.7 Sync frame table area

The sync frame table area is used to provide a copy of the internal sync frame tables for application access. Refer to *Section 46.6.12: Sync frame ID and sync frame deviation tables* for the description of the sync frame table area.

46.6.5 Physical message buffer description

This section provides a detailed description of the usage and the content of the two parts of a physical message buffer: the message buffer header field and the message buffer data field.

46.6.5.1 Message buffer protection and data consistency

The physical message buffers are located in the FlexRay memory area. The CC provides no means to protect the FlexRay memory area from uncontrolled or illegal host or other client write access. To ensure data consistency of the physical message buffers, the application must follow the write access scheme that is given in the description of each of the physical message buffer fields.

46.6.5.2 Message buffer header field description

This section provides a detailed description of the usage and content of the message buffer header field. A description of the structure of the message buffer header fields is given in *Section 46.6.2.1: Message buffer header field*. Each message buffer header field consists of two sections: the frame header section and the slot status section.

46.6.5.2.1 Frame header description

46.6.5.2.1.1 Frame header content

The semantic and content of the frame header section depends on the message buffer type.

For individual receive message buffers and receive FIFOs, the frame header receives the frame header data of the *first valid frame* received on the assigned channels.

For receive shadow buffers, the frame header receives the frame header data of the current frame received regardless of whether the frame is valid or not.

For transmit message buffers, the application writes the frame header of the frame to be transmitted into this location. The frame header will be read out when the frame is transferred to the FlexRay bus.

The structure of the frame header in the message buffer header field for receive message buffers and the receive FIFO is given in *Figure 930*. A detailed description is given in *Table 895*.

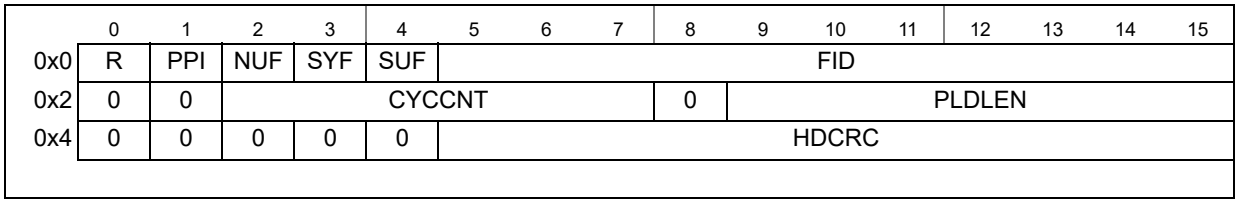


Figure 930. Frame header structure (receive message buffer and receive FIFO)

The structure of the frame header in the message buffer header field for transmit message buffers is given in *Figure 931*. A detailed description is given in *Table 896*. The checks that will be performed are described in Frame Header Checks.

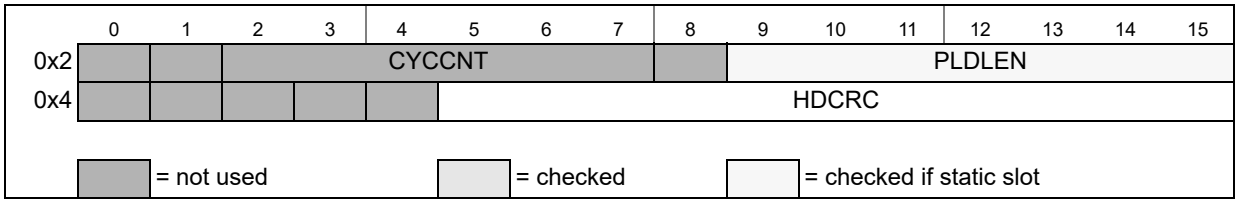


Figure 931. Frame header structure (transmit message buffer)

The structure of the frame header in the message buffer header field for transmit message buffers assigned to key slot is given in *Figure 932*.

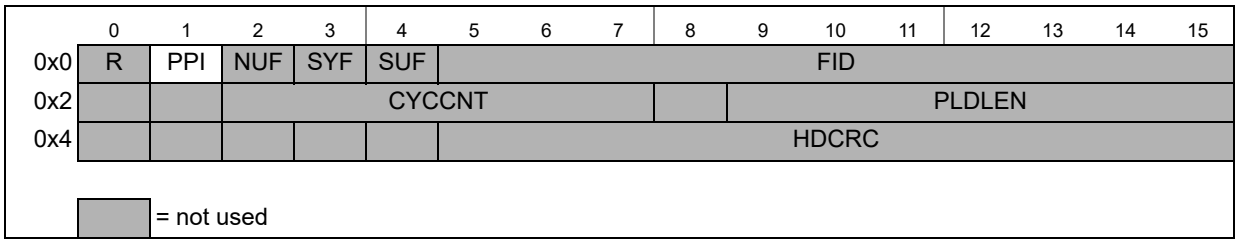


Figure 932. Frame header structure (transmit message buffer for key slot)

46.6.5.2.1.2 Frame Header Access

The frame header is located in the FlexRay memory area. To ensure data consistency, the application must follow the write access scheme described below.

For receive message buffers, receive shadow buffers, and receive FIFOs, the application must not write to the frame header field.

For transmit message buffers, the application must follow the write access restrictions given in *Table 894*. This table shows the condition under which the application can write to the frame header entries without corrupting the FlexRay message transmission.

Table 894. Frame header write access constraints (transmit message buffer)

Field	Static segment	Dynamic segment
FID	*POC:config* or MB_DIS	
PPI, PLDLEN, HDCRC	*POC:config* or MB_DIS or	
		MB_LCK

46.6.5.2.1.3 Frame Header Checks

As shown in *Figure 931* and *Figure 932*, not all fields in the message buffer frame header are used for transmission. Some fields in the message buffer frame header are ignored, some are used for transmission, and some of them are checked for correct values. All checks that will be performed are described below.

For message buffers assigned to the key slot, no checks will be performed.

The value of the FID field must be equal to the value of the corresponding Message Buffer Frame ID Registers (FR_MBFIDR*n*). If the CC detects a mismatch while transmitting the frame header, it will set the frame ID error flag, FID_EF, in the CHI Error Flag Register (FR_CHIERFR). The value of the FID field will be ignored and replaced by the value provided in the Message Buffer Frame ID Registers (FR_MBFIDR*n*).

For transmit message buffers assigned to the *static* segment, the PLDLEN value must be equal to the value of the payload_length_static field in the Protocol Configuration Register 19 (FR_PCR19). If this is not fulfilled, the static payload length error flag SPL_EF in the CHI Error Flag Register (FR_CHIERFR) is set when the message buffer is under transmission. A syntactically and semantically correct frame is generated with payload_length_static payload words and the payload length field in the transmitted frame header set to payload_length_static.

For transmit message buffers assigned to the *dynamic* segment, the PLDLEN value must be less than or equal to the value of the max_payload_length_dynamic field in the Protocol Configuration Register 24 (FR_PCR24). If this is not fulfilled, the dynamic payload length error flag DPL_EF in the CHI Error Flag Register (FR_CHIERFR) is set when the message buffer is under transmission. A syntactically and semantically correct dynamic frame is generated with PLDLEN payload words and the payload length field in the frame header set to PLDLEN.

Table 895. Frame header field descriptions (receive message buffer and receive FIFO)

Field	Description
R	Reserved Bit This is the value of the *Reserved bit* of the received frame stored in the message buffer.
PPI	Payload Preamble Indicator This is the value of the *Payload Preamble Indicator* of the received frame stored in the message buffer.
NUF	Null Frame Indicator This is the value of the *Null Frame Indicator* of the received frame stored in the message buffer.
SYF	Sync Frame Indicator This is the value of the *Sync Frame Indicator* of the received frame stored in the message buffer.
SUF	Startup Frame Indicator This is the value of the *Startup Frame Indicator* of the received frame stored in the message buffer.
FID	Frame ID This is the value of the *Frame ID* field of the received frame stored in the message buffer.
CYCCNT	Cycle Count This is the number of the communication cycle in which the frame stored in the message buffer was received.
PLDLEN	Payload Length This is the value of the *Payload Length* field of the received frame stored in the message buffer.
HDCRC	Header CRC This is the value of the *Header CRC* field of the received frame stored in the message buffer.

Table 896. Frame header field descriptions (transmit message buffer)

Field	Description
R	Reserved Bit This bit is not used. The value of the Reserved bit is generated internally according to FlexRay Communications System Protocol Specification, Version 2.1 Rev A.
PPI	Payload Preamble Indicator This bit provides the value of the Payload Preamble Indicator for the frame transmitted from the message buffer.
NUF	Null Frame Indicator This bit is not used, the value of the Null Frame Indicator is generated internally according to FlexRay Communications System Protocol Specification, Version 2.1 Rev A.

Table 896. Frame header field descriptions (transmit message buffer) (continued)

Field	Description
SYF	Sync Frame Indicator This bit is not used. The value of the Sync Frame Indicator is generated internally according to FlexRay Communications System Protocol Specification, Version 2.1 Rev A.
SUF	Startup Frame Indicator This bit is not used. The value of the Startup Frame Indicator is generated internally according to FlexRay Communications System Protocol Specification, Version 2.1 Rev A.
FID	Frame ID This field is checked as described in Frame Header Checks.
CYCCNT	Cycle Count This field is not used, the value of the transmitted Cycle Count field is taken from the internal communication cycle counter.
PLDLEN	Payload Length This field is checked and used as described in Frame Header Checks.
HDCRC	Header CRC This field provides the value of the Header CRC field for the frame transmitted from the message buffer.

46.6.5.2.2 Slot status description

The slot status is a read-only structure for the application and a write-only structure for the CC. The meaning and content of the slot status in the message buffer header field depends on the message buffer type.

46.6.5.2.2.1 Receive message buffer and receive FIFO slot status description

This section describes the slot status structure for the individual receive message buffers and receive FIFOs. The content of the slot status structure for receive message buffers depends on the message buffer type and on the channel assignment for individual receive message buffers as given by *Table 897*.

Table 897. Receive message buffer slot status content

Receive message buffer type	Slot status content
Individual Receive Message Buffer assigned to both channels FR_MBCCFR*n*[CHA] = 1 and FR_MBCCFR*n*[CHB] = 1	See *Figure 933*
Individual Receive Message Buffer assigned to channel A FR_MBCCFR*n*[CHA] = 1 and FR_MBCCFR*n*[CHB] = 0	See *Figure 934*
Individual Receive Message Buffer assigned to channel B FR_MBCCFR*n*[CHA] = 0 and FR_MBCCFR*n*[CHB] = 1	See *Figure 935*
Receive FIFO Channel A Message Buffer	See *Figure 934*
Receive FIFO Channel B Message Buffer	See *Figure 935*

The meaning of the bits in the slot status structure is explained in *Table 898*.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	VFB	SYB	NFB	SUB	SEB	CEB	BVB	CH	VFA	SYA	NFA	SUA	SEA	CEA	BVA	0
Reset	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

Figure 933. Receive message buffer slot status structure (ChAB)

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	VFA	SYA	NFA	SUA	SEA	CEA	BVA	0
Reset	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

Figure 934. Receive message buffer slot status structure (ChA)

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	VFB	SYB	NFB	SUB	SEB	CEB	BVB	1	0	0	0	0	0	0	0	0
Reset	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

Figure 935. Receive message buffer slot status structure (ChB)

Table 898. Receive message buffer slot status field description

Field	Description
Common Message Buffer Status Bits	
VFB	Valid Frame on Channel B Protocol-related variable: *vSS!ValidFrame* channel B. 0 *vSS!ValidFrame* = 0 1 *vSS!ValidFrame* = 1
SYB	Sync Frame Indicator Channel B Protocol-related variable: *vRF!Header!SyFIndicator* channel B. 0 *vRF!Header!SyFIndicator* = 0 1 *vRF!Header!SyFIndicator* = 1
NFB	Null Frame Indicator Channel B Protocol-related variable: *vRF!Header!NFIndicator* channel B. 0 *vRF!Header!NFIndicator* = 0 1 *vRF!Header!NFIndicator* = 1
SUB	Startup Frame Indicator Channel B Protocol-related variable: *vRF!Header!SuFIndicator* channel B. 0 *vRF!Header!SuFIndicator* = 0 1 *vRF!Header!SuFIndicator* = 1
SEB	Syntax Error on Channel B Protocol-related variable: *vSS!SyntaxError* channel B. 0 *vSS!SyntaxError* = 0 1 *vSS!SyntaxError* = 1
CEB	Content Error on Channel B Protocol related variable: *vSS!ContentError* channel B. 0 *vSS!ContentError* = 0 1 *vSS!ContentError* = 1

Table 898. Receive message buffer slot status field description (continued)

Field	Description
BVB	Boundary Violation on Channel B Protocol-related variable: *vSS!BViolation* channel B. 0 *vSS!BViolation* = 0 1 *vSS!BViolation* = 1
CH	Channel first valid received This status bit applies only to receive message buffers assigned to the static segment and to both channels. It indicates the channel that has received the *first valid* frame in the slot. This flag is set to 0 if no valid frame was received at all in the subscribed slot. 0 First valid frame received on channel A, or no valid frame received at all 1 First valid frame received on channel B
VFA	Valid Frame on Channel A Protocol-related variable: *vSS!ValidFrame* channel A. 0 *vSS!ValidFrame* = 0 1 *vSS!ValidFrame* = 1
SYA	Sync Frame Indicator Channel A Protocol-related variable: *vRF!Header!SyFIndicator* channel A. 0 *vRF!Header!SyFIndicator* = 0 1 *vRF!Header!SyFIndicator* = 1
NFA	Null Frame Indicator Channel A Protocol-related variable: *vRF!Header!NFIndicator* channel A. 0 *vRF!Header!NFIndicator* = 0 1 *vRF!Header!NFIndicator* = 1
SUA	Startup Frame Indicator Channel A Protocol-related variable: *vRF!Header!SuFIndicator* channel A. 0 *vRF!Header!SuFIndicator* = 0 1 *vRF!Header!SuFIndicator* = 1
SEA	Syntax Error on Channel A Protocol-related variable: *vSS!SyntaxError* channel A. 0 *vSS!SyntaxError* = 0 1 *vSS!SyntaxError* = 1
CEA	Content Error on Channel A Protocol-related variable: *vSS!ContentError* channel A. 0 *vSS!ContentError* = 0 1 *vSS!ContentError* = 1
BVA	Boundary Violation on Channel A Protocol-related variable: *vSS!BViolation* channel A. 0 *vSS!BViolation* = 0 1 *vSS!BViolation* = 1

46.6.5.2.2.2 Transmit Message Buffer Slot Status Description

This section describes the slot status structure for transmit message buffers. Only the TCA and TCB status bits are directly related to the transmission process. All other status bits in this structure are related to a receive process that may have occurred. The content of the slot status structure for transmit message buffers depends on the channel assignment as given by *Table 899*.

Table 899. Transmit message buffer slot status content

Transmit Message Buffer Type	Slot Status Content
Individual Transmit Message Buffer assigned to both channels FR_MBCCFR*n*[CHA]=1 and FR_MBCCFR*n*[CHB]=1	See *Figure 936*
Individual Transmit Message Buffer assigned to channel A FR_MBCCFR*n*[CHA]=1 and FR_MBCCFR*n*[CHB]=0	See *Figure 937*
Individual Transmit Message Buffer assigned to channel B FR_MBCCFR*n*[CHA]=0 and FR_MBCCFR*n*[CHB]=1	See *Figure 938*

The meaning of the bits in the slot status structure is described in *Table 900*.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	VFB	SYB	NFB	SUB	SEB	CEB	BVB	TCB	VFA	SYA	NFA	SUA	SEA	CEA	BVA	TCA
Reset	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

Figure 936. Transmit message buffer slot status structure (ChAB)

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	VFA	SYA	NFA	SUA	SEA	CEA	BVA	TCA
Reset	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

Figure 937. Transmit message buffer slot status structure (ChA)

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	VFB	SYB	NFB	SUB	SEB	CEB	BVB	TCB	0	0	0	0	0	0	0	0
Reset	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

Figure 938. Transmit message buffer slot status structure (ChB)

Table 900. Transmit message buffer slot status structure field description

Field	Description
VFB	Valid Frame on Channel B Protocol-related variable: *vSS!ValidFrame* channel B. 0 *vSS!ValidFrame* = 0 1 *vSS!ValidFrame* = 1
SYB	Sync Frame Indicator Channel B Protocol-related variable: *vRF!Header!SyFIndicator* channel B. 0 *vRF!Header!SyFIndicator* = 0 1 *vRF!Header!SyFIndicator* = 1
NFB	Null Frame Indicator Channel B Protocol-related variable: *vRF!Header!NFIndicator* channel B. 0 *vRF!Header!NFIndicator* = 0 1 *vRF!Header!NFIndicator* = 1
SUB	Startup Frame Indicator Channel B Protocol-related variable: *vRF!Header!SuFIndicator* channel B. 0 *vRF!Header!SuFIndicator* = 0 1 *vRF!Header!SuFIndicator* = 1

Table 900. Transmit message buffer slot status structure field description (continued)

Field	Description
SEB	Syntax Error on Channel B Protocol-related variable: *vSS!SyntaxError* channel B. 0 *vSS!SyntaxError* = 0 1 *vSS!SyntaxError* = 1
CEB	Content Error on Channel B Protocol-related variable: *vSS!ContentError* channel B. 0 *vSS!ContentError* = 0 1 *vSS!ContentError* = 1
BVB	Boundary Violation on Channel B Protocol-related variable: *vSS!BViolation* channel B. 0 *vSS!BViolation* = 0 1 *vSS!BViolation* = 1
TCB	Transmission Conflict on Channel B Protocol-related variable: *vSS!TxConflict* channel B. 0 *vSS!TxConflict* = 0 1 *vSS!TxConflict* = 1
VFA	Valid Frame on Channel A Protocol-related variable: *vSS!ValidFrame* channel A. 0 *vSS!ValidFrame* = 0 1 *vSS!ValidFrame* = 1
SYA	Sync Frame Indicator Channel A Protocol-related variable: *vRF!Header!SyFIndicator* channel A. 0 *vRF!Header!SyFIndicator* = 0 1 *vRF!Header!SyFIndicator* = 1
NFA	Null Frame Indicator Channel A Protocol-related variable: *vRF!Header!NFIndicator* channel A. 0 *vRF!Header!NFIndicator* = 0 1 *vRF!Header!NFIndicator* = 1
SUA	Startup Frame Indicator Channel A Protocol-related variable: *vRF!Header!SuFIndicator* channel A. 0 *vRF!Header!SuFIndicator* = 0 1 *vRF!Header!SuFIndicator* = 1
SEA	Syntax Error on Channel A Protocol-related variable: *vSS!SyntaxError* channel A. 0 *vSS!SyntaxError* = 0 1 *vSS!SyntaxError* = 1
CEA	Content Error on Channel A Protocol-related variable: *vSS!ContentError* channel A. 0 *vSS!ContentError* = 0 1 *vSS!ContentError* = 1

Table 900. Transmit message buffer slot status structure field description (continued)

Field	Description
BVA	Boundary Violation on Channel A Protocol-related variable: *vSS!BViolation* channel A. 0 *vSS!BViolation* = 0 1 *vSS!BViolation* = 1
TCA	Transmission Conflict on Channel A Protocol-related variable: *vSS!TxConflict* channel A. 0 *vSS!TxConflict* = 0 1 *vSS!TxConflict* = 1

46.6.5.3 Message buffer data field description

The message buffer data field is used to store the frame payload data, or a part of it, of the frame to be transmitted to or received from the FlexRay bus. The minimum required length of this field depends on the message buffer type that the physical message buffer is assigned to and is given in *Table 901*. The structure of the message buffer data field is given in *Figure 939*.

Table 901. Message buffer data field minimum length

Physical message buffer assigned to	Minimum length defined by
Individual message buffer in segment 1	FR_MBDSR[MBSEG1DS]
Receive shadow buffer in segment 1	FR_MBDSR[MBSEG1DS]
Individual message buffer in segment 2	FR_MBDSR[MBSEG2DS]
Receive shadow buffer in segment 2	FR_MBDSR[MBSEG2DS]
Receive FIFO for channel A	FR_RFDSR[ENTRY_SIZE] (FR_RFWMSR[SEL] = 0)
Receive FIFO for channel B	FR_RFDSR[ENTRY_SIZE] (FR_RFWMSR[SEL] = 1)

Note: The CC will not access any locations outside the message buffer data field boundaries given by Table 901.

	0 1 2 3	4 5 6 7	8 9 10 11	12 13 14 15
0x0	DATA0 / MID0 / NMV0		DATA1 / MID1 / NMV1	
0x2	DATA2 / NMV2		DATA3 / NMV3	
…	…		…	
0xN-2	DATA N-2		DATA N-1	

Figure 939. Message buffer data field structure

The message buffer data field is located in the FlexRay memory area; thus, the CC has no means to control application write access to the field. To ensure data consistency, the application must follow a write and read access scheme.

46.6.5.3.1 Message buffer data field read access

For transmit message buffers, the CC will not modify the content of the message buffer data field. Thus the application can read back the data at any time without any impact on data consistency.

For receive message buffers the application must lock the related receive message buffer and retrieve the message buffer header index from the Message Buffer Index Registers (FR_MBIDXR*n*). While the message buffer is locked, the CC will not update the message buffer data field.

For receive FIFOs, the application can read the message buffer indicated by the Receive FIFO A Read Index Register (FR_RFARIR) or the Receive FIFO B Read Index Register (FR_RFBRIR) when the related fill levels in the Receive FIFO Fill Level and POP Count Register (FR_RFFLPCR) indicate an non-empty FIFO.

46.6.5.3.2 Message buffer data field write access

For receive message buffers, receive shadow buffers, and receive FIFOs, the application must not write to the message buffer data field.

For transmit message buffers, the application must follow the write access restrictions given in *Table 902*.

Table 902. Frame data write access constraints

Field	CC/MB State
DATA, MID, NMV	*POC:config* or MB_DIS or MB_LCK

Table 903. Frame data field descriptions

Field	Description
DATA 0, DATA 1, … DATA N–1	Message Data Provides the message data received or to be transmitted. For receive message buffer and receive FIFOs, this field provides the message data received for this message buffer. For transmit message buffers, the field provides the message data to be transmitted.
MID 0, MID 1	Message Identifier If the payload preamble bit PPI is set in the message buffer frame header, the MID field holds the message ID of a dynamic frame located in the message buffer. The receive FIFO filter uses the received message ID for message ID filtering.
NMV 0, NMV 1, … NMV 11	Network Management Vector If the payload preamble bit PPI is set in the message buffer frame header, the network management vector field holds the network management vector of a static frame located in the message buffer. **Note:** The MID and NMV bytes replace the corresponding DATA bytes.

46.6.6 Individual message buffer functional description

The CC provides the following basic types of individual message buffers:

- Transmit message buffers
- Receive message buffers

Before an individual message buffer can be used, it must be configured by the application. After the initial configuration, the message buffer can be reconfigured later. The set of the configuration data for individual message buffers is given in *Section 46.6.3.4.1: Individual message buffer configuration data*.

46.6.6.1 Individual message buffer configuration

The individual message buffer configuration consists of two steps. The first step is the allocation of the required amount of memory for the FlexRay memory area. The second step is the programming of the message buffer configuration registers, which is described in this section.

46.6.6.1.1 Common configuration data

One part of the message buffer configuration data is common to all individual message buffers and the receive shadow buffers. This data can only be set when the protocol is in the *POC:config* state.

The application configures the number of utilized individual message buffers by writing the message buffer number of the last utilized message buffer into the LAST_MB_UTIL field in the Message Buffer Segment Size and Utilization Register (FR_MBSSUTR).

The application configures the size of the two segments of individual message buffers by writing the message buffer number of the last message buffer in the first segment into the LAST_MB_SEG1 field in the Message Buffer Segment Size and Utilization Register (FR_MBSSUTR).

The application configures the length of the message buffer data fields for both of the message buffer segments by writing to the MBSEG2DS and MBSEG1DS fields in the Message Buffer Data Size Register (FR_MBDSR).

Depending on the current receive functionality of the CC, the application must configure the receive shadow buffers. For each segment and for each channel with at least one individual receive message buffer assigned, the application must configure the related receive shadow buffer using the Receive Shadow Buffer Index Register (FR_RSBIR).

46.6.6.1.2 Specific configuration data

The second part of the message buffer configuration data is specific for each message buffer.

This data can be changed only when either

- The protocol is in the POC:config state, or
- The message buffer is disabled (that is, FR_MBCCSR*n*[EDS] = 0)

The individual message buffer type is defined by the MTD bit in the Message Buffer Configuration, Control, Status Registers (FR_MBCCSR*n*) as given in *Table 904*.

Table 904. Individual message buffer types

FR_MBCCSRn[MTD]	Individual message buffer description
0	Receive message buffer
1	Transmit message buffer

The message-buffer-specific configuration data includes the following:

1. MTD bits in the Message Buffer Configuration, Control, Status Registers (FR_MBCCSR*n*)
2. All fields in the Message Buffer Cycle Counter Filter Registers (FR_MBCCFR*n*)
3. All fields in the Message Buffer Frame ID Registers (FR_MBFIDR*n*)
4. All fields in the Message Buffer Index Registers (FR_MBIDXR*n*)

The meaning of the specific configuration data depends on the message buffer type, as given in the detailed message buffer type descriptions. See *Section 46.6.6.2: Transmit message buffers* and *Section 46.6.6.3: Receive message buffers* for further information.

46.6.6.2 Transmit message buffers

The section provides a detailed description of the functionality of single-buffered transmit message buffers.

A transmit message buffer is used by the application to provide message data to the CC that will be transmitted over the FlexRay Bus. The CC uses the transmit message buffers to provide information about the transmission process and status information about the slot in which message was transmitted.

The individual message buffer with message buffer number *n* is configured to be a transmit message buffer by the following setting:

- FR_MBCCSR*n*[MTD] = 1 (transmit message buffer)

46.6.6.2.1 Access regions

For certain message buffer fields, both the application and the CC have access. To ensure data consistency, a message buffer locking scheme is used to control the access to the data, control, and status bits of a message buffer. The access regions for transmit message buffers are depicted in *Figure 940*. A description of the regions is given in *Table 906*. If a region is active as indicated in *Table 907*, the access scheme given for that region applies to the message buffer.

Figure 940. Transmit message buffer access regions

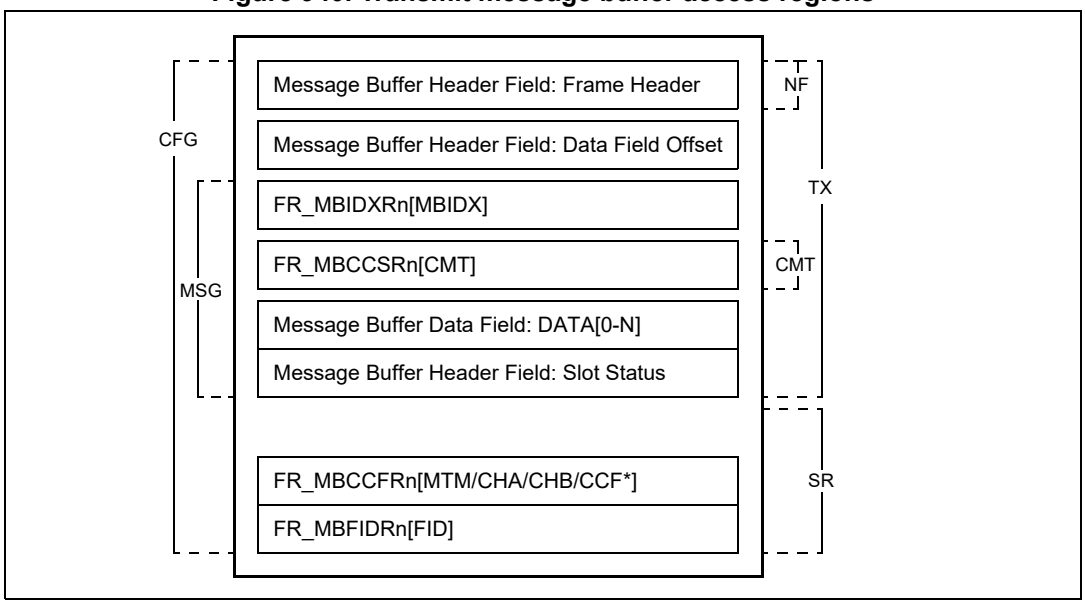


Table 905. Transmit message buffer access regions description

Region	Access from		Region used for
	Application	Module	
CFG	Read/write	—	Message buffer configuration
MSG	Read/write	—	Message data and slot status access
NF	—	Read-only	Message header access for null frame transmission
TX	—	Read/write	Message transmission and slot status update
CMT	—	Read-only	Message buffer validation
SR	—	Read-only	Message buffer search

The trigger bits FR_MBCCSR*n*[EDT] and FR_MBCCSR*n*[LCKT], and the interrupt enable bit FR_MBCCSR*n*[MBIE], are not under access control and can be accessed from the application at any time. The status bits FR_MBCCSR*n*[EDS] and FR_MBCCSR*n*[LCKS] are not under access control and can be accessed from the CC at any time.

The interrupt flag FR_MBCCSR*n*[MBIF] is not under access control and can be accessed from the application and the CC at any time. CC clear access has higher priority.

The CC restricts its access to the regions depending on the current state of the message buffer. The application must adhere to these restrictions in order to ensure data consistency. The transmit message buffer states are given in *Figure 941*. A description of the states is given in *Table 906*, which also provides the access scheme for the access regions.

The status bits FR_MBCCSR*n*[EDS] and FR_MBCCSR*n*[LCKS] provide the application with the required message buffer status information. The internal status information is not visible to the application.

46.6.6.2.2 Message buffer states

This section describes the transmit message buffer states and provides a state diagram.

Figure 941. Transmit message buffer states

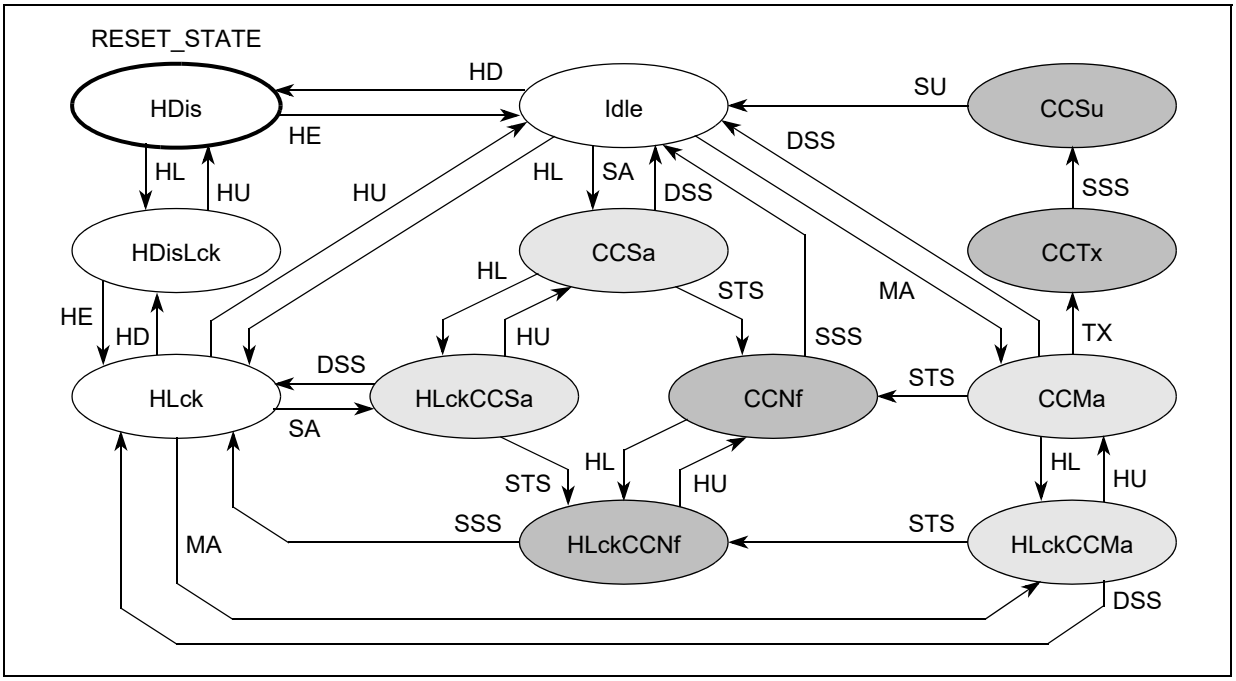


Table 906. Transmit message buffer state description

State	FR_MBCCSR*n*		Access region		Description
	EDS	LCKS	Appl.	Module	
Idle	1	0	—	CMT, SR	Idle – Message Buffer is idle. Included in message buffer search
HDis	0	0	CFG	—	Disabled – Message Buffer under configuration. Excluded from message buffer search.
HDisLck	0	1	CFG	—	Disabled and Locked – Message Buffer under configuration. Excluded from message buffer search.
HLck	1	1	MSG	SR	Locked – Application access to data, control, and status. Included in message buffer search.
CCSa	1	0	—	—	Slot Assigned – Message buffer assigned to next static slot. Ready for Null Frame transmission.
HLckCCSa	1	1	MSG	—	Locked and Slot Assigned – Application access to data, control, and status.Message buffer assigned to next static slot.
CCNf	1	0	—	NF	Null Frame Transmission – Header is used for null frame transmission.
HLckCCNf	1	1	MSG	NF	Locked and Null Frame Transmission – Application access to data, control, and status. Header is used for null frame transmission.
CCMa	1	0	—	CMT	Message Available – Message buffer is assigned to next slot and cycle counter filter matches.

Table 906. Transmit message buffer state description (continued)

State	FR_MBCCSR*n*		Access region		Description
	EDS	LCKS	Appl.	Module	
HLckCCMa	1	1	MSG	—	Locked and Message Available – Applications access to data, control, and status. Message buffer is assigned to next slot and cycle counter filter matches.
CCTx	1	0	—	TX	Message Transmission – Message buffer data transmit. Payload data from buffer transmitted.
CCSu	1	0	—	TX	Status Update – Message buffer status update. Update of status flags, the slot status field, and the header index.

46.6.6.2.3 Message buffer transitions

46.6.6.2.3.1 Application transitions

The application transitions can be triggered by the application using the commands described in *Table 912*. The application issues the commands by writing to the Message Buffer Configuration, Control, Status Registers (FR_MBCCSR*n*). Only one command can be issued with one write access. Each command is executed immediately. If the command is ignored, it must be issued again.

- Message Buffer Enable and Disable

The enable and disable commands are issued by writing 1 to the trigger bit FR_MBCCSR*n*[EDT]. The transition that will be triggered by each of these command depends on the current value of the status bit FR_MBCCSR*n*[EDS]. If the command triggers the disable transition HD and the message buffer is in the CCSa, HLckCCSa, CCMa, HLckCCMa, CCNf, HLckCCNf, or CCTx state, the disable transition has no effect (command is ignored) and the message buffer state is not changed. No notification is given to the application.

- Message Buffer Lock and Unlock

The lock and unlock commands are issued by writing 1 to the trigger bit FR_MBCCSR*n*[LCKT]. The transition that will be triggered by each of these commands depends on the current value of the status bit FR_MBCCSRn[LCKS]. If the command triggers the lock transition HL and the message buffer is in the state CCTx, the lock transition has no effect (command is ignored) and message buffer state is not changed. In this case, the message buffer lock error flag LCK_EF in the CHI Error Flag Register (FR_CHIERFR) is set.

Table 907. Transmit message buffer application transitions

Transition	Command	Condition	Description
HE	FR_MBCCSR*n*[EDT] = 1	FR_MBCCSR*n*[EDS] = 0	Application triggers message buffer enable.
HD		FR_MBCCSR*n*[EDS] = 1	Application triggers message buffer disable.

Table 907. Transmit message buffer application transitions (continued)

Transition	Command	Condition	Description
HL	FR_MBCCSR*n*[LCKT] = 1	FR_MBCCSR*n*[LCKS] = 0	Application triggers message buffer lock.
HU		FR_MBCCSR*n*[LCKS] = 1	Application triggers message buffer unlock.

46.6.6.2.3.2 Module transitions

The module transitions that can be triggered by the CC are described in *Table 908*. Each transition will be triggered for certain message buffers when the related condition is fulfilled.

Table 908. Transmit message buffer module transitions

Transition	Condition	Description
SA	Slot match and static slot	Slot Assigned – Message buffer is assigned to next static slot.
MA	Slot match and CycleCounter match	Message Available – Message buffer is assigned to next slot and cycle counter filter matches.
TX	Slot start and FR_MBCCSR*n*[CMT] = 1	Transmission Slot Start – Slot Start and commit bit CMT is set. In case of a dynamic slot, pLatestTx is not exceeded.
SU	Status updated	Status Updated – Slot Status field and message buffer status flags updated. Interrupt flag set.
STS	Static slot start	Static Slot Start – Start of static slot.
DSS	Dynamic slot start or symbol window start or NIT start	Dynamic Slot or Segment Start – Start of dynamic slot or symbol window or NIT.
SSS	Slot start or symbol window start or NIT start	Slot or Segment Start – Start of static slot or dynamic slot or symbol window or NIT.

46.6.6.2.3.3 Transition priorities

The application can trigger only one transition at a time. There is no need to specify priorities among them.

As shown in the first part of *Table 909*, the module transitions have a higher priority than the application transitions. For all states except the CCMa state, both a lock/unlock transition HL/HD and a module transition can be executed at the same time. The result state is reached by first applying the application transition and, subsequently, the module transition to the intermediately reached state. For example, if the message buffer is in the HLck state and the application unlocks the message buffer by the HU transition and the module triggers the slot assigned transition SA, the intermediate state is Idle and the resulting state is CCSa.

The priorities among the module transitions are given in the second part of *Table 909*.

Table 909. Transmit message buffer transition priorities

State	Priorities	Description
Module vs. application		
Idle, HLck	SA > HD MA > HD	Slot Assigned > Message Buffer Disable Message Available > Message Buffer Disable
CCMa	TX > HL	Transmission Start > Message Buffer Lock
Module internal		
Idle, HLck	MA > SA	Message Available > Slot Assigned
CCMa	TX > STS TX > DSS	Transmission Slot Start > Static Slot Start Transmission Slot Start > Dynamic Slot Start

46.6.6.2.4 Transmit message setup

To transmit a message over the FlexRay bus, the application writes the message data into the message buffer data field and sets the commit bit, CMT, in the Message Buffer Configuration, Control, Status Registers (FR_MBCCSR*n*). The physical access to the message buffer data field is described in *Section 46.6.3.1: Individual Message Buffers*.

As indicated by *Table 900*, the application will write to the message buffer data field and change the commit bit CMT only if the transmit message buffer is in one of the following states: HDis, HDisLck, HLck, HLckCCSa, HLckCCMa, or HLckCCMa. The application can change the state of a message buffer if it issues the appropriate commands shown in *Table 900*. The state change is indicated through the FR_MBCCSR*n*[EDS] and FR_MBCCSRn[LCKS] status bits.

If the transmit message buffer enters the HDis, HDisLck, HLck, HLckCCSa, HLckCCMa, or HLckCCMa state, the FR_MBCCSR*n*[DVAL] flag is negated.

46.6.6.2.5 Message transmission

As a result of the message buffer search described in *Section 46.6.7: Individual message buffer search*, the CC triggers the message available transition MA for up to two transmit message buffers. This changes the message buffer state from Idle to CCMa and the message buffers can be used for message transmission in the next slot.

The CC transmits a message from a message buffer if both of the following two conditions are fulfilled at the start of the transmission slot:

1. The message buffer is in the message available state CCMa
2. The message data is still valid (that is, FR_MBCCSR*n*[CMT] = 1)

In this case, the CC triggers the TX transition and changes the message buffer state to CCTx. A transmit message buffer timing and state change diagram for message transmission is given in *Figure 942*. In this example, the message buffer with message buffer number *n* is Idle at the start of the search slot, matches the slot and cycle number of the next slot, and message buffer data are valid (that is, FR_MBCCSR*n*[CMT] = 1).

Figure 942. Message transmission timing

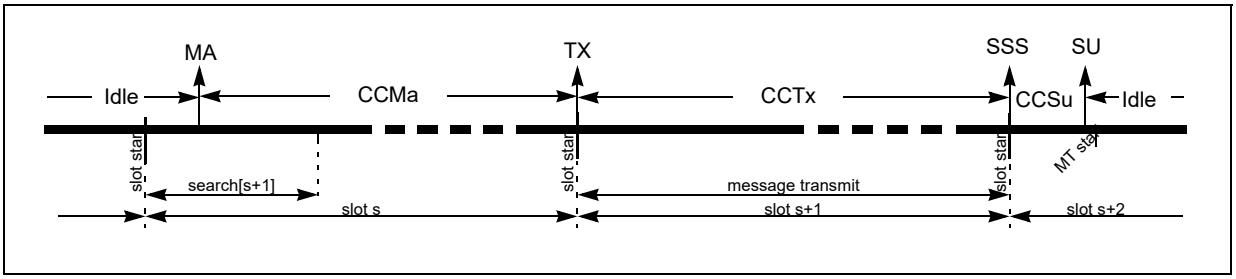
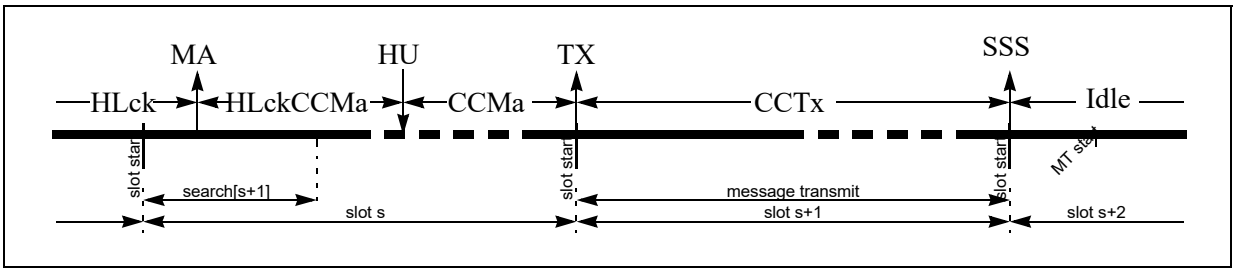


Figure 943. Message transmission from HLck state with unlock



The amount of message data read from the FlexRay memory area and transferred to the FlexRay bus is determined by the following three items

1. The message buffer segment that the message buffer is assigned to, as defined by the Message Buffer Segment Size and Utilization Register (FR_MBSSUTR).
2. The message buffer data field size, as defined by the related field of the Message Buffer Data Size Register (FR_MBDSR)
3. The value of the PLDLEN field in the message buffer header field, as described in *Section 46.6.5.2.1: Frame header description*

If a message buffer is assigned to message buffer segment 1, and PLDLEN > MBSEG1DS, then 2 × MBSEG1DS bytes will be read from the message buffer data field and zero padding is used for the remaining bytes for the FlexRay bus transfer. If PLDLEN ≤ MBSEG1DS, the CC reads and transfers 2 ×PLDLEN bytes. The same holds for segment 2 and MBSEG2DS.

46.6.6.2.6 Null frame transmission

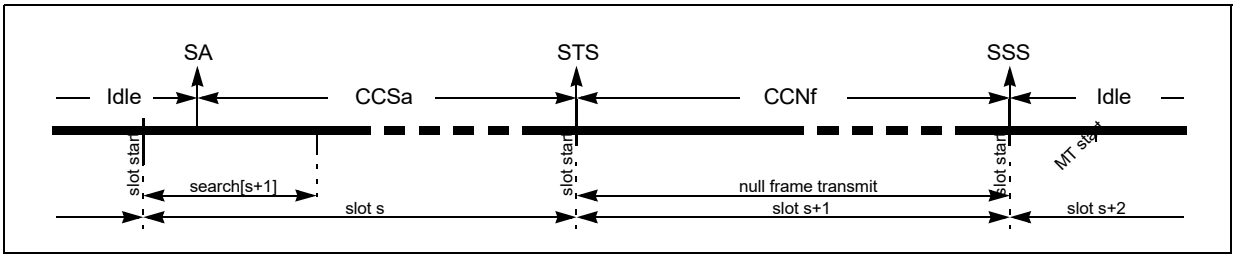
A static slot with slot number S is assigned to the CC for channel A, if at least one transmit message buffer is configured with the FR_MBFIDR*n*[FID] set to S and FR_MBCCFR*n*[CHA] set to 1. A null frame is transmitted in the static slot *s* on channel A, if this slot is assigned to the CC for channel A, and all transmit message buffers with FR_MBFIDR*n*[FID] = S and FR_MBCCFR*n*[CHA] = 1 are either not committed (that is FR_MBCCSR*n*[CMT] = 0), or are locked by the application (that is, FR_MBCCSR*n*[LCKS] = 1), or the cycle counter filter is enabled and does not match.

Additionally, the application can clear the commit bit of a message buffer that is in the CCMa state, which is called *uncommit* or *transmit abort*. This message buffer will be used for null frame transmission.

As a result of the message buffer search described in *Section 46.6.7: Individual message buffer search*, the CC triggers the slot assigned transition SA for up to two transmit message buffers if at least one of the conditions mentioned above is fulfilled for these message buffers. The transition SA changes the message buffer states from either Idle to CCSa or

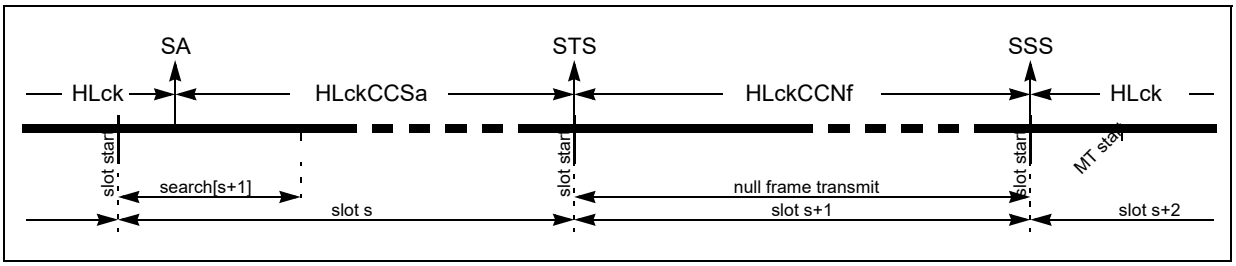
from HLck to HLckCCSa. In each case, these message buffers will be used for null frame transmission in the next slot. A message buffer timing and state change diagram for null frame transmission from Idle state is given in *Figure 944*.

Figure 944. Null frame transmission from Idle state



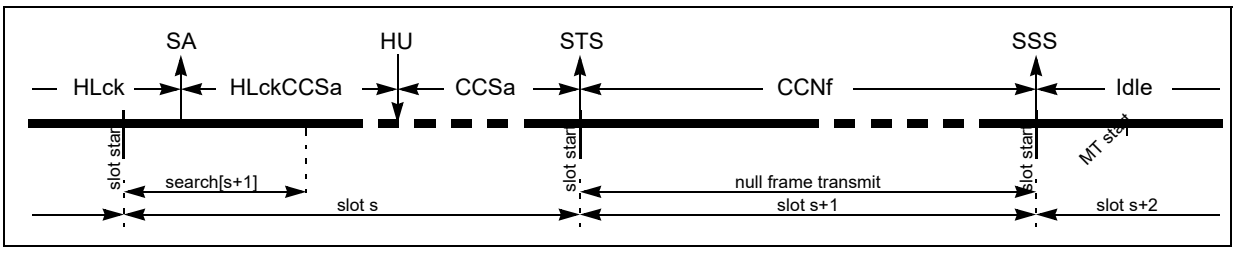
A message buffer timing and state change diagram for null frame transmission from HLck state is given in *Figure 945*.

Figure 945. Null frame transmission from HLck state



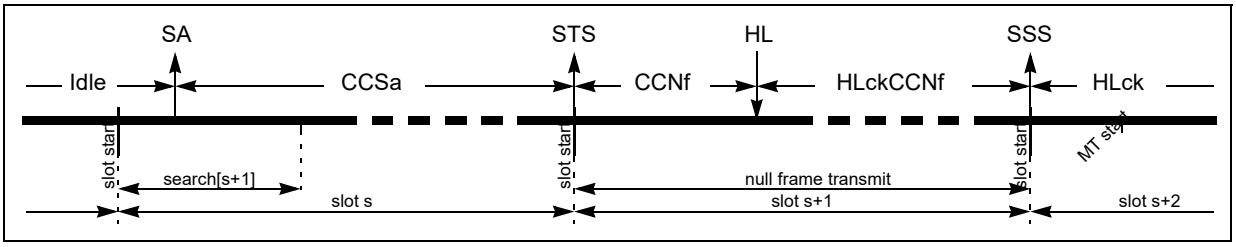
If a transmit message buffer is in the CCSa or HLckCCSa state at the start of the transmission slot, a null frame is transmitted in any case, even if the message buffer is unlocked or committed before the transmission slot starts. A transmit message buffer timing and state change diagram for null frame transmission for this case is given in *Figure 946*.

Figure 946. Null frame transmission from HLck state with unlock



Since the null frame transmission will not use the message buffer data, the application can lock/unlock the message buffer during null frame transmission. A transmit message buffer timing and state change diagram for null frame transmission for this case is given in *Figure 947*.

Figure 947. Null frame transmission from Idle state with locking



46.6.6.2.7 Message buffer status update

After the end of each slot, the PE generates the slot status vector. Depending on the status, the transmitted frame type, and the amount of transmitted data, the message buffer status is updated.

46.6.6.2.7.1 Message buffer status update after complete message transmission

The term complete message transmission refers to the fact that all payload data stored in the message buffer was sent to the FlexRay bus. In this case, the CC updates the slot status field of the message buffer and triggers the status updated transition SU. With the SU transition, the CC sets the message buffer interrupt flag FR_MBCCSR*n*[MBIF] to indicate the successful message transmission.

Depending on the Transmission Mode flag FR_MBCCFR*n*[MTM], the CC changes the commit flag FR_MBCCSR*n*[CMT] and the valid flag FR_MBCCSR*n*[DVAL]. If the FR_MBCCFRn[MTM] flag is negated, the message buffer is in *Event Transmission mode.* In this case, each committed message is transmitted only once. The commit flag FR_MBCCSR*n*[CMT] is cleared with the SU transition. If the FR_MBCCFR*n*[MTM] flag is asserted, the message buffer is in the *State Transmission mode*. In this case, each committed message is transmitted as long as the application provides new data or locks the message buffers. The CC will not clear the FR_MBCCSR*n*[CMT] flag at the end of transmission and will set the valid flag FR_MBCCSR*n*[DVAL] to indicate that the message will be transmitted again.

46.6.6.2.7.2 Message buffer status update after incomplete message transmission

The term *incomplete message transmission* refers to the fact that not all payload data that should be transmitted was sent to FlexRay bus. This may be caused by the following regular conditions in the dynamic segment:

1. The transmission slot starts in a minislot with a minislot number greater than *pLatestTx*.
2. The transmission slot did not exist in the dynamic segment at all.

Additionally, an incomplete message transmission can be caused by internal communication errors. If those error occur, the Protocol Engine Communication Failure Interrupt Flag PECF_IF is set in the Protocol Interrupt Flag Register 1 (FR_PIFR1).

In either of these two cases, the status of the message buffer is not changed at all with the SU transition. The slot status field is not updated, the status and control flags are not changed, and the interrupt flag is not set.

46.6.6.2.7.3 Message buffer status update after null frame transmission

After the transmission of a null frame, the status of the message buffer that was used for the null frame transmission is not changed at all. The slot status field is not updated, the status and control flags are not changed, and the interrupt flag is not set.

46.6.6.3 Receive message buffers

The section provides a detailed description of the functionality of the receive message buffers. If receive message buffers are used, the related receive shadow buffer must be configured as described in *Section 46.6.3.2: Receive shadow buffers*.

A receive message buffer is used to receive a message from the FlexRay bus based on individual filter criteria. The CC uses the receive message buffer to provide the following data to the application:

- Message data received
- Information about the reception process
- Status information about the slot in which the message was received

A individual message buffer with message buffer number *n* is configured as a receive message buffer by the following configuration settings:

- FR_MBCCSR*n*[MTD] = 0 (receive message buffer)

Both the application and the CC have access to certain message buffers. To ensure data consistency, a message buffer locking scheme controls the access to the data, control, and status bits of a message buffer. The access regions for receive message buffers are depicted in *Figure 948*. A description of the regions is given in *Table 910*. If an region is active as indicated in *Table 911*, the access scheme given for that region applies to the message buffer.

Figure 948. Receive message buffer access regions

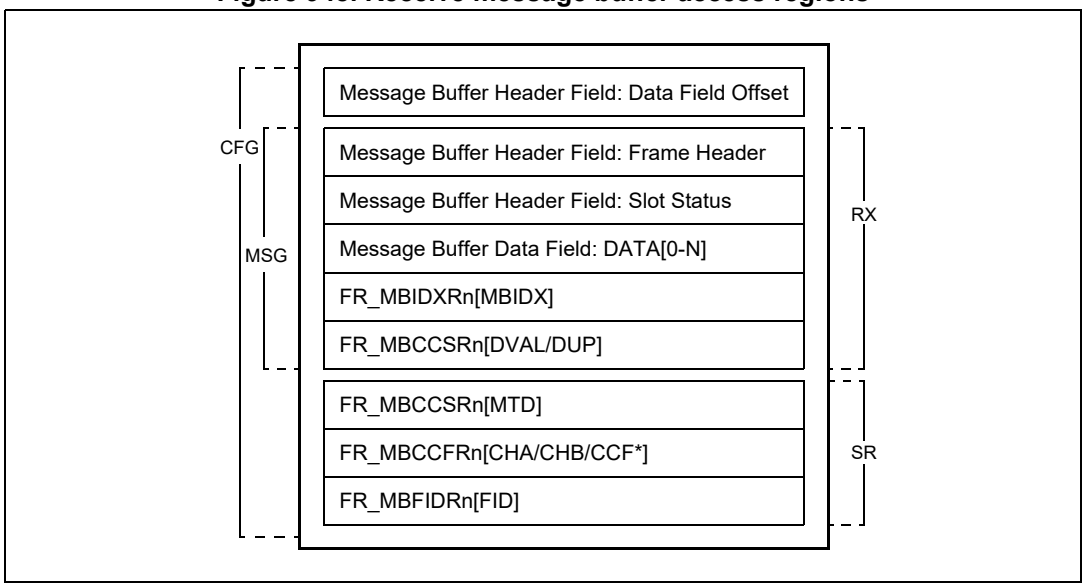


Table 910. Receive message buffer access region description

Region	Access from		Region used for
	Application	Module	
CFG	Read/write	—	Message buffer configuration, message data, and status access
MSG	Read/write	—	Message data, header, and status access
RX	—	Write-only	Message reception and status update
SR	—	Read-only	Message buffer search data

The trigger bits FR_MBCCSR*n*[EDT] and FR_MBCCSR*n*[LCKT] and the interrupt enable bit FR_MBCCSR*n*[MBIE] are not under access control and can be accessed from the application at any time. The status bits FR_MBCCSR*n*[EDS] and FR_MBCCSR*n*[LCKS] are not under access control and can be accessed from the CC at any time.

The interrupt flag FR_MBCCSR*n*[MBIF] is not under access control and can be accessed from the application and the CC at any time. CC set access has higher priority.

The CC restricts its access to the regions depending on the current state of the message buffer. The application must adhere to these restrictions in order to ensure data consistency. The receive message buffer states are given in *Figure 949*. A description of the message buffer states is given in *Table 906*, which also provides the access scheme for the access regions.

The status bits FR_MBCCSR*n*[EDS] and FR_MBCCSR*n*[LCKS] provide the application with the required status information. The internal status information is not visible to the application.

Figure 949. Receive message buffer states

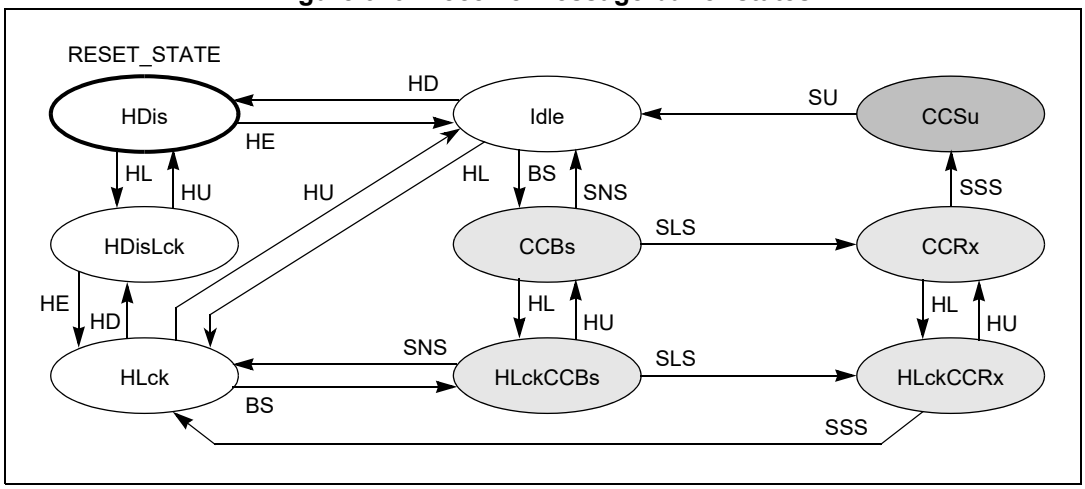


Table 911. Receive message buffer states and access

State	FR_MBCCSR*n*		Access from		Description
	EDS	LCKS	Appl.	Module	
Idle	1	0	—	SR	Idle – Message buffer is idle. Included in message buffer search.
HDis	0	0	CFG	—	Disabled – Message buffer under configuration. Excluded from message buffer search.
HDisLck	0	1	CFG	—	Disabled and Locked – Message buffer under configuration. Excluded from message buffer search.
HLck	1	1	MSG	—	Locked – Application access to data, control, and status. Included in message buffer search.
CCBs	1	0	—	—	Buffer Subscribed – Message buffer subscribed for reception. Filter matches next (slot, cycle, channel) tuple.
HLckCCBs	1	1	MSG	—	Locked and Buffer Subscribed – Application access to data, control, and status. Message buffer subscribed for reception.
CCRx	1	0	—	—	Message Receive – Message data received into related shadow buffer.
HLckCCRx	1	1	MSG	—	Locked and Message Receive – Application access to data, control, and status. Message data received into related shadow buffer.
CCSu	1	0	—	RX	Status Update – Message buffer status update. Update of status flags, the slot status field, and the header index.

46.6.6.3.1 Message buffer transitions

46.6.6.3.1.1 Application transitions

The application transitions that can be triggered by the application using the commands are described in *Table 907*. The application issues the commands by writing to the Message Buffer Configuration, Control, Status Registers (FR_MBCCSR*n*). Only one command can be issued with one write access. Each command is executed immediately. If the command is ignored, it must be issued again.

Message buffer enable and disable

The enable and disable commands are issued by writing 1 to the trigger bit FR_MBCCSR*n*[EDT]. The transition that will be triggered by each of these command depends on the current value of the status bit FR_MBCCSR*n*[EDS]. If the command triggers the disable transition HD and the message buffer is in the CCBs, HLckCCBs, or CCRx state, the disable transition has no effect (command is ignored) and the message buffer state is not changed. No notification is given to the application.

Message buffer lock and unlock

The lock and unlock commands are issued by writing 1 to the trigger bit FR_MBCCSR*n*[LCKT]. The transition that will be triggered by each of these commands depends on the current value of the status bit FR_MBCCSR*n*[LCKS]. If the command triggers the lock transition HL while the message buffer is in the CCRx state, the lock transition has no effect (command is ignored) and the message buffer state is not changed.

In this case, the message buffer lock error flag LCK_EF in the CHI Error Flag Register (FR_CHIERFR) is set.

Table 912. Receive message buffer application transitions

Transition	Host command	Condition	Description
HE	FR_MBCCSR*n*[EDT] = 1	FR_MBCCSR*n*[EDS] = 0	Application triggers message buffer enable
HD		FR_MBCCSR*n*[EDS] = 1	Application triggers message buffer disable
HL	FR_MBCCSR*n*[LCKT] = 1	FR_MBCCSR*n*[LCKS = 0	Application triggers message buffer lock
HU		FR_MBCCSR*n*[LCKS] = 1	Application triggers message buffer unlock

Module Transitions

The module transitions that can be triggered by the CC are described in *Table 913*. Each transition will be triggered for certain message buffers when the related condition is fulfilled.

Table 913. Receive message buffer module transitions

Transition	Condition	Description
BS	Slot match and cyclecounter match	Buffer Subscribed – The message buffer filter matches next slot and cycle.
SLS	Slot start	Slot Start – Start of either Static Slot or Dynamic Slot.
SNS	Symbol window start or NIT start	Symbol Window or NIT Start – Start of either Symbol Window or NIT.
SSS	Slot start, symbol window start, or NIT start	Slot or Segment Start – Start of either Static Slot, Dynamic Slot, Symbol Window, or NIT.
SU	Status updated	Status Updated – Slot Status field, message buffer status flags, header index updated. Interrupt flag set.

46.6.6.3.1.2 Transition priorities

The application can trigger only one transition at a time. There is no need to specify priorities among them.

As shown in *Table 914*, the module transitions have a higher priority than the application transitions. For all states except the CCRx state, a module transition and the application lock/unlock transition HL/HU can be executed at the same time. The result state is reached by first applying the module transition and subsequently the application transition to the intermediately reached state. For example, if the message buffer is in the CCBs buffer subscribed state and the module triggers the SLS slot start transition at the same time that the application locks the message buffer through the HL transition, the intermediate state is CCRx and the resulting state is locked buffer subscribed state, HLckCCRx.

Table 914. Receive message buffer transition priorities

State	Priorities	Description
Module vs. Application		
Idle	BS > HD	Buffer subscribed > Message buffer disable

Table 914. Receive message buffer transition priorities (continued)

State	Priorities	Description
HLck	BS > HD	Buffer subscribed > Message buffer disable
CCRx	SSS > HL	Slot or segment start > Message buffer lock

46.6.6.3.2 Message reception

As a result of the message buffer search, the CC changes the state of up to two enabled receive message buffers from either idle state Idle or locked state HLck to either subscribed state CCBs or locked buffer subscribed state HLckCCBs by triggering the buffer subscribed transition BS.

If the receive message buffers for the next slot are assigned to both channels, then at most one receive message buffer is changed to a buffer subscribed state.

If more than one matching message buffer is assigned to a certain channel, then only the message buffer with the lowest message buffer number is in one of the states mentioned above.

With the start of the next static or dynamic slot, the module triggers the slot start transition SLS. This changes the state of the subscribed receive message buffers from either CCBs to CCRx or from HLckCCBs to HLckCCRx, respectively.

During the reception slot, the received frame data is written into the shadow buffers. For details on receive shadow buffers, see *Section 46.6.6.3.5: Receive shadow buffer concept*. The data and status of the receive message buffers that are the CCRx or HLckCCRx are not modified in the reception slot.

46.6.6.3.3 Message buffer update

With the start of the next static or dynamic slot or with the start of the symbol window or NIT, the module triggers the slot or segment start transition SSS. This transition changes the state of the receiving receive message buffers from either CCRx to CCSu or from HLckCCRx to HLck, respectively.

If a message buffer is in the locked state HLckCCRx, no update will be performed. The received data is lost. This is indicated by setting the Frame Lost Channel A/B Error Flag FRLA_EF/FRLB_EF in the CHI Error Flag Register (FR_CHIERFR).

If a message buffer was in the CCRx state, it is now in the CCSu state. After the evaluation of the slot status provided by the PE, the message buffer is updated. The message buffer update depends on the slot status bits and the segment the message buffer is assigned to, as described in *Table 915*.

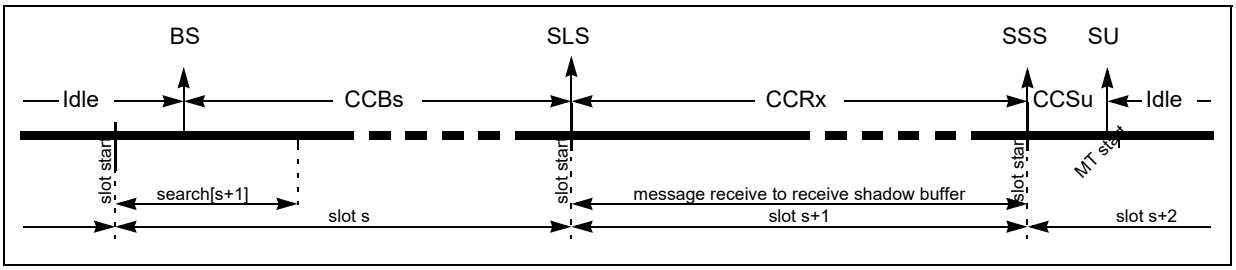
Table 915. Receive message buffer update

vSS!ValidFrame	vRF!Header!NFIndicator	Update description
1	1	Valid non-null frame received. – Message Buffer Data field updated. – Frame Header field updated. – Slot Status field updated. – DUP = 1 – DVAL = 1 – MBIF = 1
1	0	Valid null frame received. – Message Buffer Data field *not* updated. – Frame Header field *not* updated. – Slot Status field updated. – DUP = 0 – DVAL *not* changed – MBIF = 1
0	x	No valid frame received. – Message Buffer Data field not updated. – Frame Header field not updated. – Slot Status field updated. – DUP = 0 – DVAL *not* changed. – MBIF = 1, if the slot was not an empty dynamic slot. **Note:** An empty dynamic slot is indicated by the following frame and slot status bit values: vSS!ValidFrame = 0 and vSS!SyntaxError = 0 and vSS!ContentError = 0 and vSS!BViolation = 0.

Note: If the number of the last slot in the current communication cycle on a given channel is n, then all receive message buffers assigned to this channel with FR_MBFIDRn[FID] > n will not be updated at all.

When the receive message buffer update has finished the status updated transition SU is triggered, which changes the buffer state from CCSu to Idle. An example receive message buffer timing and state change diagram for a normal frame reception is given in *Figure 950*.

Figure 950. Message Reception timing



The amount of message data written into the message buffer data field of the receive shadow buffer is determined by the following items:

- The message buffer segment that the message buffer is assigned to, as defined by the Message Buffer Segment Size and Utilization Register (FR_MBSSUTR).
- The message buffer data field size, as defined by the related field of the Message Buffer Data Size Register (FR_MBDSR).
- The number of bytes received over the FlexRay bus.

If the message buffer is assigned to the message buffer segment 1, and the number of received bytes is greater than 2 ×FR_MBDSR.MBSEG1DS, the CC writes only 2 × FR_MBDSR.MBSEG1DS bytes into the message buffer data field of the receive shadow buffer. If the number of received bytes is less than 2 × FR_MBDSR.MBSEG1DS, the CC writes only the received number of bytes and will not change the trailing bytes in the message buffer data field of the receive shadow buffer. The same holds for the message buffer segment 2 with FR_MBDSR.MBSEG2DS.

46.6.6.3.4 Received message access

To access the message data received over the FlexRay bus, the application reads the message data stored in the message buffer data field of the corresponding receive message buffer. The access to the message buffer data field is described in *Section 46.6.3.1: Individual Message Buffers*.

The application can read the message buffer data field if the receive message buffer is one of the states HDis, HDisLck, or HLck. If the message buffer is in one of these states, the CC will not change the content of the message buffer.

46.6.6.3.5 Receive shadow buffer concept

The receive shadow buffer concept applies only to individual receive message buffers. The intention of this concept is to ensure that only syntactically and semantically valid received non-null frames are presented to the application in a receive message buffer. The basic structure of a receive shadow buffer is described in *Section 46.6.3.2: Receive shadow buffers*.

The receive shadow buffers temporarily store the received frame header and message data. The slot status information is generated after the slot boundary. If the slot status information indicates the reception of the valid non-null frame (see *Table 915*), the CC writes the slot status into the slot status field of the receive shadow buffer and exchanges the content of the Message Buffer Index Registers (FR_MBIDXR*n*) with the content of the corresponding internal shadow buffer index register. In all cases, the CC writes the slot status into the identified receive message buffer, depending on the slot status and the FlexRay segment the message buffer is assigned to.

The shadow buffer concept, with its index exchange, results in the fact that the FlexRay memory area located message buffer associated to an individual receive message buffer changes after successful reception of a valid frame. This means that the message buffer area in the FlexRay memory area accessed by the application for reading the received message is different from the initial setting of the message buffer. Therefore, the application must not rely on the index information written initially into the Message Buffer Index Registers (FR_MBIDXR*n*). Instead, the index of the message buffer header field must be fetched from the Message Buffer Index Registers (FR_MBIDXR*n*).

46.6.7 Individual message buffer search

This section provides a detailed description of the message buffer search algorithm.

The message buffer search determines for each enabled channel whether a slot *s* in a communication cycle *c* is assigned for frame or null frame transmission or if it is subscribed for frame reception on that channel.

The message buffer search is a sequential algorithm invoked at the following protocol-related events:

- NIT start
- Slot start in the static segment
- Minislot start in the dynamic segment

The message buffer search within the NIT searches for message buffers assigned or subscribed to slot 1. The message buffer search within slot *s* searches for message buffers assigned or subscribed to slot *s* + *1*.

In general, the message buffer search for the next slot *s* considers only message buffers that

1. Are enabled (that is, FR_MBCCSR*n*[EDS] = 1), and
2. Match the next slot *s*, (that is, FR_MBFIDR*n*[FID] = *s*)

On top of that, for the static segment, only those message buffers are considered that match the condition of at least one row of *Table 916*. For the dynamic segment, only those message buffers are considered that match the condition of at least one row of *Table 917*. These message buffers are called *matching* message buffers.

For each enabled channel, the message buffer search may identify multiple *matching* message buffers. Among all matching message buffers, the message buffers with highest priority according to *Table 916* for the static segment and according to *Table 917* for the dynamic segment are selected.

Table 916. Message buffer search priority (static segment)

Priority	MTD	LCKS	CMT	CCFM(1)	Description	Transition
(highest) 0	1	0	1	1	Transmit buffer, matches cycle count, not locked and committed	MA
1	1	—	0	1	Transmit buffer, matches cycle count, not committed	SA
	1	1	—	1	Transmit buffer, matches cycle count, locked	SA
2	1	—	—	—	Transmit buffer	SA
3	0	0	n/a	1	Receive buffer, matches cycle count, not locked	BS
(lowest) 4	0	1	n/a	1	Receive buffer, matches cycle count, locked	BS

1. Cycle Counter Filter Match, see *Section 46.6.7.1: Message buffer cycle counter filtering*.

Table 917. Message buffer search priority (dynamic segment)

Priority	MTD	LCKS	CMT	CCFM(1)	Description	Transition
(highest) 0	1	0	1	1	Transmit buffer, matches cycle count, not locked and committed	MA
1	0	0	n/a	1	Receive buffer, matches cycle count, not locked	BS
(lowest) 2	0	1	n/a	1	Receive buffer, matches cycle count, locked	BS

1. Cycle Counter Filter Match, see *Section 46.6.7.1: Message buffer cycle counter filtering*

If there are multiple message buffer with highest priority, the message buffer with the lowest message buffer number is selected. All message buffers that have the highest priority must have a consistent channel assignment as specified in *Section 46.6.7.2: Message buffer channel assignment consistency*.

Depending on the message buffer channel assignment, the same message buffer can be found for both channel A and channel B. In this case, this message buffer is used as described in *Section 46.6.3.1: Individual Message Buffers*.

46.6.7.1 Message buffer cycle counter filtering

The message buffer cycle counter filter is a value-mask filter defined by the CCFE, CCFMSK, and CCFVAL fields in the Message Buffer Cycle Counter Filter Registers (FR_MBCCFR*n*). This filter determines a set of communication cycles in which the message buffer is considered for message reception or message transmission. If the cycle counter filter is disabled (CCFE = 0), this set of cycles consists of all communication cycles.

If the cycle counter filter of a message buffer does not match a certain communication cycle number, this message buffer is not considered for message transmission or reception in that communication cycle. In case of a transmit message buffer assigned to a slot in the static segment, though, this buffer is added to the matching message buffers to indicate the slot assignment and to trigger the null frame transmission.

The cycle counter filter of a message buffer matches the communication cycle with the number CYCCNT if at least one of the following conditions evaluates to true:

Equation 50

$$MBCCFRn[CCFE] = 0$$

Equation 51

$$CYCCNT \& MBCCFRn[CCFMSK] = MBCCFRn[CCFVAL] \& MBCCFRn[CCFMSK]$$

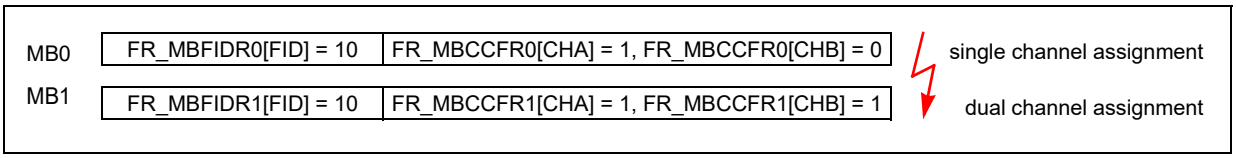
46.6.7.2 Message buffer channel assignment consistency

The message buffer channel assignment given by the CHA and CHB bits in the Message Buffer Cycle Counter Filter Registers (FR_MBCCFR*n*) defines the channels on which the message buffer will receive or transmit. The message buffer with number *n* transmits or receives on channel A if FR_MBCCFR*n*[CHA] = 1 and transmits or receives on channel B if FR_MBCCFR*n*[CHB] = 1.

To ensure correct message buffer operation, all message buffers assigned to the same slot and with the same priority must have a *consistent* channel assignment. That means they

must be either assigned to one channel only, or must be assigned to *both* channels. The behavior of the message buffer search is not defined if both types of channel assignments occur for one slot and priority. An inconsistent channel assignment for message buffer 0 and message buffer 1 is depicted in *Figure 951*.

Figure 951. Inconsistent channel assignment



46.6.7.3 Node-related slot multiplexing

The term *node-related slot multiplexing* applies to the dynamic segment only and refers to the functionality if transmit as well as receive message buffers are configured for the same slot.

According to *Table 917* the transmit buffer is only found if the cycle counter filter matches, and the buffer is not locked and committed. In all other cases, the receive buffer will be found. Thus, if the block has no data to transmit in a dynamic slot, it is able to receive frames on that slot.

46.6.7.4 Message buffer search error

There are two kinds of errors that may occur during message buffer search[o].

46.6.7.4.1 Message buffer search start while running

If the message buffer search is running in slot *s – 1* and the next message buffer search start event appears due to the start of slot *s*, the message buffer search engine is stopped and the Message Buffer Search Error Flag MBS_EF is set in the CHI Error Flag Register (FR_CHIERFR). As a result of this stop, no individual message buffer is identified for transmission or reception in slot *s*. Additionally, the search engine will not be started in slot *s*, and consequently no individual message buffer is identified for transmission or reception in slot *s + 1*.

A message buffer search error appears only if the CHI frequency is too slow to allow the search through all message buffers to be completed within the NIT or a minislot.

For more details of minimum required CHI frequency see *Section 46.7.5: Number of usable message buffers*.

46.6.7.4.2 Illegal message buffer index found

If the message buffer search has finished the message buffer search in slot *s – 1*, it retrieves the data offset values for the found message buffers and the receive shadow buffers. If one of these message buffers contains an illegal message buffer index, the Message Buffer Search Error Flag MBS_EF is set in the CHI Error Flag Register (FR_CHIERFR) and no individual message buffer is identified for transmission or reception in slot *s*. The legal message buffer index values for the individual and receive shadow

o. The FIFO reception is not affected by the search errors. Additionally, if no rx buffer has been found due to an search error, the received frame is considered for FIFO reception.

buffers are specified in *Section 46.5.2.52: Receive Shadow Buffer Index Register (FR_RSBIR)* and *Section 46.5.2.84: Message Buffer Index Registers (FR_MBIDXRn)*.

46.6.8 Individual message buffer reconfiguration

The initial configuration of each individual message buffer can be changed even when the protocol is not in the *POC:config* state. This is referred to as individual message buffer *reconfiguration*. The configuration bits and fields that can be changed are given in the section on Specific Configuration Data. The common configuration data given in the section on Specific Configuration Data can not be reconfigured when the protocol is out of the *POC:config* state.

46.6.8.1 Reconfiguration schemes

Depending on the target and destination basic state of the message buffer that is to be reconfigured, there are three reconfiguration schemes.

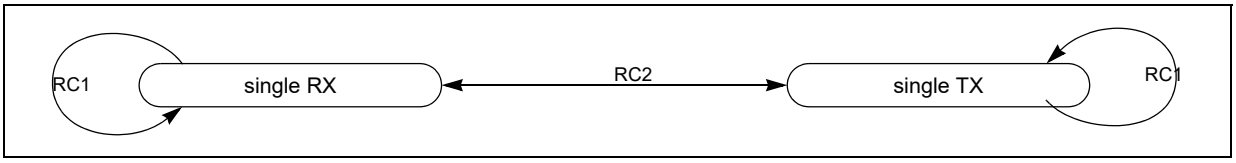
46.6.8.1.1 Basic type not changed (RC1)

A reconfiguration will not change the basic type of the individual message buffer, if the message buffer transfer direction bit FR_MBCCSR*n*[MTD] is not changed. This type of reconfiguration is denoted by RC1 in *Figure 952*. Transmit and receive message buffers can be RC1-reconfigured when in the HDis or HDisLck state.

46.6.8.1.2 Buffer type not changed (RC2)

A reconfiguration will not change the buffer type of the individual message buffer. This type of reconfiguration is denoted by RC2 in *Figure 952*. It applies to transmit and receive message buffers. Transmit and receive message buffers can be RC2-reconfigured when in the HDis or HDisLck state.

Figure 952. Message buffer reconfiguration scheme



46.6.9 Receive FIFOs

This section provides the functional description of the two receive FIFOs.

46.6.9.1 Overview

The two receive FIFOs implement the queued message buffer concept defined by the *FlexRay Communications System Protocol Specification, Version 2.1 Rev A.* One FIFO is assigned to channel A, and the other FIFO is assigned to channel B. Both FIFOs work independently from each other.

The message buffer structure of each FIFO is described in *Section 46.6.3.3: Receive FIFO*. The area in the FlexRay memory area for each of the two FIFOs is characterized by:

- The FIFO system memory base address
- The index of the first FIFO entry given by the Receive FIFO Start Index Register (FR_RFSIR)
- The data field offset of the data field belonging to the first FIFO entry given by the Receive FIFO Start Data Offset Register (FR_RFSDOR)
- The number of FIFO entries and the length of each FIFO entry as given by the Receive FIFO Depth and Size Register (FR_RFDSR)

46.6.9.2 FIFO configuration

The FIFOs can be configured for two different locations of the system memory base address via the FIFO Address Mode bit FAM in the Module Configuration Register (FR_MCR).

46.6.9.2.1 Single System Memory Base Address mode

This mode is configured when the FIFO address mode flag FR_MCR[FAM] is set to 0. In this mode, the location of the system memory base address for the FIFO buffers is the System Memory Base Address Register (FR_SYMBADR).

46.6.9.2.2 Dual System Memory Base Address mode

This mode is configured when the FIFO address mode flag FR_MCR[FAM] is set to 1. In this mode, the location of the system memory base address for the FIFO buffers is the Receive FIFO System Memory Base Address Register (FR_RFSYMBADR).

The FIFO control and configuration data is given in *Section 46.6.3.7: Receive FIFO control and configuration data*. The configuration of the FIFOs consists of two steps.

The first step is the allocation of the required amount of FlexRay memory area for the FlexRay window. This includes the allocation of the message buffer header area and the allocation of the message buffer data fields. For more details, see *Section 46.6.4: FlexRay memory area layout*.

The second step is the programming of the configuration data register while the PE is in *POC:config*.

The following steps configure the layout of the FIFO.

1. Configure the FIFO update and address modes in Module Configuration Register (FR_MCR).
2. Configure the FIFO system memory base address.
3. Configure the Receive FIFO Start Index Register (FR_RFSIR) with the first message buffer header index that belongs to the FIFO.
4. Configure the Receive FIFO Start Data Offset Register (FR_RFSDOR) with the data field offset of the data field belonging to the first message buffer that belongs to the FIFO.
5. Configure the Receive FIFO Depth and Size Register (FR_RFDSR) with FIFO entry size.
6. Configure the Receive FIFO Depth and Size Register (FR_RFDSR) with FIFO depth.
7. Configure the FIFO filters.

46.6.9.3 FIFO periodic timer

The FIFO periodic timer is used to generate a FIFO almost-full interrupt at certain point in time, if the almost-full watermark is not reached but the FIFO is not empty. This can be used to prevent frames from getting stuck in the FIFO for a long time.

The FIFO periodic timer is configured via the Receive FIFO Periodic Timer Register (FR_RFPTR). If the periodic timer duration FR_RFPTR[PTD] is configured to 0x0000, the periodic timer is continuously expired. If the periodic timer duration FR_RFPTR[PTD] is configured to 0x3FFF, the periodic timer never expires. If the periodic timer is configured to a value *ptd,* greater than 0x0000 and smaller 0x3FFF, the periodic timer expires and is restarted at the start of every communication cycle, and expires and is restarted after *ptd* macroticks have been elapsed.

46.6.9.4 FIFO reception

The FIFO reception is a CC internal operation.

A message frame reception is directed into the FIFO if no individual message buffer is assigned for transmission or subscribed for reception for the current slot. In this case, the FIFO filter path shown in *Figure 953* is activated.

If the FIFO filter path indicates that the received frame has to be appended to the FIFO and the FIFO is not full, the CC writes the received frame header into the message buffer header field indicated by the CC internal FIFO write index. The frame payload data are written into the corresponding message buffer data field. If the status of the received frame indicates a valid non-null frame, the slot status information is written into the message buffer header field and the CC internal FIFO write index is updated by 1 and the fifo fill level FLA (FLB) in the Receive FIFO Fill Level and POP Count Register (FR_RFFLPCR) is incremented.If the status of the received frame indicates an invalid or null frame, the frame is not appended to the FIFO.

46.6.9.5 FIFO almost-full interrupt generation

If the FIFO fill level FLA (FLB) is updated after a frame reception and exceeds the FIFO watermark level WM (that is, FLA > WM_A (FLB > WM_B)), then the FIFO almost-full interrupt flag FR_GIFER[FAFAIF] (FR_GIFER[FAFBIF]) is asserted.If the periodic timer expires, and FIFOA (FIFOB) is not empty (that is, FLA > 0 (FLB > 0), then the FIFO almost-full interrupt flag FR_GIFER[FAFAIF] (FR_GIFER[FAFBIF]) is asserted.

46.6.9.6 FIFO overflow error generation

If the FIFOA (FIFOB) is full (that is, FLA = $FIFO_DEPTH_A$ (FLB=$FIFO_DEPTH_B$)) and the conditions for a FIFO reception as described in *Section 46.6.9.4: FIFO reception* are fulfilled, then the FIFO overflow error flag FR_CHIERFR[FOVA_EF] (FR_CHIERFR[FOVB_EF]) is asserted.

46.6.9.7 FIFO message access

The FIFOA (FIFOB) contains valid messages if the FIFO fill level given in the fields FLA (FLB) in the Receive FIFO Fill Level and POP Count Register (FR_RFFLPCR) is greater than 0. The Receive FIFO A Read Index Register (FR_RFARIR) and the (Receive FIFO B Read Index Register (FR_RFBRIR)) point to a message buffer with valid content and the oldest frames stored in the FIFO.The respective read data field offsets can be calculated according to *Equation 46*.

If the FIFO fill level FLA (FLB) in the Receive FIFO Fill Level and POP Count Register (FR_RFFLPCR) is 0, then the FIFOA (FIFOB) contains no valid messages and the corresponding read index register Receive FIFO A Read Index Register (FR_RFARIR) or (Receive FIFO B Read Index Register (FR_RFBRIR) point to a message buffer with invalid content. In this case, the application must not read data from this FIFO.

To access the oldest message in the FIFOA (FIFOB), the application first reads the read index RDIDX out of the Receive FIFO A Read Index Register (FR_RFARIR) (Receive FIFO B Read Index Register (FR_RFBRIR)). This read index points to the message buffer header field of the oldest message buffer that contains valid received message data. The data field offset belonging to this message buffer must be calculated by the application according to *Equation 46*. The application can access the message data as described in *Section 46.6.3.3: Receive FIFO*. When the application has read the message buffer data and status information, it can update the FIFO as described in *Section 46.6.9.8: FIFO update*.

46.6.9.8 FIFO update

The application updates the FIFOA (FIFOB) by writing a pop count value *pc* different from 0 to the PCA (PCB) field in the Receive FIFO Fill Level and POP Count Register (FR_RFFLPCR).

As a result of the this operation, the CC removes the oldest *pc* entries from FIFOA (FIFOB).

If the specified pop count value *pc* is greater than the current fill level *fl* provided in FLA (FAB) field, then only *fl* entries are removed from the FIFOA (FIFOB), the remaining *fl-pc* requested pop operations are discarded without any notification. In this case FIFOA (FIFOB) is empty after the update operation.

The read index in the Receive FIFO A Read Index Register (FR_RFARIR) (Receive FIFO B Read Index Register (FR_RFBRIR)) is incremented by the number of removed items. If the read index reaches the top of the FIFO, it wraps around to the FIFO start index defined in Receive FIFO Start Index Register (FR_RFSIR) automatically.

46.6.9.8.1 FIFO interrupt flag update

The FIFO Interrupt Flag Update mode is configured when the FIFO update mode flag FR_MCR[FUM] is set to 0. In this mode, FIFOA (FIFOB) will be updated by one entry when the interrupt flag FR_GIFER[FAFAIF] (FR_GIFER[FAFBIF]) is written with 1 by the application.

If the FIFO is empty, the update request is ignored without any notification.

The read index in the Receive FIFO A Read Index Register (FR_RFARIR) (Receive FIFO B Read Index Register (FR_RFBRIR)) is incremented by 1 if the FIFO was not empty. If the read index reaches the top of the FIFO, it wraps around to the FIFO start index automatically.

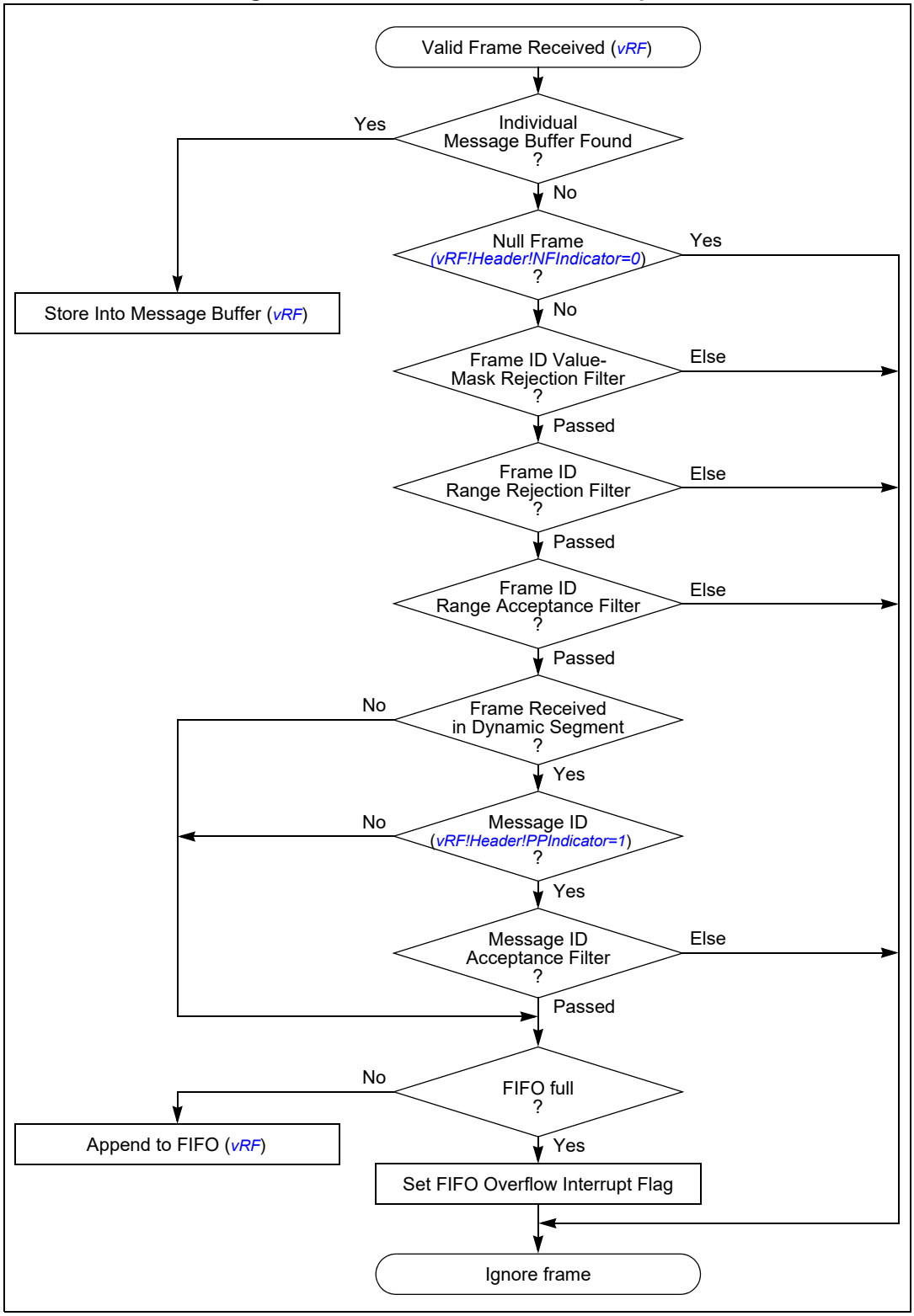
46.6.9.9 FIFO filtering

The FIFO filtering is activated after all enabled individual receive message buffers have been searched without success for a message buffer to receive the current frame.

The CC provides four sets of FIFO filters. The FIFO filters are applied to valid non-null frames only. The FIFO will not receive invalid or null-frames. For each FIFO filter, the pass

criteria is specified in the related section given below. Only frames that have passed all filters will be appended to the FIFO. The FIFO filter path is depicted in *Figure 953*.

Figure 953. Received frame FIFO filter path



A received frame passes the FIFO filtering if it has passed all four type of filter.

46.6.9.9.1 RX FIFO Frame ID Value-mask Rejection filter

The Frame ID Value-mask Rejection filter is a value-mask filter and is defined by the fields in the Receive FIFO Frame ID Rejection Filter Value Register (FR_RFFIDRFVR) and the Receive FIFO Frame ID Rejection Filter Mask Register (FR_RFFIDRFMR). Each received frame with a frame ID FID that does not match the value-mask filter value passes the filter (that is, it is not rejected).

Consequently, a received valid frame with the frame ID FID passes the RX FIFO Frame ID Value-mask Rejection filter if *Equation 52* is fulfilled.

Equation 52

$$FID \,\&\, FR_RFFIDRFMR[FIDRFMSK] \neq FR_RFFIDRFVR[FIDRFVAL] \,\&\, FR_RFFIDRFMR[FIDRFMSK]$$

The RX FIFO Frame ID Value-mask Rejection filter can be configured to pass all frames by using the following settings:

- FR_RFFIDRFVR[FIDRFVAL] = 0x000 and FR_RFFIDRFMR[FIDRFMSK] = 0x7FF

Using the settings above, only the frame with frame ID 0 will be rejected, which is an invalid frame. All other frames will pass.

The RX FIFO Frame ID Value-mask Rejection filter can be configured to reject all frames by using the following settings:

- FR_RFFIDRFMR[FIDRFMSK] = 0x000

Using the settings above, *Equation 52* can never be fulfilled (0!= 0) and thus all frames are rejected; no frame will pass. This is the reset value for the RX FIFO.

46.6.9.9.2 RX FIFO Frame ID Range Rejection filter

Each of the four RX FIFO Frame ID Range filters can be configured as a rejection filter. The filters are configured by the Receive FIFO Range Filter Configuration Register (FR_RFRFCFR) and controlled by the Receive FIFO Range Filter Control Register (FR_RFRFCTR). The RX FIFO frame ID range filters apply to all received valid frames. A received frame with the frame ID FID passes the RX FIFO Frame ID Range Rejection filters if either no rejection filter is enabled, or, if for all of the enabled RX FIFO Frame ID Range Rejection filters (that is, FR_RFRFCTR[FiMD] = 1 and FR_RFRFCTR[FiEN] = 1), *Equation 53* is fulfilled.

Equation 53

$$(FID < FR_RFRFCFR_{SEL}[SID_{IBD=0}]) \text{ or } (FR_RFRFCFR_{SEL}[SID_{IBD=1}] < FID)$$

Consequently, all frames with a frame ID that fulfills *Equation 54* for at least one of the enabled rejection filters will be rejected and thus not pass.

Equation 54

$$FR_RFRFCFR_{SEL}[SID_{IBD=0}] \leq FID \leq FR_RFRFCFR_{SEL}[SID_{IBD=1}]$$

46.6.9.9.3 RX FIFO Frame ID Range Acceptance filter

Each of the four RX FIFO Frame ID Range filters can be configured as an acceptance filter. The filters are configured by the Receive FIFO Range Filter Configuration Register (FR_RFRFCFR) and controlled by the Receive FIFO Range Filter Control Register (FR_RFRFCTR). The RX FIFO Frame ID range filters apply to all received valid frames. A received frame with the frame ID FID passes the RX FIFO Frame ID Range acceptance filters if either no acceptance filter is enabled, or, if for at least one of the enabled RX FIFO Frame ID Range acceptance filters (that is, FR_RFRFCTR[FiMD] = 0 and FR_RFRFCTR[FiEN] = 1), *Equation 55* is fulfilled.

Equation 55

$$FR_RFRFCFR_{SEL}[SID_{IBD=0}] \leq FID \leq FR_RFRFCFR_{SEL}[SID_{IBD=1}]$$

46.6.9.9.4 RX FIFO Message ID Acceptance filter

The RX FIFO Message ID Acceptance filter is a value-mask filter and is defined by the Receive FIFO Message ID Acceptance Filter Value Register (FR_RFMIDAFVR) and the Receive FIFO Message ID Acceptance Filter Mask Register (FR_RFMIDAFMR). This filter applies only to valid frames received in the dynamic segment with the payload preamble indicator bit PPI set to 1. All other frames will pass this filter.

A received valid frame in the dynamic segment with the payload preamble indicator bit PPI set to 1 and with the message ID MID (the first two bytes of the payload) will pass the RX FIFO Message ID Acceptance filter if *Equation 56* is fulfilled.

Equation 56

$$MID \,\&\, FR_RFMIDAFMR[MIDAFMSK] = FR_RFMIDAFMR[MIDAFVAL] \,\&\, FR_RFMIDAFMR[MIDAFMSK]$$

The RX FIFO Message ID Acceptance filter can be configured to accept all frames by setting

- FR_RFMIDAFMR[MIDAFMSK]:= 0x000

Using the settings above, *Equation 56* is always fulfilled and all frames will pass.

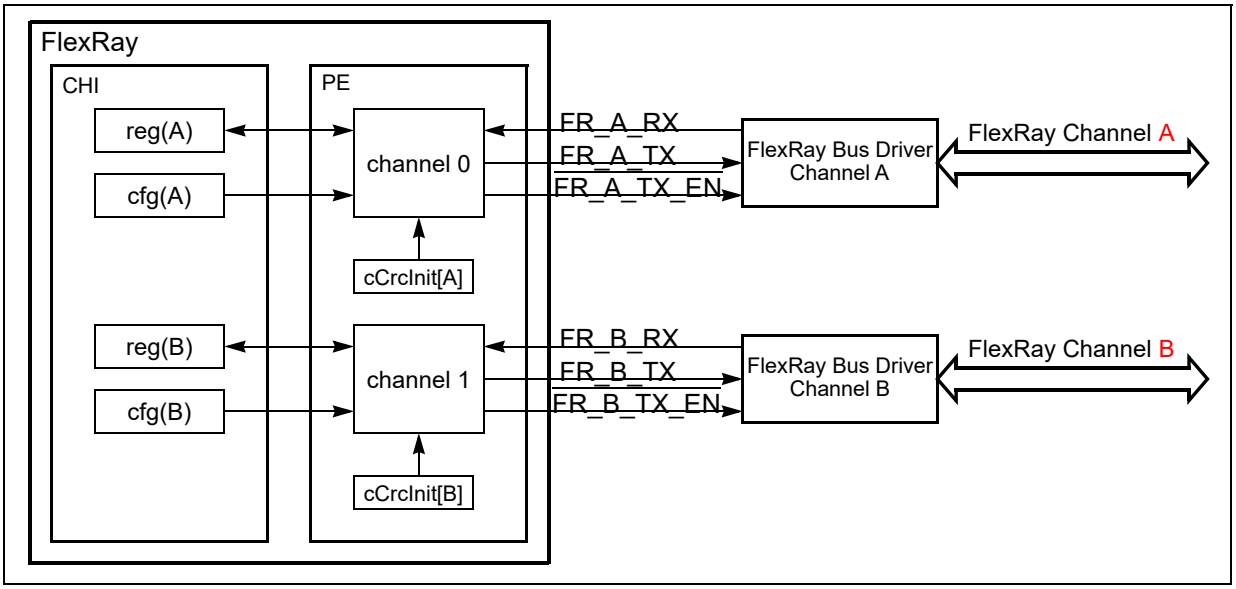
46.6.10 Channel device modes

This section describes the two FlexRay channel device modes that are supported by the CC.

46.6.10.1 Dual-Channel Device mode

In Dual-Channel Device mode, both FlexRay ports are connected to physical FlexRay bus lines. The FlexRay port consisting of FR_A_RX, FR_A_TX, and FR_A_TX_EN is connected to the physical bus channel A and the FlexRay port consisting of FR_B_RX, FR_B_TX, and FR_B_TX_EN is connected to the physical bus channel B. The dual-channel system is shown in *Figure 954*.

Figure 954. Dual-Channel Device mode



46.6.10.2 Single-Channel Device mode

Single-Channel Device mode supports devices that have only one FlexRay port available. This FlexRay port consists of the signals FR_A_RX, FR_A_TX, and FR_A_TX_EN and can be connected to either the physical bus channel A (shown in *Figure 955*) or the physical bus channel B (shown in *Figure 956*).

If the device is configured as a single-channel device by setting FR_MCR[SCM] to 1, only the internal channel A and FlexRay port A are used. Depending on the setting of FR_MCR[CHA] and FR_MCR[CHB], the internal channel A behaves either as a FlexRay channel A or FlexRay channel B. The bit FR_MCR[CHA] must be set, if FlexRay port A is connected to FlexRay channel A. The bit FR_MCR[CHB] must be set if FlexRay port B is connected to FlexRay channel B. The two FlexRay channels differ only in the initial value for the frame CRC *cCrcInit*. For a single-channel device, the application can access and configure only the registers related to internal channel A.

Figure 955. Single-Channel Device mode (channel A)

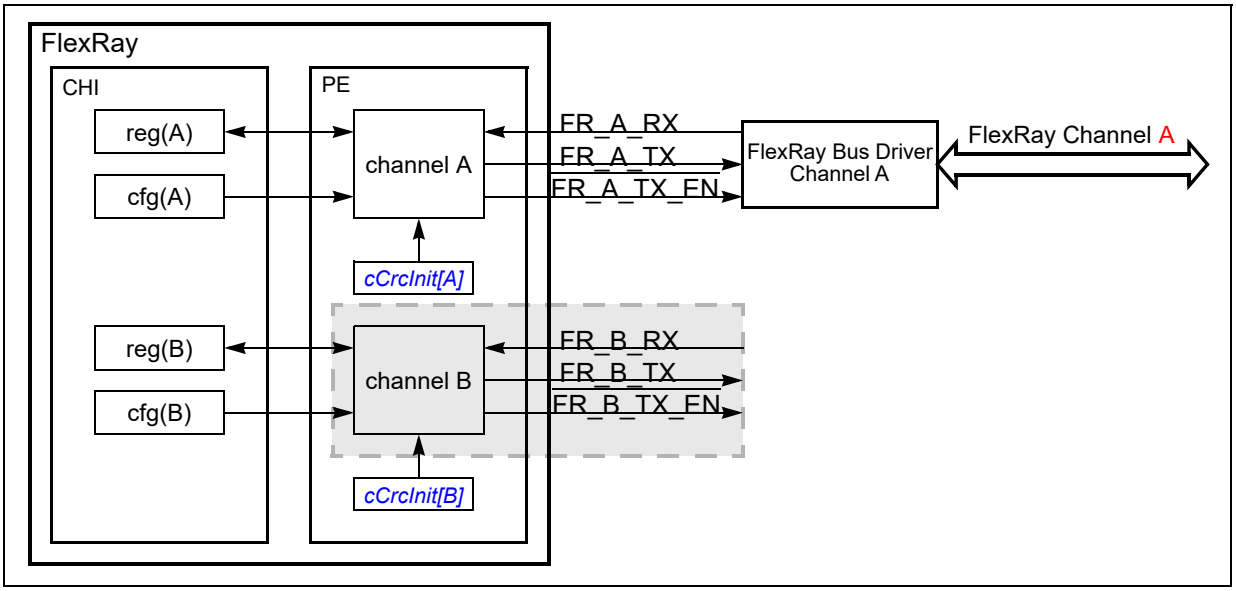
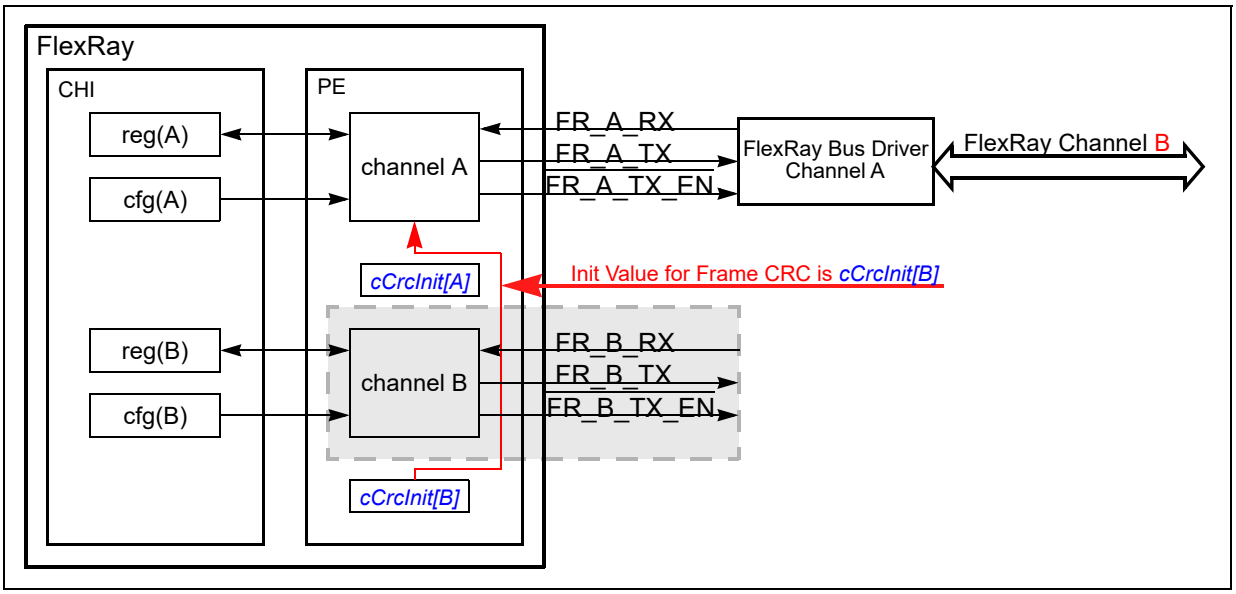


Figure 956. Single-Channel Device mode (channel B)



46.6.11 External clock synchronization

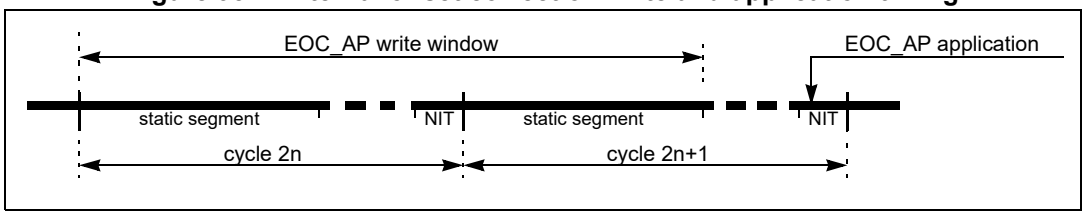
The application of the external rate and offset correction is triggered when the application writes to the EOC_AP and ERC_AP fields in the Protocol Operation Control Register (FR_POCR). The PE applies the external correction values in the next even-odd cycle pair as shown in *Figure 957* and *Figure 958*.

Note: The values provided in the EOC_AP and ERC_AP fields are the values that were written from the application most recently. If these value were already applied, they will not be applied in the current cycle pair again.

If the offset correction applied in the NIT of cycle 2n + 1 will be affected by the external offset correction, the EOC_AP field must be written to after the start of cycle 2n and before the end

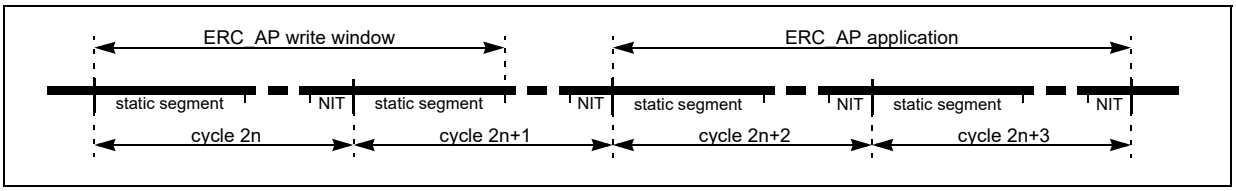
of the static segment of cycle 2n + 1. If this field is written to after the end of the static segment of cycle 2n + 1, it is not guaranteed that the external correction value is applied in cycle 2n + 1. If the value is not applied in cycle 2n + 1, then the value will be applied in the cycle 2n+3. Refer to *Figure 957* for timing details.

Figure 957. External offset correction write and application timing



If the rate correction for the cycle pair [2n + 2, 2n + 3] will be affected by the external offset correction, the ERC_AP field must be written to after the start of cycle 2n and before the end of the static segment start of cycle 2n + 1. If this field is written to after the end of the static segment of cycle 2n + 1, it is not guaranteed that the external correction value is applied in cycle pair [2n + 2, 2n + 3]. If the value is not applied for cycle pair [2n + 2, 2n + 3], then the value will be applied for cycle pair [2n+4, 2n+5]. Refer to *Figure 958* for details.

Figure 958. External rate correction write and application timing



46.6.12 Sync frame ID and sync frame deviation tables

The FlexRay protocol requires the provision of a snapshot of the synchronization frame ID tables for the even and odd communication cycle for both channels. The CC provides the means to write a copy of these internal tables into the FlexRay memory area and ensures application access to consistent tables by means of table locking. Once the application has locked the table successfully, the CC will not overwrite these tables and the application can read a consistent snapshot.

Note: Only synchronization frames that have passed the synchronization frame filters are considered for clock synchronization and appear in the sync frame tables.

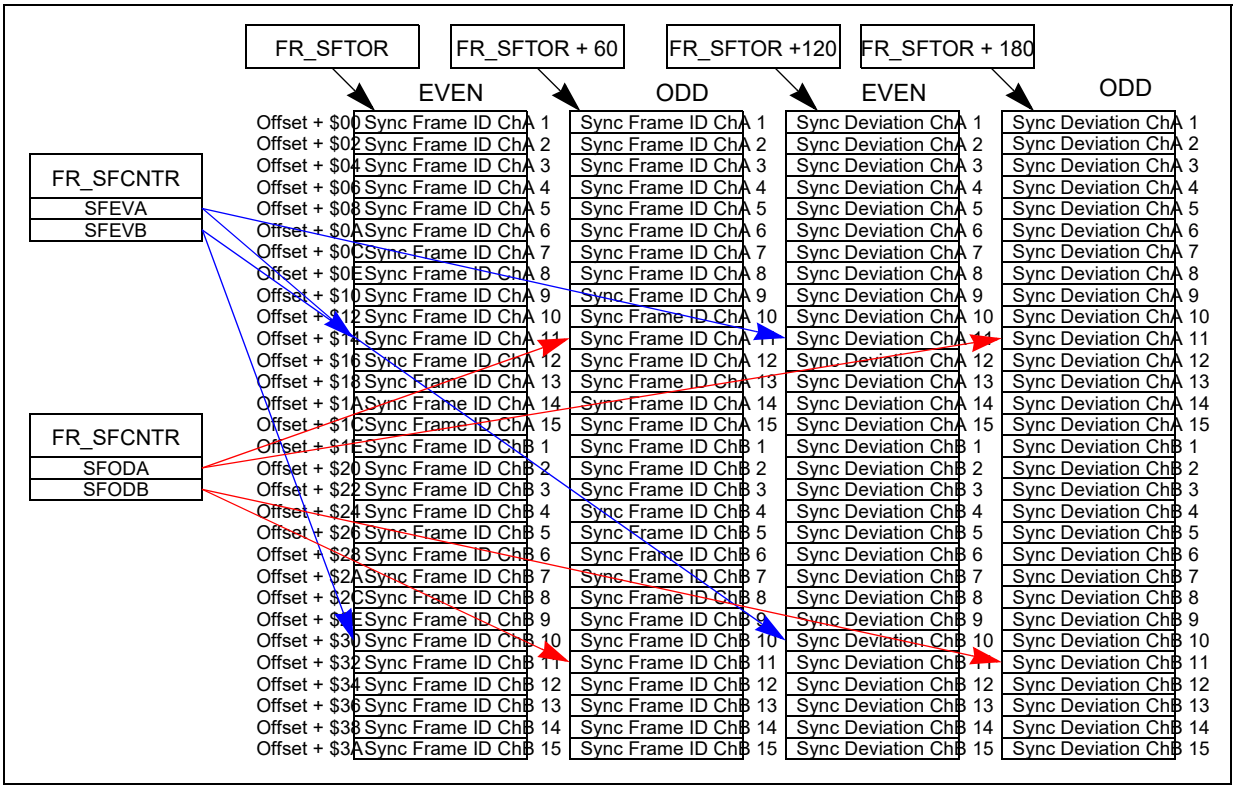
46.6.12.1 Sync frame ID table content

The sync frame ID table is a snapshot of the protocol-related variables *vsSyncIdListA* and *vsSyncIdListB* for each even and odd communication cycle. This table provides a list of the frame IDs of the synchronization frames received on the corresponding channel and cycle that are used for the clock synchronization.

46.6.12.2 Sync frame deviation table content

The sync frame deviation table is a snapshot of the protocol-related variable zsDev(id)(oe)(ch)!Value. Each sync frame deviation table entry provides the deviation value for the sync frame, with the frame ID presented in the corresponding entry in the sync frame ID table.

Figure 959. Sync table memory layout



46.6.12.3 Sync frame ID and sync frame deviation table setup

The CC writes a copy of the internal synchronization frame ID and deviation tables into the FlexRay memory area if requested by the application. The application must provide the appropriate amount of FlexRay memory area for the tables. The memory layout of the tables is given in *Figure 959*. Each table occupies 120 16-bit entries.

While the protocol is in *POC:config* state, the application must program the offsets for the tables into the Sync Frame Table Offset Register (FR_SFTOR).

46.6.12.4 Sync frame ID and sync frame deviation table generation

The application controls the generation process of the sync frame ID and sync frame deviation tables into the FlexRay memory area using the Sync Frame Table Configuration, Control, Status Register (FR_SFTCCSR). A summary of the copy modes is given in *Table 918*.

Table 918. Sync frame table generation modes

FR_SFTCCSR			Description
OPT	SDVEN	SIDEN	
0	0	0	No sync frame table copy
0	0	1	Sync frame ID tables will be copied continuously
0	1	0	Reserved

Table 918. Sync frame table generation modes (continued)

FR_SFTCCSR			Description
OPT	SDVEN	SIDEN	
0	1	1	Sync frame ID tables and sync frame deviation tables will be copied continuously
1	0	0	No sync frame table copy
1	0	1	Sync frame ID tables for next even-odd-cycle pair will be copied
1	1	0	Reserved
1	1	1	Sync frame ID tables and sync frame deviation tables for next even-odd-cycle pair will be copied

The sync frame table generation process is described in the following for the even cycle. The same sequence applies to the odd cycle.

If the application has enabled the sync frame table generation by setting FR_SFTCCSR[SIDEN] to 1, the CC starts the update of the even-cycle-related tables after the start of the NIT of the next even cycle. The CC checks whether the application has locked the tables by reading the FR_SFTCCSR[ELKS] lock status bit. If this bit is set, the CC will not update the table in this cycle. If this bit is cleared, the CC locks this table and starts the table update. To indicate that these tables are currently updated and may contain inconsistent data, the CC clears the even table valid status bit FR_SFTCCSR[EVAL]. When all table entries related to the even cycle have been transferred into the FlexRay memory area, the CC sets the even table valid bit FR_SFTCCSR[EVAL] and the Even Cycle Table Written Interrupt Flag EVT_IF in the Protocol Interrupt Flag Register 1 (FR_PIFR1). If the interrupt enable flag EVT_IE is set, an interrupt request is generated.

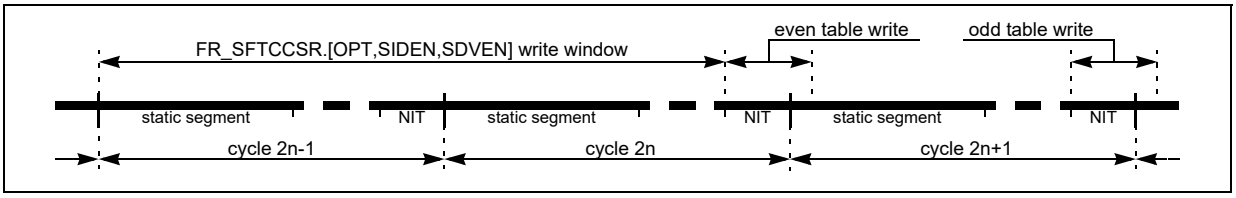
To read the generated tables, the application must lock the tables to prevent the CC from updating them. The locking is initiated by writing a 1 to the Even Table Lock trigger, FR_SFTCCSR[ELKT]. When the even table is not currently updated by the CC, the lock is granted and the even table lock status bit FR_SFTCCSR[ELKS] is set. This indicates that the application has successfully locked the even sync tables and the corresponding status information fields, SFRA and SFRB, in the Sync Frame Counter Register (FR_SFCNTR). The value in the FR_SFTCCSR[CYCNUM] field provides the number of the cycle that this table is related to.

The number of available table entries per channel is provided in the FR_SFCNTR[SFEVA] and FR_SFCNTR[SFEVB] fields. The application can now start to read the sync table data from the locations given in *Figure 959*.

After reading all the data from the locked tables, the application must unlock the table by writing to the Even Table Lock trigger FR_SFTCCSR[ELKT] again. The even table lock status bit FR_SFTCCSR[ELKS] is reset immediately.

If the sync frame table generation is disabled, the table valid bits FR_SFTCCSR[EVAL] and FR_SFTCCSR[EVAL] are reset when the counter values in the Sync Frame Counter Register (FR_SFCNTR) are updated. This is done because the tables stored in the FlexRay memory area are no longer related to the values in the Sync Frame Counter Register (FR_SFCNTR).

Figure 960. Sync frame table trigger and generation timing



46.6.12.5 Sync frame table access

The sync frame tables will be transferred into the FlexRay memory area during the table write windows shown in *Figure 959*. During the table write, the application can not lock the table that is currently written. If the application locks the table outside of the table write window, the lock is granted immediately.

46.6.12.5.1 Sync Frame Table Locking and Unlocking

The application locks the even/odd sync frame table by writing 1 to the lock trigger bit ELKT/OLKT in the Sync Frame Table Configuration, Control, Status Register (FR_SFTCCSR). If the affected table is not currently written to the FlexRay memory area, the lock is granted immediately, and the lock status bit ELKS/OLKS is set. If the affected table is currently written to the FlexRay memory area, the lock is not granted. In this case, the application must issue the lock request again until the lock is granted.

The application unlocks the even/odd sync frame table by writing 1 to the lock trigger bit ELKT/OLKT. The lock status bit ELKS/OLKS is cleared immediately.

46.6.13 MTS generation

The CC provides a flexible means to request the transmission of the Media Access Test Symbol (MTS) in the symbol window on channel A or channel B.

The application can configure the set of communication cycles that transmit the MTS over the FlexRay bus by programming the CYCCNTMSK and CYCCNTVAL fields in the MTS A Configuration Register (FR_MTSACFR) and MTS B Configuration Register (FR_MTSBCFR).

The application enables or disables the generation of the MTS on either channel by setting or clearing the MTE control bit in the MTS A Configuration Register (FR_MTSACFR) or MTS B Configuration Register (FR_MTSBCFR). If an MTS is to be transmitted in a certain communication cycle, the application must set the MTE control bit during the static segment of the preceding communication cycle.

The MTS is transmitted over channel A in the communication cycle with number CYCCNT if *Equation 57*, *Equation 58*, and *Equation 59* are fulfilled.

Equation 57

FR_PSR0[PROTSTATE] = *POC:normal active*

Equation 58

FR_MTSACRF[MTE] = 1

Equation 59

CYCCNT & FR_MTSACFR[CYCCNTMSK] = FR_MTSACFR[CYCCNTVAL] & FR_MTSACFR[CYCCNTMSK]

The MTS is transmitted over channel B in the communication cycle with number CYCCNT if *Equation 57*, *Equation 60* and *Equation 61* are fulfilled.

Equation 60

$$FR_MTSBCRF[MTE] = 1$$

Equation 61

$$CYCCNT \& FR_MTSBCFR[CYCCNTMSK] = FR_MTSBCFR[CYCCNTVAL] \& FR_MTSBCFR[CYCCNTMSK]$$

46.6.14 Key slot transmission

46.6.14.1 Key slot assignment

A key slot is assigned to the CC if the key_slot_id field in the Protocol Configuration Register 18 (FR_PCR18) is configured with a value greater than 0 and less than or equal to number_of_static_slots in Protocol Configuration Register 2 (FR_PCR2); otherwise no key slot is assigned.

46.6.14.2 Key slot transmission in *POC:startup*

If a key slot is assigned and the CC is in the *POC:startup* state, startup null frames will be transmitted as specified by FlexRay Communications System Protocol Specification, Version *2.1 Rev A*.

46.6.14.3 Key slot transmission in *POC:normal active*

If a key slot is assigned and the CC is in *POC:normal active*, a frame of the type as shown in *Table 919* is transmitted. If a transmit message buffer is configured for the key slot and a valid message is available, a message frame is transmitted (see *Section 46.6.6.2.5*). If no transmit message buffer is configured for the key slot or no valid message is available, a null frame is transmitted (see *Section 46.6.6.2.6*).

Table 919. Key slot frame type

FR_PCR11[key_slot_used_for_sync]	FR_PCR11[key_slot_used_for_startup]	key slot frame type
0	0	Normal frame
0	1	Normal frame(1)
1	0	Sync frame
1	1	Startup frame

1. The frame transmitted has a semantically incorrect header and will be detected as an invalid frame at the receiver.

46.6.15 Sync frame filtering

Each received synchronization frame must pass the Sync Frame Acceptance filter and the Sync Frame Rejection filter before it is considered for clock synchronization. If the synchronization frame filtering is globally disabled (that is, the SFFE control bit in the Module Configuration Register (FR_MCR) is cleared), all received synchronization frames are considered for clock synchronization. If a received synchronization frame did not pass at

least one of the two filters, this frame is processed as a normal frame and is not considered for clock synchronization.

46.6.15.1 Sync frame acceptance filtering

The synchronization frame acceptance filter is implemented as a value-mask filter. The value is configured in the Sync Frame ID Acceptance Filter Value Register (FR_SFIDAFVR) and the mask is configured in the Sync Frame ID Acceptance Filter Mask Register (FR_SFIDAFMR). A received synchronization frame with the frame ID FID passes the sync frame acceptance filter, if *Equation 62* or *Equation 63* evaluates to true.

Equation 62

$$FR_MCR[SFFE] = 0$$

Equation 63

$$FID \,\&\, FR_SFIDAFMR[FMSK] = FR_SFIDAFVR[FVAL] \,\&\, FR_SFIDAFMR[FMSK]$$

Note: Sync frames are transmitted in the static segment only. Thus FID ≤ 1023.

46.6.15.2 Sync Frame Rejection Filtering

The synchronization frame rejection filter is a comparator. The compare value is defined by the Sync Frame ID Rejection Filter Register (FR_SFIDRFR). A received synchronization frame with the frame ID FID passes the sync frame rejection filter if *Equation 64* or *Equation 65* evaluates to true.

Equation 64

$$FR_MCR[SFFE] = 0$$

Equation 65

$$FID \neq FR_SFIDRFR[SYNFRID]$$

Note: Sync frames are transmitted in the static segment only. Thus FID ≤ 1023.

46.6.16 Strobe signal support

The CC provides a number of strobe signals for observing internal protocol timing related signals in the protocol engine. The signals are listed and described in *Table 805*.

46.6.16.1 Strobe signal assignment

Each of the strobe signals listed in *Table 805* can be assigned to one of the four strobe ports using the Strobe Signal Control Register (FR_STBSCR). To assign multiple strobe signals, the application must write multiple times to the Strobe Signal Control Register (FR_STBSCR) with appropriate settings.

To read out the current settings for a strobe signal with number *n*, the application must execute the following sequence.

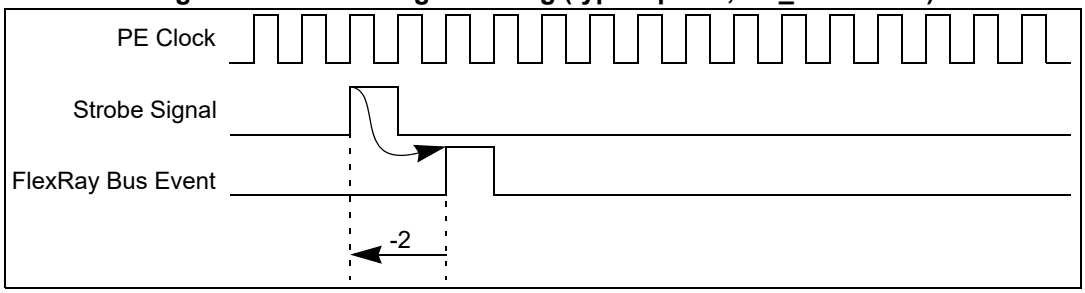
1. Write to FR_STBSCR with WMD = 1 and SEL = *n*. (Updates the SEL field only.)
2. Read STBCSR.
   The SEL field provides *n* and the ENB and STBPSEL fields provides the settings for signal *n*.

46.6.16.2 Strobe signal timing

This section provides detailed timing information of the strobe signals with respect to the protocol engine clock.

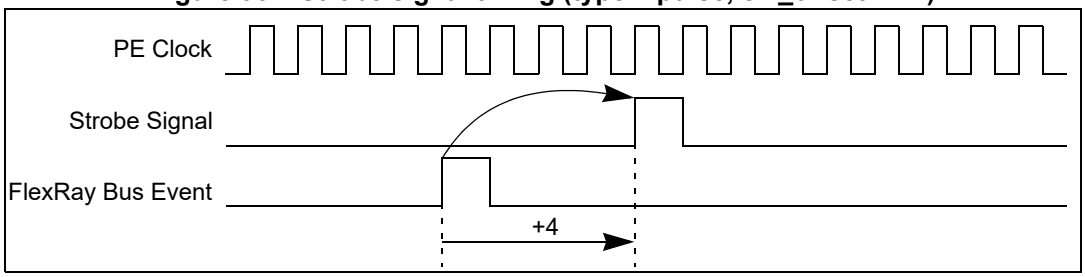
The strobe signals display internal PE signals. Due to the internal architecture of the PE, some signals are generated several PE clock cycles before the actual action is performed on the FlexRay bus. These signals are listed in *Table 805* with a negative clock offset. An example waveform is given in *Figure 961*.

Figure 961. Strobe signal timing (type = pulse, clk_offset = –2)



Other signals refer to events that occurred on the FlexRay bus some cycles before the strobe signal is changed. These signals are listed in *Table 805* with a positive clock offset. An example waveform is given in *Figure 962*.

Figure 962. Strobe signal timing (type = pulse, clk_offset = +4)



46.6.17 Timer support

The CC provides two timers that run on the FlexRay time base. Each timer generates a maskable interrupt when it reaches a configured point in time. Timer T1 is an absolute timer. Timer T2 can be configured to be an absolute or a relative timer. Both timers can be configured to be repetitive. In Non-repetitive mode, the timer stops if it expires. In Repetitive mode, the timer is restarted when it expires.

Both timers are active only when the protocol is in *POC:normal active* or *POC:normal passive* state. If the protocol is not in one of these modes, the timers are stopped. The application must restart the timers when the protocol has reached the *POC:normal active* or *POC:normal passive* state.

46.6.17.1 Absolute timer T1

Absolute timer T1 has the protocol cycle count and the macrotick count as the time base. The timer 1 interrupt flag TI1_IF in the Protocol Interrupt Flag Register 0 (FR_PIFR0) is set at the macrotick start event, if *Equation 66* and *Equation 67* are fulfilled.

Equation 66

$$CYCTR[CTCCNT] \& FR_TI1CYSR[T1_CYC_MSK] = FR_TI1CYSR[T1_CYC_VAL] \& FR_TI1CYSR[T1_CYC_MSK]$$

Equation 67

$$FR_MTCTR[MTCT] = FR_TI1MTOR[T1_MTOFFSET]$$

If the timer 1 interrupt enable bit TI1_IE in the Protocol Interrupt Enable Register 0 (FR_PIER0) is asserted, an interrupt request is generated.

The status bit T1ST is set when the timer is triggered, and is cleared when the timer expires and is non-repetitive. If the timer expires but is repetitive, the T1ST bit is not cleared and the timer is restarted immediately. The T1ST is cleared when the timer is stopped.

The corresponding interrupt condition (that is, when *Equation 66* and *Equation 67* are fulfilled) also leads to an event-out indication on an output port (*fr_evt_tim1*) based on the FR_PEOER[TIM1_EE].

46.6.17.2 Absolute/relative timer T2

Timer T2 can be configured to be an absolute or relative timer by setting the T2_CFG control bit in the Timer Configuration and Control Register (FR_TICCR). The status bit T2ST is set when the timer is triggered, and is cleared when the timer expires and is non-repetitive. If the timer expires but is repetitive, the T2ST bit is not cleared and the timer is restarted immediately. The T2ST is cleared when the timer is stopped.

46.6.17.2.1 Absolute timer T2

If timer T2 is configured as an absolute timer, it has the same functionality as timer T1 but the configuration from Timer 2 Configuration Register 0 (FR_TI2CR0) and Timer 2 Configuration Register 1 (FR_TI2CR1) is used. When timer T2 expires, the interrupt flag TI2_IF in the Protocol Interrupt Flag Register 0 (FR_PIFR0) is set. If the timer 1 interrupt enable bit TI1_IE in the Protocol Interrupt Enable Register 0 (FR_PIER0) is asserted, an interrupt request is generated.

The corresponding interrupt condition for T2 also leads to an event out indication on an output port (*fr_evt_tim2*) based on the FR_PEOER[TIM2_EE].

46.6.17.2.2 Relative Timer T2

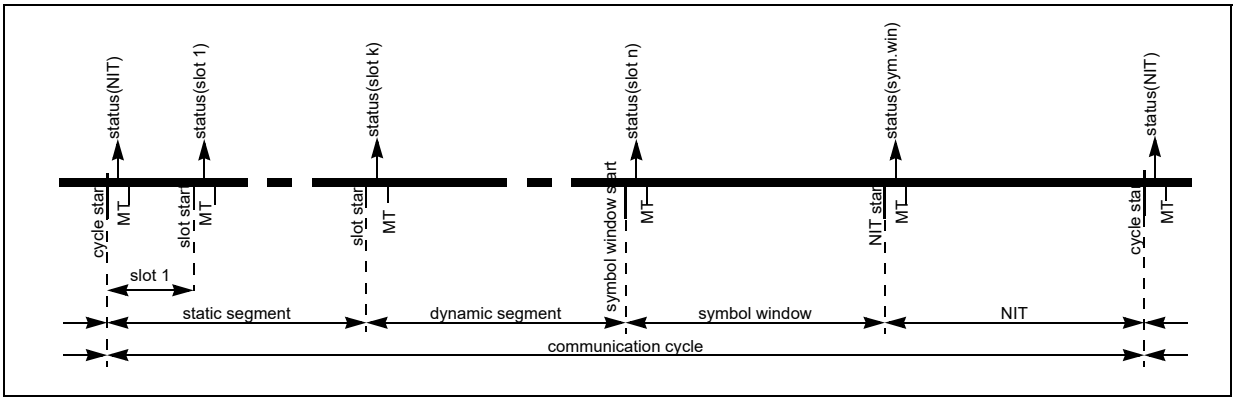
If timer T2 is configured as a relative timer, the interrupt flag TI2_IF in the Protocol Interrupt Flag Register 0 (FR_PIFR0) is set when the programmed amount of macroticks MT[31:0] – defined by Timer 2 Configuration Register 0 (FR_TI2CR0) and Timer 2 Configuration Register 1 (FR_TI2CR1) – has elapsed since the trigger or restart of timer T2. The relative timer is implemented as a down counter and expires when it has reached 0. At the macrotick start event, the value of MT[31:0] is checked and then decremented. Thus, if the timer is started with MT[31:0] == 0, it expires at the next macrotick start.The corresponding interrupt condition for T2 also leads to an event out indication on an output port (*fr_evt_tim2*) based on the FR_PEOER[TIM2_EE].

46.6.18 Slot status monitoring

The CC provides several means for slot status monitoring. All slot status monitors use the same slot status vector provided by the PE. The PE provides a slot status vector for each static slot, for each dynamic slot, for the symbol window, and for the NIT, on a per-channel base. The content of the slot status vector is described in *Table 920*. The PE provides the

slot status vector within the first macrotick after the end of the related slot/window/NIT, as shown in *Figure 963*.

Figure 963. Slot status vector update



Note: The slot status for the NIT of cycle n is provided after the start of cycle n+1.

Table 920. Slot status content

Slot	Status content

static/dynamic Slot	Slot-related status
	vSS!ValidFrame: valid frame received
	vSS!SyntaxError: syntax error occurred while receiving
	vSS!ContentError: content error occurred while receiving
	vSS!BViolation: boundary violation while receiving
	for slots in which the module transmits:
	vSS!TxConflict: reception ongoing while transmission starts
	for slots in which the module does not transmit:
	vSS!TxConflict: reception ongoing while transmission starts
	first valid - channel that has received the first valid frame
	Received-frame-related status
	extracted from
	a) header of valid frame, if vSS!ValidFrame = 1
	b) last received header, if vSS!ValidFrame = 0
	c) set to 0, if nothing was received
	vRF!Header!NFIndicator: null frame indicator (0 for null frame)
	vRF!Header!SuFIndicator: startup frame indicator
	vRF!Header!SyFIndicator: sync frame indicator

Symbol window	Window-related status *vSS!ValidFrame*: always 0 *vSS!ContentError*: content error occurred while receiving *vSS!SyntaxError*: syntax error occurred while receiving *vSS!BViolation*: boundary violation while receiving *vSS!TxConflict*: reception ongoing while transmission starts Received-symbol-related status *vSS!ValidMTS*: valid Media Test Access Symbol received Received-frame-related status see static/dynamic slot

Table 920. Slot status content (continued)

Slot	Status content
NIT	NIT-related status *vSS!ValidFrame*: always 0 *vSS!ContentError*: content error occurred while receiving *vSS!SyntaxError*: syntax error occurred while receiving *vSS!BViolation*: boundary violation while receiving *vSS!TxConflict*: always 0 Received-frame-related status see static/dynamic slot

46.6.18.1 Channel Status Error Counter Registers

The two channel status error counter registers, Channel A Status Error Counter Register (FR_CASERCR) and Channel B Status Error Counter Register (FR_CBSERCR) are incremented by one if at least one of four slot status error bits, *vSS!SyntaxError*, *vSS!ContentError*, *vSS!BViolation*, or *vSS!TxConflict* is set to 1. The status vectors for all slots in the static and dynamic segment, in the symbol window, and in the NIT are taken into account. The counters wrap around after they have reached the maximum value.

46.6.18.2 Protocol Status Registers

The Protocol Status Register 2 (FR_PSR2) provides slot status information about the NIT and the symbol window. The Protocol Status Register 3 (FR_PSR3) provides aggregated slot status information.

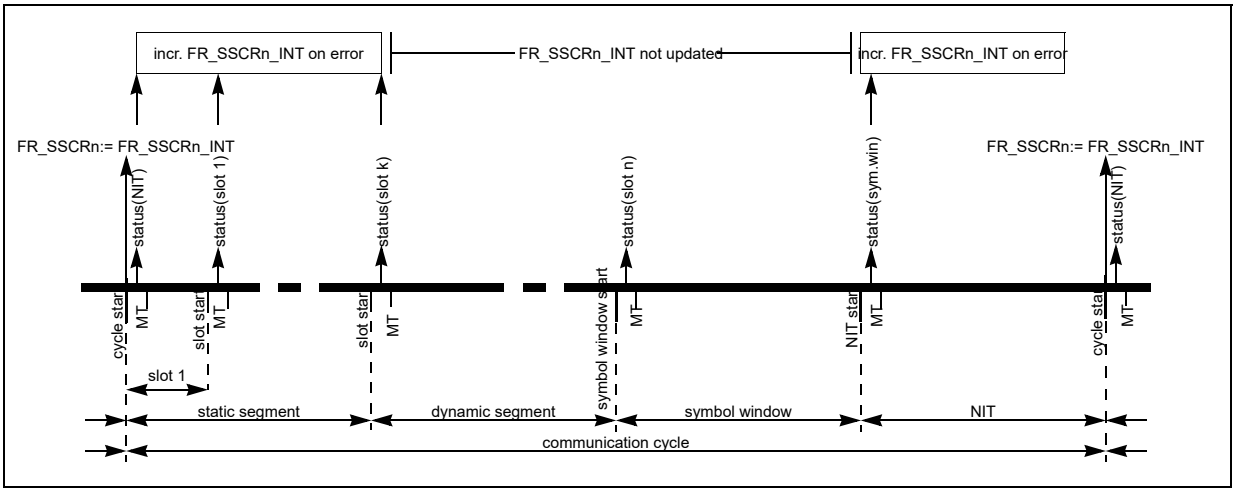
46.6.18.3 Slot Status Registers

The eight slot status registers, Slot Status Registers (FR_SSR0–FR_SSR7), can be used to observe the status of static slots, dynamic slots, the symbol window, or the NIT without individual message buffers. These registers provide all slot status-related and received frame/symbol-related status information, as given in *Table 920*, except for the *first valid* indicator for non-transmission slots.

46.6.18.4 Slot Status Counter Registers

The CC provides four slot status error counter registers, Slot Status Counter Registers (FR_SSCR0–FR_SSCR3). Each of these slot status counter registers is updated with the value of an internal slot status counter at the start of a communication cycle. The internal slot status counter is incremented if its increment condition, defined by the Slot Status Counter Condition Register (FR_SSCCR), matches the status vector provided by the PE. All static slots, the symbol window, and the NIT status are taken into account. *Dynamic* slots are *excluded*. The internal slot status counting and update timing is shown in *Figure 964*.

Figure 964. Slot status counting and FR_SSCR*n* update



The PE provides the status of the NIT in the first slot of the next cycle. Due to these facts, the FR_SSCR*n* register reflects, in cycle n, the status of the NIT of cycle n – 2, and the status of all static slots and the symbol window of cycle n – 1.

The increment condition for each slot status counter consists of two parts: the frame-related condition part and the slot-related condition part. The internal slot status counter FR_SSCR*n*_INT is incremented if at least one of the following conditions is fulfilled:

1. Frame-related condition:

- (FR_SSCCR*n*[VFR] | FR_SSCCR*n*[SYF] | FR_SSCCR*n*[NUF] | FR_SSCCR*n*[SUF]) // count on frame condition
  = 1;

and

- ((~FR_SSCCR*n*[VFR] | vSS!ValidFrame) & // valid frame restriction
  (~FR_SSCCR*n*[SYF] | vRF!Header!SyFIndicator) & // sync frame indicator restriction
  (~FR_SSCCR*n*[NUF] | ~vRF!Header!NFIndicator) & // null frame indicator restriction
  (~FR_SSCCR*n*[SUF] | vRF!Header!SuFIndicator)) // startup frame indicator restriction
  = 1;

Note: The indicator bits SYF, NUF, and SUF are valid only when a valid frame was received. Thus, the VFR must always be set whenever the count-on-frame condition is used.

2. Slot-related condition:

- ((FR_SSCCR*n*[STATUSMASK[3]] & vSS!ContentError) | // increment on content error
  (FR_SSCCR*n*[STATUSMASK[2]] & vSS!SyntaxError) | // increment on syntax error
  (FR_SSCCR*n*[STATUSMASK[1]] & vSS!BViolation) | // increment on boundary violation
  (FR_SSCCR*n*[STATUSMASK[0]] & vSS!TxConflict)) // increment on transmission conflict
  = 1;

If the slot status counter is in Single-cycle mode (that is, FR_SSCCR*n*[MCY] = 0), the internal slot status counter FR_SSCR*n*_INT is reset at each cycle start. If the slot status counter is in Multicycle mode (that is, FR_SSCCR*n*[MCY] = 1), the counter is not reset and incremented until the maximum value is reached.

46.6.18.5 Message Buffer Slot Status field

Each individual message buffer and each FIFO message buffer provides a slot status field, which provides the information shown in *Table 920* for the static/dynamic slot. The update conditions for the slot status field depend on the message buffer type. Refer to the Message Buffer Update sections in *Section 46.6.3.4.1*.

46.6.19 System bus access

This section provides a description of the system bus access failures and the related CC behavior. System bus access failures may occur when the CC transfers data to or from the FlexRay memory area.

The system bus access failure types are described in *Section 46.6.19.1*.

The behavior of the CC after the occurrence of a system bus access failure is described in *Section 46.6.19.2*.

46.6.19.1 System bus access failure types

This section describes the two types of system bus access failures.

46.6.19.1.1 System bus illegal address access

A system bus illegal address access is detected when the CC has used an illegal or invalid address to access the FlexRay system memory area. There are three conditions that are treated as a system bus illegal address access:

- The system bus subsystem detects a CC access to an illegal system memory address.
- The CC detects the usage of an data field offset with the value of 0.
- The CC detects a memory error while reading a data field offset from the CHI LRAM memory (see *Section 46.6.24.3.1*).

If a system bus illegal address access is detected, the CC sets the ILSA_EF flag in the CHI Error Flag Register (FR_CHIERFR).

46.6.19.1.2 System bus access timeout

A system bus access timout is detected if an access to the FlexRay memory area is not finished in time. The timeout value is derived from the SYMATOR[TIMEOUT] setting (see *Section 46.7.1.1*).

If a system bus access timout is detected, the CC sets the SBCF_EF flag in the CHI Error Flag Register (FR_CHIERFR).

46.6.19.2 System bus access failure response

This section describes the two types of behavior of the CC after the occurrence of a system bus access failure. The actual behavior is defined by the SBFF bit in the Module Configuration Register (FR_MCR).

46.6.19.2.1 Continue after system bus access failure

If FR_MCR[SBFF] is 0, the CC will continue its operation after the occurrence of the system bus access failure, but it will not generate any system bus accesses until the start of the next communication cycle.Since no data is read from or written to the FlexRay memory

area, no messages are received or transmitted. Consequently, none of the individual message buffers or receive FIFOs will be updated until the next communication cycle starts.

If a frame is under transmission when the system bus failure occurs, a correct frame is generated in which the remaining header and frame data are replaced by all zeros. Depending on the point in time, this can affect the PPI bit, the header CRC, the payload length in the case of a dynamic slot, and the payload data. Starting from the next slot in the current cycle, no frames will be transmitted or received, except for the key slot; a sync or startup null-frame is transmitted if the key slot is assigned.

If a frame is received when the system bus failure occurs, the reception is aborted and the related receive message buffer is not updated.

Normal operation is resumed after the start of next communication cycle.

46.6.19.2.2 Freeze after system bus access failure

If FR_MCR[SBFF] is set to 1, the CC will go into Freeze mode immediately after the occurrence of one of the system bus access failures.

46.6.20 Interrupt support

The CC provides 108 individual interrupt sources and five combined interrupt sources.

46.6.20.1 Individual interrupt sources

46.6.20.1.1 Message buffer interrupts

The CC provides 64 message buffer interrupt sources.

Each individual message buffer provides an interrupt flag FR_MBCCSR*n*[MBIF] and an interrupt enable bit FR_MBCCSR*n*[MBIE]. The CC sets the interrupt flag when the slot status of the message buffer was updated. If the interrupt enable bit is asserted, an interrupt request is generated.

46.6.20.1.2 FIFO interrupts

The CC provides two FIFO interrupt sources.

Each of the two FIFOs provides a Receive FIFO Almost Full Interrupt Flag. The CC sets the Receive FIFO Almost Full Interrupt Flags (FR_GIFER[FAFBIF], FR_GIFER[FAFAIF]) in the Global Interrupt Flag and Enable Register (FR_GIFER) if the corresponding receive FIFO fill level exceeds the defined watermark.

46.6.20.1.2.1 Wakeup interrupt

The CC provides one interrupt source related to the wakeup.

The CC sets the Wakeup Interrupt Flag FR_GIFER[WUPIF] when it has received a wakeup symbol on the FlexRay bus. The CC generates an interrupt request if the interrupt enable bit FR_GIFER[WUPIE] is asserted.

46.6.20.1.2.2 Protocol interrupts

The CC provides 25 interrupt sources for protocol-related events. For details, see the Protocol Interrupt Flag Register 0 (FR_PIFR0) and Protocol Interrupt Flag Register 1 (FR_PIFR1). Each interrupt source has its own interrupt enable bit.

46.6.20.1.2.3 CHI interrupts

The CC provides 16 interrupt sources for CHI-related error events. For details, see the CHI Error Flag Register (FR_CHIERFR). There is one common interrupt enable bit FR_GIFER[CHIE] for all CHI error interrupt sources.

46.6.20.2 Combined interrupt sources

Each combined interrupt source generates an interrupt request only when at least one of the interrupt sources that is combined generates an interrupt request.

46.6.20.2.1 Receive message buffer interrupt

The receive message buffer interrupt request is generated when at least one of the individual receive message buffers generates an interrupt request MBXIRQ[n] and the interrupt enable bit FR_GIFER[RBIE] is set.

46.6.20.2.2 Transmit message buffer interrupt

The transmit message buffer interrupt request is generated when at least one of the individual transmit message buffers generates an interrupt request MBXIRQ[n] and the interrupt enable bit FR_GIFER[TBIE] is asserted.

46.6.20.2.3 Protocol interrupt

The protocol interrupt request is generated when at least one of the individual protocol interrupt sources generates an interrupt request and the interrupt enable bit FR_GIFER[PRIE] is set..

46.6.20.2.4 CHI interrupt

The CHI interrupt request is generated when at least one of the individual chi error interrupt sources generates an interrupt request and the interrupt enable bit FR_GIFER[CHIE] is set.

46.6.20.2.5 Module interrupt

The module interrupt request is generated if at least one of the combined interrupt sources generates an interrupt request and the interrupt enable bit FR_GIFER[MIE] is set.

Figure 965. Scheme of FR_GIFER interrupt signal generation

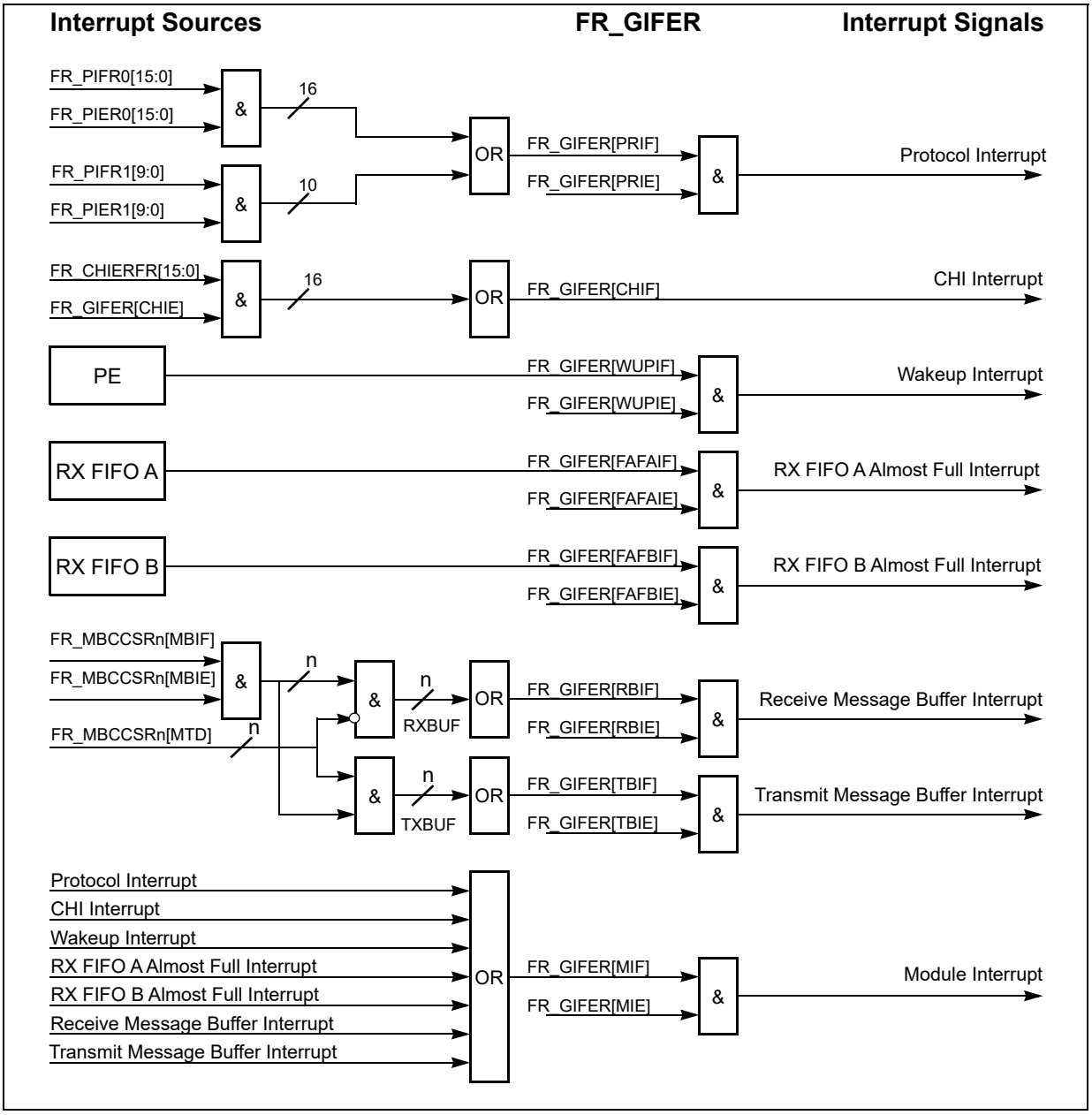


Figure 966. FR_EEIFER interrupt signal generation scheme

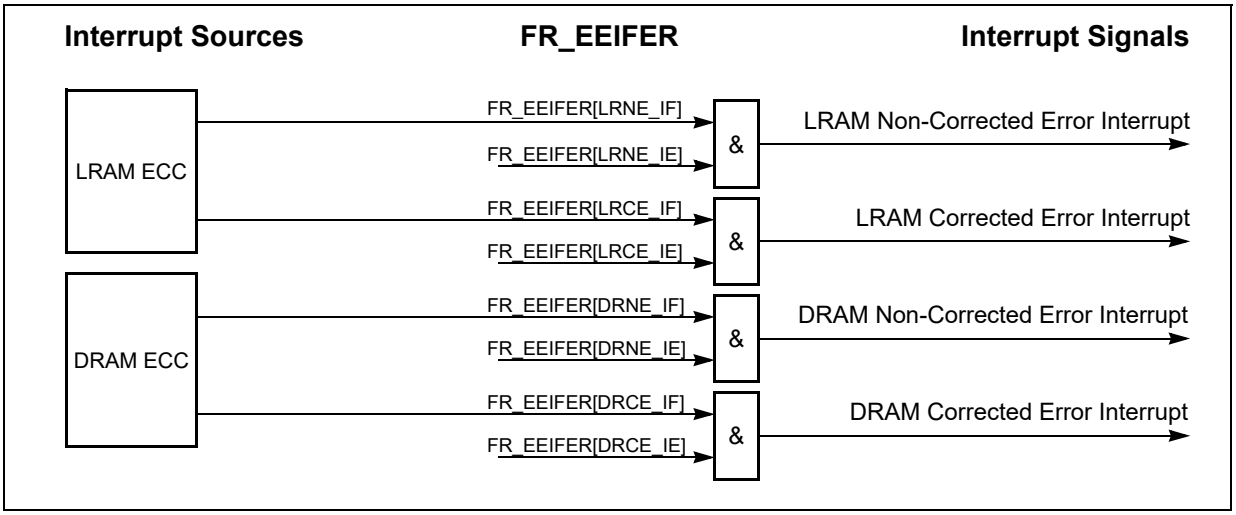
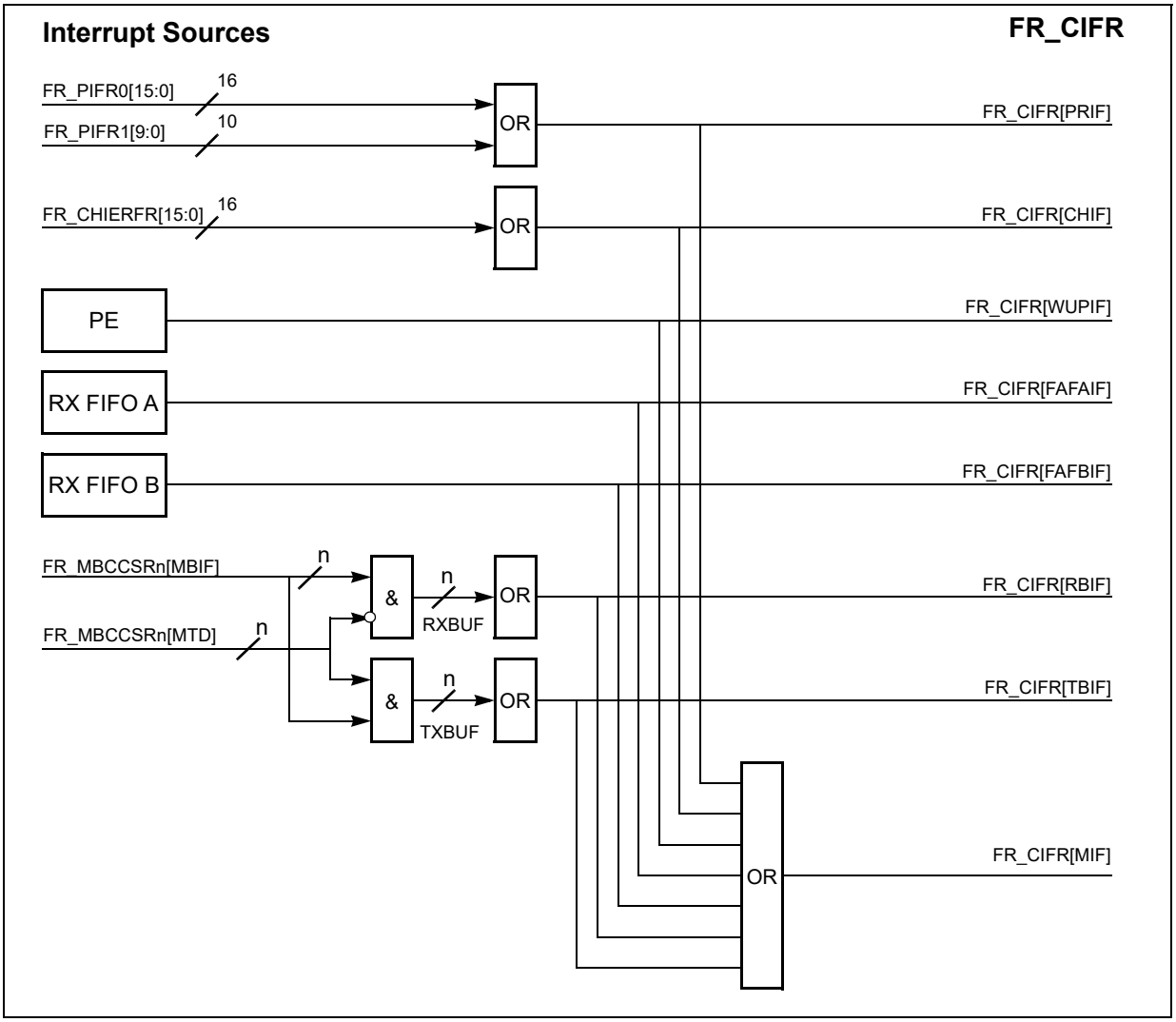


Figure 967. FR_CIFR flag generation scheme



46.6.21 Lower bit rate support

The CC supports a number of lower bit rates on the FlexRay bus channels. The lower bit rates are implemented by modifying the duration of the microtick *pdMicrotick*, the number of samples per microtick *pSamplesPerMicrotick*, the number of samples per bit *cSamplesPerBit*, and the strobe offset *cStrobeOffset.* The application configures the FlexRay channel bit rate by setting the BITRATE field in the Module Configuration Register (FR_MCR). The protocol values are set internally. The available bit rates, the related BITRATE field configuration settings, and the related protocol parameter values are shown in *Table 921*.

Table 921. FlexRay channel bit rate control

FlexRay channel bit rate [Mbit/s]	FR_MCR[BITRATE]	*pdMicrotick* [ns]	*gdSampleClockPeriod* [ns]	pSamplesPerMicrotick	cSamplesPerBit	cStrobeOffset
10.0	000	25.0	12.5	2	8	5
8.0	011	25.0	12.5	2	10	6
5.0	001	25.0	25.0	1	8	5
2.5	010	50.0	50.0	1	8	5

Note: The bit rate of 8 Mbit/s is not defined by the FlexRay Communications System Protocol Specification, Version 2.1 Rev A.

46.6.22 PE data memory (PE DRAM)

The PE data memory (PE DRAM) is 128 word, 16-bit wide memory with byte access that contains the program data of the PE internal CPU. The PE DRAM is divided into two banks of 8 bits each. The memory data [7:0] is assigned to BANK0 and the memory data [15:8] is assigned to BANK1.

Table 922. PE DRAM layout

ADDR	BANK1	BANK0
0x00	byte1	byte0
0x01	byte3	byte2
…		
0x7F	byte255	byte254

The FlexRay module provides means to access the PE DRAM from the application. The PE DRAM application access is initiated and controlled via *Section 46.5.2.9: PE DRAM Access Register (FR_PEDRAR)* and *Section 46.5.2.9: PE DRAM Access Register (FR_PEDRAR)*. This functionality is used to check the memory error detection.

46.6.22.1 PE DRAM read access

A read access from the PE DRAM can be initiated in any protocol state. The following sequence describes a read access from the PE DRAM address 0x70.

1. FR_PEDRAR:= 0x50E0;
   // INST=0x5; ADDR=070
2. Wait until FR_PEDRAR[DAD] == 1;
   // wait for end of PE DRAM access
3. val = FR_PEDRDR[DATA];
   // read PE DRAM data

The read access is handled by the PE internal CPU with the lowest execution priority. This may cause a response delay with a maximum of 1000 PE clock cycle (25 µs).

46.6.22.2 PE DRAM write access

A write access into the PE DRAM can be initiated in any protocol state. The following sequence describes a write access to the PE DRAM address 0x70.

1. FR_PEDRDR:= DATA;
   // write value to be written into data register
2. FR_PEDRAR:= 0x30E0;
   // INST=0x3; ADDR=0x70
3. wait until FR_PEDRAR[DAD] == 1;
   // wait for end of PE DRAM access
4. val = FR_PEDRDR[DATA];
   // read back PE DRAM data

The write access is handled by the PE internal CPU with the lowest execution priority. This may cause an response delay with a maximum of 1000 PE clock cycle (25 µs).

If the conditions given in *Section 46.6.22.3: PE DRAM write access limitations* are fulfilled, the data provided in PE DRAM Access Register (FR_PEDRAR) is written into the PE DRAM, read back in the next clock cycle, and stored into the PE DRAM Access Register (FR_PEDRAR). Otherwise, data is not written into the PE DRAM and 0x0000 is stored into the PE DRAM Access Register (FR_PEDRAR).

46.6.22.3 PE DRAM write access limitations

The PE DRAM is used by the protocol engine if the module is not in *POC:default config* state. The only address not used by the protocol engine is 0x70. To prevent the corruption of protocol engine data the following PE DRAM write access limitations apply for application writes.

1. When the module is in *POC:default config* state, all PE DRAM addresses are writable.
2. When the module is not in *POC:default config* state, only PE DRAM address 0x70 is writable.

46.6.23 CHI lookup-table memory (CHI LRAM)

The CHI lookup-table memory (CHI LRAM) is an CHI-internal memory that contains the message buffer configuration data and the data field offsets for the physical message buffers. The configuration data for two message buffers or six data field offsets are contained in one memory row. The CHI LRAM is divided into six memory banks.

Table 923. CHI LRAM layout

ADR	BANK5	BANK4	BANK3	BANK2	BANK1	BANK0
0x00	FR_MBIDXR1	FR_MBFIDR1	FR_MBCCFR1	FR_MBIDXR0	FR_MBFIDR0	FR_MBCCFR0
0x01	FR_MBIDXR3	FR_MBFIDR3	FR_MBCCFR3	FR_MBIDXR2	FR_MBFIDR2	FR_MBCCFR2
…						
0x1F	FR_MBIDXR63	FR_MBFIDR63	MBCCF63	FR_MBIDXR62	MBFID62	FR_MBCCFR62
0x20	FR_MBDOR5	FR_MBDOR4	FR_MBDOR3	FR_MBDOR2	FR_MBDOR1	FR_MBDOR0
…						
0x2B	-	-	-	-	FR_MBDOR67	FR_MBDOR66
0x2C	FR_LEETR5	FR_LEETR4	FR_LEETR3	FR_LEETR2	FR_LEETR1	FR_LEETR0

46.6.23.1 CHI LRAM read and write access

The CHI LRAM is accessed by the application via regular register read and write accesses.

46.6.24 Memory content error detection

The FlexRay module provides integrated memory content error detection for both the CHI LRAM and PE DRAM, and memory content error correction for the PE DRAM. The memory error detection for the CHI LRAM uses a standard Hamming code with a Hamming distance of 3 and detects all single-bit and double-bit errors (SEDDED). The memory error detection and correction for the PE DRAM uses an enhanced Hamming code with a Hamming distance of 4 and detects and corrects all single-bit errors and detects all double-bit errors (SECDED).

This section describes the reporting of the occurrence of memory content errors, the reaction of the module on the occurrence, and how the application can inject memory errors in order to trigger the report and response behavior.

46.6.24.1 Memory error types

A memory error is the distortion of one or more bits read out of the memory. The reading of the values of all zeros and all ones is considered as a special case. The FlexRay module detects and indicates the memory errors as shown in *Table 924*. The entries on the top have higher priority.

Each memory read access reads out *all* banks of the addressed row and runs error detection on *all* banks, even in the case that the application has triggered a read from only one bank. This may lead to the reporting of an memory error if at least one bank contains a memory error, even if an error-free bank has been read.

Table 924. Detected memory error types

Memory	Priority	Memory data	Indication
CHI LRAM	0 (highest)	All zeros	Non-corrected error
PE DRAM			Non-corrected error
CHI LRAM		All ones	Non-corrected error
PE DRAM			
CHI LRAM	1 (lowest)	One bit flipped	Non-corrected error
PE DRAM			Corrected error
CHI LRAM		Two bits flipped	Non-corrected error
PE DRAM			
CHI LRAM		Three or more bits flipped	One out of {no error, non-corrected error}, defined by coding given in *Section 46.6.24.2.3* and *Section 46.6.24.2.4*
PE DRAM			One out of {no error, corrected error, non-corrected error}, defined by coding given in *Section 46.6.24.2.1* and *Section 46.6.24.2.2*

46.6.24.2 Memory error reporting

The memory error reporting is enabled only if the ECC functionality enable bit ECCE in the Module Configuration Register (FR_MCR) is set.

For each of the two memories, there are two sets of internal registers to store the detection of one corrected and one non-corrected memory error.

If a memory error is detected, the module checks whether the related error interrupt flag in the ECC Error Interrupt Flag and Enable Register (FR_EEIFER) is set.

- If the error interrupt flag is set, the related internal error reporting register is not updated and the related error overflow flag is set to 1 to indicate a loss of error condition.
- If the error interrupt flag is not set, the internal reporting register is updated and the error interrupt flag is set to 1. If two or more memory errors of the same type are detected, the error for the bank with the lower bank number will be reported, and the error overflow flag will be set to 1.

If a memory error is detected for at least two banks of one memory, the related error overflow flag is set to 1 to indicate a loss of error condition.

In addition to the above, the error indications (corrected/non-corrected) along with the failing address are sent out from the module for external error logging and corrective actions by the software.

46.6.24.2.1 PE DRAM checkbits

The coding of the checkbits reported in ECC Error Report Code Register (FR_EERCR) for PE DRAM memory errors is shown in *Table 924*. This table shows the implemented enhanced Hamming code. If the error injection was applied to distort the checkbits, then the distorted checkbits are reported.

Table 925. PE DRAM checkbits coding

CODE	CODE				DATA							
	3	2	1	0	7	6	5	4	3	2	1	0
4(1)	X	X	X	X	X	X	X	X	X	X	X	X
3(2)	—	—	—	—	X	X	X	X	—	—	—	—
2	—	—	—	—	X	—	—	—	X	X	X	—
1	—	—	—	—	—	X	X	—	X	X	—	X
0	—	—	—	—	—	X	—	X	X	—	X	X

1. The checkbit CODE[4] is set to 1 if and only if there is an even number of 1's in columns with X.
2. The checkbits CODE[3]… CODE[0] are set to 1 if and only if there is an odd number of 1's in all columns with X.

This coding of the checkbit ensures that neither 0x000 nor 0xFFF are valid code words written into the memory.

46.6.24.2.2 PE DRAM syndrome

The coding of the syndrome reported in the ECC Error Report Code Register (FR_EERCR) for PE DRAM memory errors is shown in *Table 926*.

Table 926. FR_EERCR[CODE] PE DRAM syndrome coding

FR_EERCR[CODE]		Description
[4]	[3:0]	
0x1	0x0	No Error (never appears in error report registers)
0x0	0x0	If data == 0: Non Corrected Error (Dedicated Handling of All Zero Code Word) If data!= 0: Corrected Error (Parity Bit 4)
0x0	0x1	Corrected error (parity bit 0)
0x0	0x2	Corrected error (parity bit 1)
0x0	0x3	Corrected error (data bit 0)
0x0	0x4	Corrected error (parity bit 2)
0x0	0x5	Corrected error (data bit 1)
0x0	0x6	Corrected error (data bit 2)
0x0	0x7	Corrected error (data bit 3)
0x0	0x8	Corrected error (parity bit 3)
0x0	0x9	Corrected error (data bit 4)
0x0	0xA	Corrected error (data bit 5)
0x0	0xB	Corrected error (data bit 6)
0x0	0xC	Corrected error (data bit 7)
0x0	0xD–0xF	Non-corrected error
0x1	0x1–0xF	Non-corrected error

46.6.24.2.3 CHI LRAM checkbits

The coding of the checkbits reported in ECC Error Report Code Register (FR_EERCR) for CHI LRAM memory errors is shown in *Table 927*. This table shows the implemented Hamming code. If the error injection was applied to distort the checkbits, then the distorted checkbits are reported.

Table 927. CHI LRAM checkbits coding

CODE(1)	DATA															
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
4	X	X	X	X	X	—	—	—	—	—	—	—	—	—	—	—
3	—	—	—	—	—	X	X	X	X	X	X	X	—	—	—	—
2	X	X	—	—	—	X	X	X	X	—	—	—	X	X	X	—
1	—	—	X	X	—	X	X	—	—	X	X	—	X	X	—	X
0	X	—	X	—	X	X	—	X	—	X	—	X	X	—	X	X

1. The checkbit CODE[n] is set to 1 if and only if there is an odd number of 1's in all columns with X.

46.6.24.2.4 CHI LRAM syndrome

The coding of the syndrome reported in ECC Error Report Code Register (FR_EERCR) for CHI LRAM memory errors is shown in *Table 928*.

Table 928. FR_EERCR[CODE] CHI LRAM syndrome coding

FR_EERCR[CODE]	Description
0x00	No Error (never appears in error report registers)
0x01–0x1F	Non-corrected error

46.6.24.3 Memory error response

The memory error response is enabled only when the ECC functionality enable bit ECCE in the Module Configuration Register (FR_MCR) is set.

In case a *corrected* memory error is detected, the FlexRay module continues its normal operation using the corrected data word. This section describes the behavior of the FlexRay module after the detection of a *non-corrected* memory error.

46.6.24.3.1 CHI LRAM error response after CC read

When the CC is out of the *POC:default config* state, it reads the configuration data and the data field offsets of all utilized message buffers in every slot and in the NIT. If a non-corrected memory error is detected during this module read access, the error response of the module depends from LRAM location where the error occurred.

- If the LRAM address belongs to physical message buffer configuration data the FlexRay module will consider the affected message buffer as disabled for the current

search and will exclude this buffer from the search. The configuration of the affected message buffer is not changed.

  – If the affected message buffer is a Tx message buffer, no frame will be transmitted from this message buffer in the next slot. If the affected message buffer is an Rx message buffer, no frame will be received to this message buffer in the next slot.

- If the LRAM address belongs to the data field offset area and the related physical message buffer is used for Rx or Tx, the first access to the system memory caused by payload read or write yields to the assertion of the FR_CHIERFR[ILSA_EF]. No memory access occurs with respect to payload access for the complete frame.

46.6.24.3.2 CHI LRAM error response after application read

The application can read the content of the CHI LRAM by reading the FR_MBCCFR*n*, FR_MBFIDR*n*, FR_MBIDXR*n*, FR_MBDOR*n* and FR_LEETR*n* registers. If a non-corrected memory error is detected during this kind of read access, the module indicates the detected memory error, delivers the non-corrected data read, and continues its normal operation.

46.6.24.3.3 PE DRAM error response after CC read

If the CC detects a non-corrected memory error during an internal read of program data contained in PE DRAM, this is considered a fatal protocol error and the module enters the protocol freeze state immediately.

46.6.24.3.4 PE DRAM error response after application read in *POC:default config* state

If the CC detects a non-corrected memory error during an application-triggered read from any PE DRAM address and the protocol is in the *POC:default config* state, this is considered a fatal protocol error and the module enters the protocol freeze state. This behavior allows for checking the freeze functionality in case of the detection of non-corrected errors.

46.6.24.3.5 PE DRAM error response after application read out of *POC:default config*

If the CC detects a non-corrected memory error during an application-triggered read from any PE DRAM address and the protocol is not in the *POC:default config* state, this error is not considered a fatal error and the protocol state is not changed. This prevents any interference of the running protocol by PE DRAM error injection reads.

46.6.25 Memory error injection

The error injection functionality is used by the application to inject data errors into the memories to trigger and check the memory error detection functionality.

The error injection is enabled only if the ECC functionality enable bit ECCE in the Module Configuration Register (FR_MCR) and the error injection enable control bit EIE in the ECC Error Report and Injection Control Register (FR_EERICR) are set.

Error Injection mode is configured by the EIM configuration bit in the ECC Error Report and Injection Control Register (FR_EERICR).When error injection is enabled, each write access to the configured memory location will be distorted.

The injector has the same behavior for FlexRay module memory writes and application memory writes.

46.6.25.1 CHI LRAM error injection

The following sequence describes an memory error injection sequence for the CHI LRAM memory. This sequence consists of the error injector setup followed by an application triggered write access to provoke an distortion of the memory content. The content of the CHI LRAM is described in *Table 923*.

When the CC is in *POC:default config*, there are no limitations for the error injection and no impacts of error injection to the application. For error injection out of *POC:default config* see *Section 46.7.3*.

Injector setup:

1. FR_MCR[ECCE]:= 1;
   // enable ECC functionality
2. FR_EERICR[EIE]:=I_MODE;
   // configure error injection mode
3. FR_EEIAR[MID]:= 1;
   // select CHI LRAM for error injection
4. FR_EEIAR[BANK]:= I_BANK;
   // select bank for error injection; I_BANK = {0,1,2,3,4,5}
5. FR_EEIAR[ADDR]:= I_ADDR;
   // select address for error injection; I_ADDR ≤ 0x2C
6. FR_EEIDR[DATA]:= D_DIST;
   // define data distortion pattern
7. FR_EEICR[CODE]:= C_DIST;
   // define checkbit distortion pattern
8. FR_EERICR[EIE]:=1;
   // enable error injection

Application Write Access:

If (I_BANK==0) -> FR_MBCCFR(2n) / FR_MBDOR(6k) / FR_LEETR0 := DATA;
If (I_BANK==1) -> FR_MBFIDR(2n) / FR_MBDOR(6k+1) / FR_LEETR1 := DATA;
If (I_BANK==2) -> FR_MBIDXR(2n) / FR_MBDOR(6k+2) / FR_LEETR2 := DATA;
If (I_BANK==3) -> FR_MBCCFR(2n+1) / FR_MBDOR(6k+3) / FR_LEETR3 := DATA;
If (I_BANK==4) -> FR_MBFIDR(2n+1) / FR_MBDOR(6k+4) / FR_LEETR4 := DATA;
If (I_BANK==5) -> FR_MBIDXR(2n+1) / FR_MBDOR(6k+5) / FR_LEETR5 := DATA;
// write DATA to the defined injection bank and injection address (see *Table 923*).

46.6.25.2 PE DRAM error injection

The following sequence describes a memory error injection sequence for the PE DRAM memory. This sequence consists of the error injector setup followed by an application triggered write access to provoke an distortion of the memory content.

When the FlexRay module is in *POC:default config*, there are no limitations for the error injection and no impacts of error injection to the application. For error injection out of *POC:default config* see *Section 46.7.3.2*.

Injector setup:

1. FR_MCR[ECCE]:= 1;
   // enable ECC functionality
2. FR_EERICR[EIE]:=I_MODE;
   // configure error injection mode
3. FR_EEIAR[MID]:= 0;
   // select PE DRAM for error injection
4. FR_EEIAR[BANK]:= I_BANK;
   // define bank for error injection; I_BANK = {0,1}
5. FR_EEIAR[ADDR]:= I_ADDR;
   // define address for error injection; I_ADDR ≤ 0x7F
6. FR_EEIDR[DATA]:= D_DIST;
   // define data distortion pattern
7. FR_EEICR[CODE]:= C_DIST;
   // define checkbit distortion pattern
8. FR_EERICR[EIE]:=1;
   // enable error injection

Application write access (for example, I_ADDR=0x70):

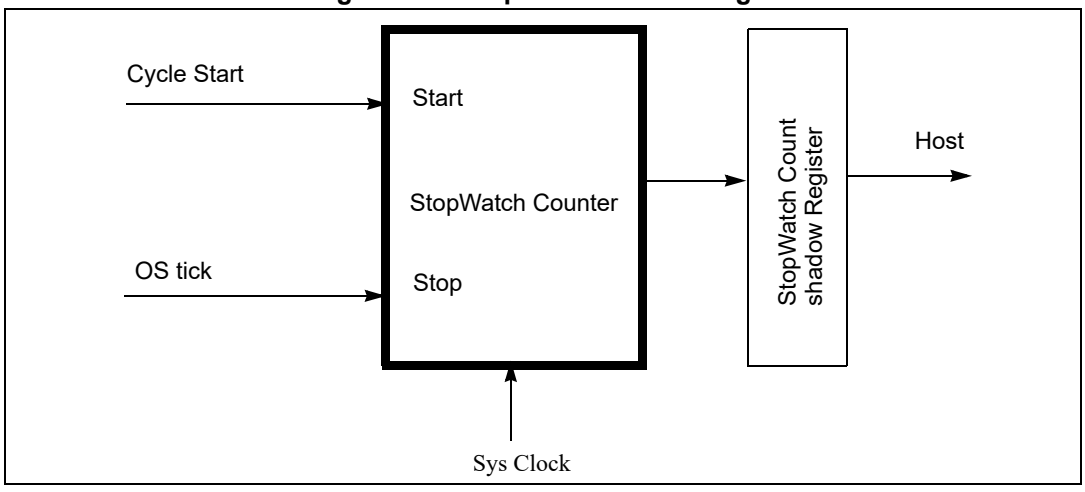
1. FR_PEDRAR:= 0x30E0;
   // INST=0x3; ADDR=0x70
2. wait until FR_PEDRAR[DAD] == 1;
   // wait for end of PE DRAM access
3. val = FR_PEDRDR[DATA]; |
   // get read back PE DRAM data

Note: The write access to the PE DRAM triggers an subsequent read access from PE DRAM in the next cycle, which triggers the detection of the distorted data.

46.6.26 StopWatch function

FlexRay supports a StopWatch function to facilitate OS synchronization to the FlexRay network. This provides a capability to capture the time difference in system clocks between the operating system timer tick and the FlexRay time base. This allows the software synchronization handler to adjust the OS clock to sync with the FR macrotick.

Figure 968. StopWatch counter logic

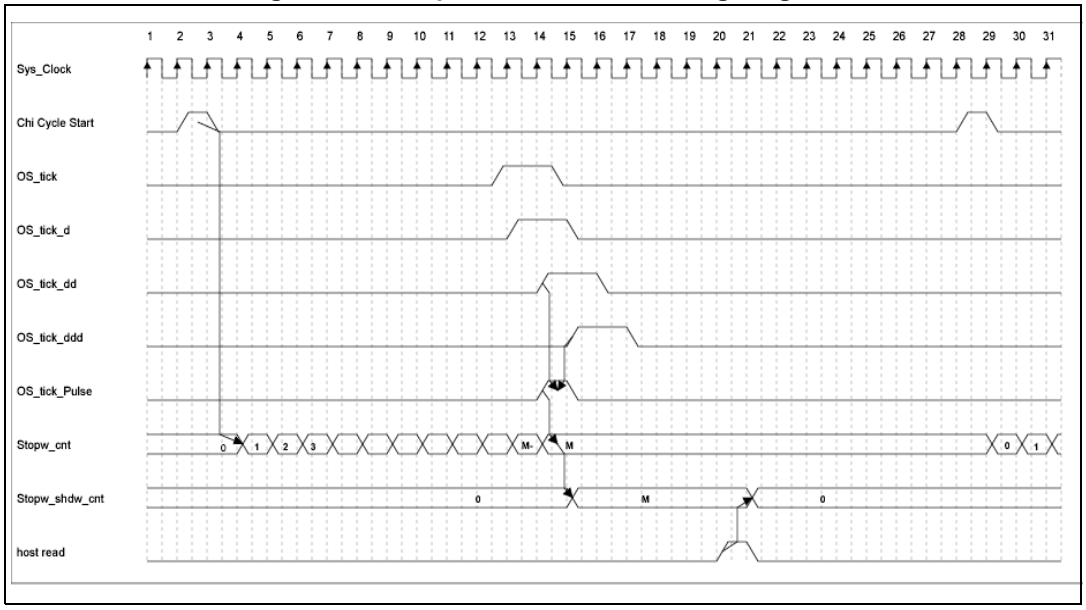


As shown here, there is a 32 bit stopwatch counter that resets and starts with each cycle start. This counter runs on the system clock and increments every cycle until an OS tick event is received, after which the counter halts and the count value is transferred to a shadow register. The shadow register (*Section 46.5.2.67: Stop Watch Count Register (FR_STPWR)*) holds the counter value till the host reads all the 32 bits, after which it automatically gets cleared.

The StopWatch logic is controlled through a control bit STPW_EN in the FR_PEOER (*Section* : . The counter runs only when this bit is set.

Figure 969 shows the actual timing of the stopwatch counter logic operation.

Figure 969. StopWatch function timing diagram



1. ‘Host read’ implies that all 32 bits have been read out completely. Any partial read (that is for 8/16 bits access) will not clear the StopWatch count register.
2. “If cycle start and OS tick occurs simultaneously, timer will remain at 0. But OS_tick has a 2 stage synchronizer and cycle_start has a single stage synchronizer into chi clock domain. So simultaneously here means OS_tick_pulse and chi_cycle_start occurring at the same cycle."

46.7 Application information

46.7.1 Module configuration

This section describes essential parts of the module configuration.

46.7.1.1 Configure System Memory Access Timeout Register (FR_SYMATOR)

To ensure reliable operation of the CC, the application must ensure that the TIMEOUT value in the System Memory Access Time-Out Register (FR_SYMATOR) and the CHI clock frequency f_{CHI} in MHz fulfill *Equation 68*[p].

p. See *Section 46.3: Controller Host Interface clocking* for all constraints of minimum CHI clock frequency.

Equation 68

$$0 \leq \text{SYMATOR[TIMEOUT]} \leq \lfloor 0.45 \cdot f_{CHI} - 8 \rfloor$$

For a given SYMATOR[TIMEOUT] value, f_{CHI} can be increased without causing unreliable operation of the CC. The same holds for reducing the SYMATOR[TIMEOUT] value for a given f_{CHI}.

Some examples for maximum values of the SYMATOR[TIMEOUT] for a minimum CHI frequency are given in *Table 929*.

Table 929. Maximum SYMATOR[TIMEOUT] examples

f_{CHI}	SYMATOR[TIMEOUT]	f_{CHI}	SYMATOR[TIMEOUT]
≥ 18 MHz	0	≥ 100 MHz	≤ 37
≥ 23 MHz	≤ 2	≥ 120 MHz	≤ 46
≥ 27 MHz	≤ 4	≥ 140 MHz	≤ 55
≥ 32 MHz	≤ 6	≥ 160 MHz	≤ 64
≥ 60 MHz	≤ 19	≥ 180 MHz	≤ 73
≥ 80 MHz	≤ 28	≥ 200 MHz	≤ 82

46.7.1.1.1 System bus wait state constraints

The SYMATOR[TIMEOUT] value corresponds directly to a certain acceptable number of wait states on the system bus.

For single-channel configurations and situations in which the sync frame table generation functionality is *not* used (FR_SFTCCSR[SDVEN,SIDEN] = 0), no timeout will be detected if fewer than 2 × SYMATOR[TIMEOUT] + 1 wait states will be seen on the system bus for each system bus access.

For dual-channel configurations, or situations in which the sync frame table generation functionality is used, no timeout will be detected if fewer than SYMATOR[TIMEOUT] – 1 wait states will be seen on the system bus for each system bus access.

46.7.1.2 Configure data field offsets

The data field offsets are located in the Message Buffer Data Field Offset Registers (FR_MBDOR*n*) and Receive FIFO Start Data Offset Register (FR_RFSDOR). The application has to configure the data field offset values for all message buffers that are used.

When the module is enabled, the initilization value of the FR_MBDOR*n*[MBDO] and FR_RFSDOR[SDO] is 0. This value is considered to be illegal (see *Section 46.6.19.1.1*).

46.7.2 Initialization sequence

This section describes the steps required to initialize the CC. The first subsection describes the steps required after a system reset; the second section describes the steps required after preceding shutdown of the CC.

46.7.2.1 Module initialization

This section describes the module-related initialization steps after a system reset.

1. Configure the CC.
   a) Configure the control bits in the Module Configuration Register (FR_MCR).
   b) Configure the system memory base address in the System Memory Base Address Register (FR_SYMBADR).
2. Enable the CC.
   a) Write 1 to the module enable bit MEN in the Module Configuration Register (FR_MCR).

The CC now enters Normal mode. The application can commence with the protocol initialization described in *Section 46.7.2.2*.

46.7.2.2 Protocol initialization

This section describes the protocol-related initialization steps.

1. Configure the Protocol Engine.
   a) Issue CONFIG command via the Protocol Operation Control Register (FR_POCR).
   b) Wait for *POC:config* in the Protocol Status Register 0 (FR_PSR0).
   c) Configure the FR_PCR0,..., FR_PCR30 registers to set all protocol parameters.
2. Configure the message buffers and FIFOs.
   a) Set the number of message buffers used and the message buffer segmentation in the Message Buffer Segment Size and Utilization Register (FR_MBSSUTR).
   b) Define the message buffer data size in the Message Buffer Data Size Register (FR_MBDSR)
   c) Configure each message buffer by setting the configuration values in the Message Buffer Configuration, Control, Status Registers (FR_MBCCSR*n*), Message Buffer Cycle Counter Filter Registers (FR_MBCCFR*n*), Message Buffer Frame ID Registers (FR_MBFIDR*n*), and Message Buffer Index Registers (FR_MBIDXR*n*)
   d) Configure the FIFOs.
   e) Issue CONFIG_COMPLETE command via the Protocol Operation Control Register (FR_POCR).
   f) Wait for *POC:ready* in the Protocol Status Register 0 (FR_PSR0).

After this sequence, the CC is configured as a FlexRay node and is ready to integrate into the FlexRay cluster.

46.7.2.3 CHI LRAM initialization

The CHI LRAM initialization is performed by the CC when it leaves Disabled mode. The unitization runs for 45 CHI clock cycles. All fields in the FR_MBCCSR*n*, FR_MBCCFR*n*, FR_MBFIDR*n*, FR_MBDOR*n* and LEETR*n* registers are initialized to 0. All application read or write accesses to these registers are delayed until the initialization is finished.

46.7.2.4 PE DRAM initialization

The PE DRAM initialization is performed by the CC in the *POC:default config* state. This initialization runs for 4.8 µs and will delay the state transition from *POC:default config* into *POC:config*.

46.7.3 Memory error injection out of *POC:default config*

This section provides information for application-driven memory error injection out if *POC:default config.* The CC provides means to inject memory errors from the application without any impacts to the internal protocol operation of the CC.

46.7.3.1 CHI LRAM error injection out of *POC:default config*

The CC will never perform any internal read access from the LRAM ECC Error Test Registers (FR_LEETR*n*). Any memory errors injected into these CHI LRAM locations will never be detected by internal access, independent from the protocol state.

The application should use these registers and related CHI LRAM location to inject memory errors into the CHI LRAM. The injection sequence is described in *Section 46.6.25.1*.

46.7.3.2 PE DRAM error injection out of *POC:default config*

The CC will never perform any internal read access from the PE DRAM address 0x70. This is the only PE DRAM address writable by the application out of the *POC:default config* state.

The application should use these PE DRAM locations to inject memory errors into the PE DRAM. The injection sequence is described in *Section 46.6.25.2*.

46.7.4 Shutdown sequence

This section describes a secure shutdown sequence to stop the CC gracefully. The main targets of this sequence are

- Finish all ongoing reception and transmission
- Do not corrupt FlexRay bus and do not disturb ongoing FlexRay bus communication

For a graceful shutdown, the application should perform the following tasks:

1. Disable all enabled message buffers.
   a) Repeatedly write 1 to FR_MBCCSR*n*[EDT] until FR_MBCCSR*n*[EDS] == 0.
2. Stop the Protocol Engine.
   a) Issue HALT command via Protocol Operation Control Register (FR_POCR).
   b) Wait for *POC:halt* in Protocol Status Register 0 (FR_PSR0).

46.7.5 Number of usable message buffers

This section describes the required minimum CHI clock frequency for a specified number of utilized message buffers configured in the Message Buffer Segment Size and Utilization Register (FR_MBSSUTR), a configured mini-slot length *gdMinislot*, and a configured nominal macrotick length *gdMacrotick*[q].

Additional constraints for the minimum CHI clock frequency are given in *Section 46.3*.

The CC uses a sequential search algorithm to determine the individual message buffer assigned or subscribed to the next slot. This search is started at the start of the slot and must be finished before the start of the next slot.

q. see *Section 46.3: Controller Host Interface clocking* for all constraints of minimum CHI clock frequency.

The shortest FlexRay slot is a corrected empty dynamic slot. A corrected empty dynamic slot is a mini-slot and consists of *gdMinislot* corrected macroticks with a duration of *gdMacrotick*. The minimum duration of an corrected macrotick is $gdMacrotick_{min}$ = 39 µT. This results in a minimum length of an correct slot.

Equation 69

$$\Delta_{slotmin} = 39 \cdot pdMicrotick \cdot gdMinislot$$

The message buffer search engine runs on the CHI clock and evaluates one individual message buffer per CHI clock cycle. For internal status update operations and to account for clock domain crossing jitter, an additional amount of 27 CHI clock cycles is required to ensure correct search engine operation.

For a given number of utilized message buffers FR_MBSSUTR[LAST_MB_UTIL] + 1 and for a given CHI clock frequency f_{chi}, this results in a search duration of

Equation 70

$$\Delta_{search} = \frac{1}{f_{chi}} \cdot (FR_MBSSUTR[LAST_MB_UTIL]+27)$$

The message buffer search must be finished within one slot which requires that *Equation 71* must be fulfilled:

Equation 71

$$\Delta_{search} \leq \Delta_{slotmin}$$

This results in the formula given in *Equation 72* which determines the required minimum CHI frequency for a given number of message buffers that are utilized.

Equation 72

$$f_{chi} \geq \frac{(FR_MBSSUTR[LAST_MB_UTIL]+27)}{39 \cdot pdMicrotick \cdot gdMinislot}$$

The required minimum CHI Clock frequency for a selected set of relevant protocol parameters and for the LAST_MB_UTIL field in the Message Buffer Segment Size and Utilization Register (FR_MBSSUTR) set to 63 is given in *Table 930*.

Table 930. Minimum CHI clock frequency[MHz] examples (64 message buffers used)

pdMicrotick [ns]	gdMinislot					
	2	3	4	5	6	7
25.0	46.7	31.2	23.4	18.7	15.6	13.4
50.0	23.4	15.6	11.7	9.4	7.8	6.7

Note: If the minimum CHI frequency is not met the CHIERFR[MBS_EF] flag is set. Refer to Section 46.5.2.17: CHI Error Flag Register (FR_CHIERFR) for details.

46.7.6 Protocol Control Command Execution

This section considers the issues of the protocol control command execution.

The application issues any of the protocol control commands listed in the POCCMD field of *Table 810* by writing the command to the POCCMD field of the Protocol Operation Control Register (FR_POCR). As a result the CC sets the BSY bit while the command is transferred to the PE. When the PE has accepted the command, the BSY flag is cleared. All commands are accepted by the PE.

The PE maintains a protocol command vector. For each command that was accepted by the PE, the PE sets the corresponding command bit in the protocol command vector. If a command is issued while the corresponding command bit is set, the command is not queued and is lost.

If the command execution block of the PE is idle, it selects the next accepted protocol command with the highest priority from the current protocol command vector according to the protocol control command priorities given in *Table 931*. If the current protocol state does not allow the execution of this protocol command (see POC state changes in *FlexRay Communications System Protocol Specification, Version 2.1 Rev A*) the CC asserts the illegal protocol command interrupt flag IPC_IF in the Protocol Interrupt Flag Register 1 (FR_PIFR1). The protocol command is not executed in this case.

Some protocol commands may be interrupted by other commands or the detection of a fatal protocol error as indicated by *Table 931*. If the application issues the FREEZE or READY command, or if the PE detects a fatal protocol error, some commands already stored in the command vector will be removed from this vector.

Table 931. Protocol control command priorities

Protocol command	Priority	Interrupted by	Cleared and terminated by
FREEZE	(highest) 1	none	
READY	2		
CONFIG_COMPLETE	3		
ALL_SLOTS	4	FREEZE, READY, CONFIG_COMPLET, fatal protocol error	FREEZE, READY, CONFIG_COMPLETE, fatal protocol error
ALLOW_COLDSTART	5		
RUN	6		FREEZE, fatal protocol error
WAKEUP	7		FREEZE, fatal protocol error
DEFAULT_CONFIG	8		FREEZE, fatal protocol error
CONFIG	9		
HALT	(lowest) 10		FREEZE, READY, CONFIG_COMPLETE, fatal protocol error

46.7.7 Message buffer search on simple message buffer configuration

This section describes the message buffer search behavior for a simplified message buffer configuration. The FIFO behavior is not considered in this section.

46.7.7.1 Simple message buffer configuration

A simple message buffer configuration is a configuration that has at most one transmit message buffer and at most one receive message buffer assigned to a slot *s*. The simple configuration used in this section utilizes two message buffers, one single buffered transmit message buffer and one receive message buffer.

The transmit message buffer has the message buffer number *t* and has following configuration.

Table 932. Transmit buffer configuration

Register	Field	Value	Description
FR_MBCCSR*t*	MTD	1	Transmit buffer
FR_MBCCFR*t*	MTM	0	Event Transition mode
	CHA	1	Assigned to channel A
	CHB	0	Not assigned to channel B
	CCFE	1	Cycle counter filter enabled
	CCFMSK	000011	Cycle set = {4n} = {0,4,8,12,…}
	CCFVAL	000000	
FR_MBFIDR*t*	FID	S	Assigned to slot s

The availability of data in the transmit buffer is indicated by the commit bit FR_MBCCSRt[CMT] and the lock bit FR_MBCCSRt[LCKS].

The receive message buffer has the message buffer number *r* and has following configuration.

Table 933. Receive buffer configuration

Register	Field	Value	Description
FR_MBCCSR*r*	MTD	0	Receive buffer
FR_MBCCFR*r*	MTM	—	n/a
	CHA	1	Assigned to channel A
	CHB	0	Not assigned to channel B
	CCFE	1	Cycle counter filter enabled
	CCFMSK	000001	Cycle set = {2n} = {0,2,4,6,…}
	CCFVAL	000000	
FR_MBFIDR*r*	FID	S	Subscribed slot

Furthermore the assumption is that both message buffers are enabled (FR_MBCCSR*t*[EDS] = 1 and FR_MBCCSR*r*[EDS] = 1).

Note: The cycle set {4n+2} = {2,6,10,…} is assigned to the receive buffer only.

The cycle set {4n} = {0,4,8,12,…} is assigned to both buffers.

46.7.7.2 Behavior in static segment

In this case, both message buffers are assigned to a slot *s* in the *static* segment.

The configuration of a transmit buffer for a static slot *s* assigns this slot to the node as a transmit slot. The FlexRay protocol requires:

- When a slot occurs, if the slot is assigned to a node on a channel that node must transmit either a normal frame or a null frame on that channel. Specifically, a null frame will be sent if there is no data ready, or if there is no match on a transmit filter (cycle counter filtering, for example).

Regardless of the availability of data and the cycle counter filter, the node will transmit a frame in the static slot *s*. In any case, the result of the message buffer search will be the transmit message buffer *t*. The receive message buffer *r* will not be found, no reception is possible.

46.7.7.3 Behavior in dynamic segment

In this case, both message buffers are assigned to a slot *s* in the *dynamic* segment. The FlexRay protocol requires:

- When a slot occurs, if a slot is assigned to a node on a channel that node only transmits a frame on that channel if there is data ready and there is a match on relevant transmit filters (no null frames are sent).

The transmission of a frame in the dynamic segment is determined by the availability of data and the match of the cycle counter filter of the transmit message buffer.

46.7.7.3.1 Transmit Data Not Available

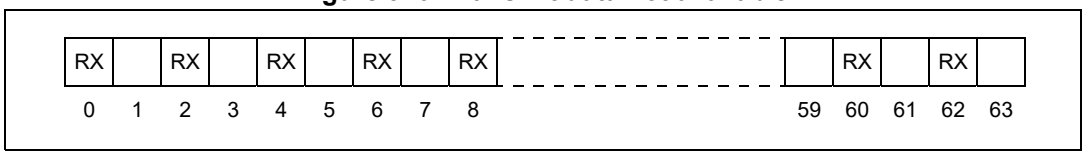
If transmit data are not available, that is the transmit buffer is not committed FR_MBCCSR*t*[CMT]=0 or locked FR_MBCCSR*t*[LCKS]=1, or both.

a) For the cycles in the set {4n}, which is assigned to both buffers, the receive buffer will be found and the node can receive data, and,

b) For the cycles in the set {4n+2}, which is assigned to the receive buffer only, the receive buffer will be found and the node can receive data.

The receive cycles are shown in *Figure 970*.

Figure 970. Transmit data not available



46.7.7.3.2 Transmit data available

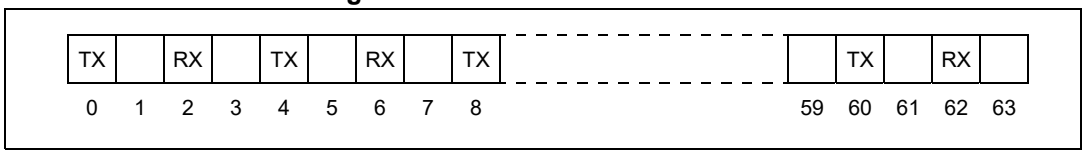
If transmit data are *available,* that is, the transmit buffer is committed FR_MBCCSR*t*[CMT] = 1 and not locked FR_MBCCSR*t*[LCKS] = 0,

a) For the cycles in the set {4n}, which is assigned to both buffers, the transmit buffer will be found and the node transmits data.

b) For the cycles in the set {4n+2}, which is assigned to the receive buffer only, the receive buffer will be found and the node can receive data.

The receive and transmit cycles are shown in *Figure 971*.

Figure 971. Transmit data available



47 Reset Generation Module (MC_RGM)

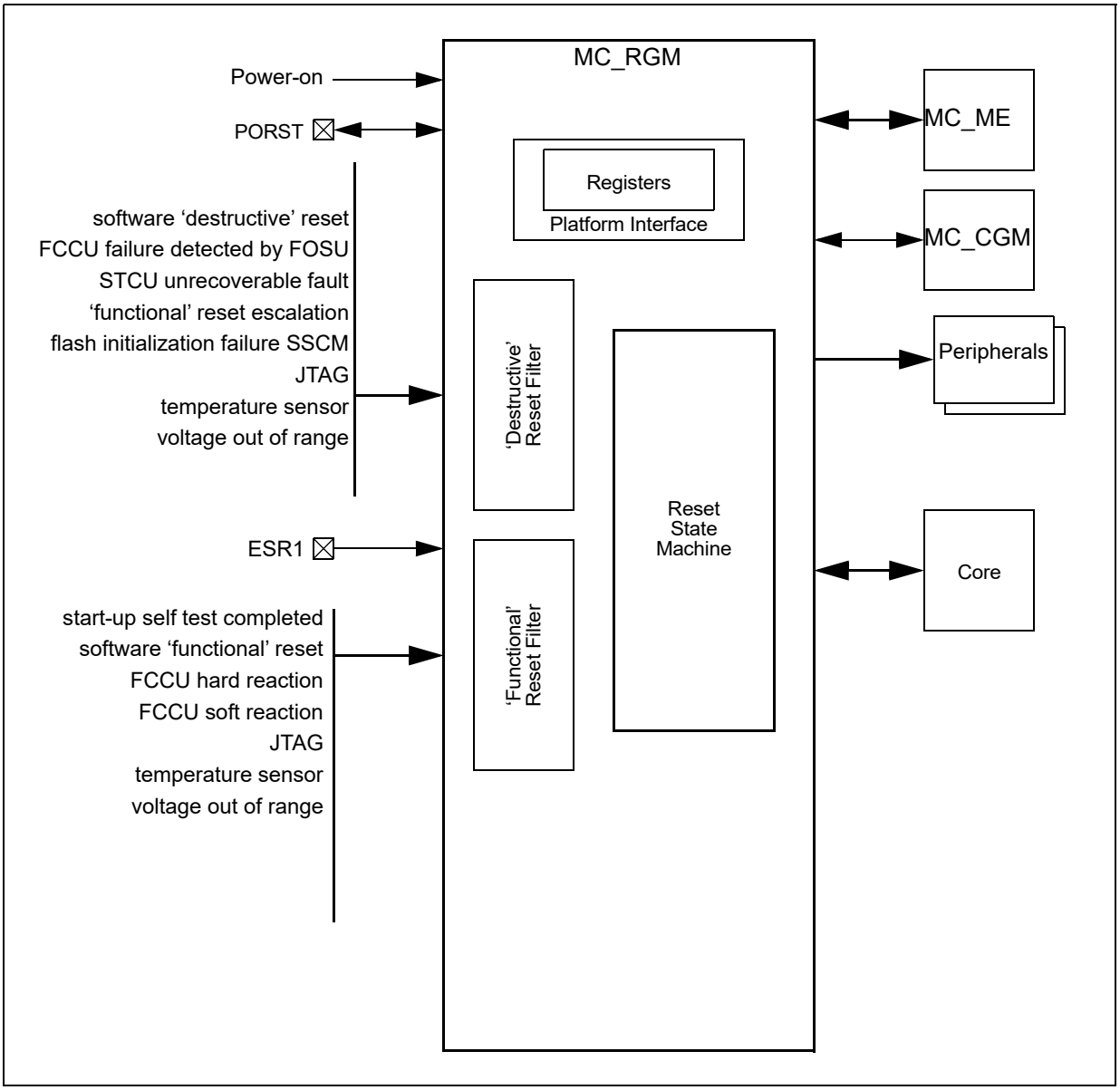
47.1 Introduction

47.1.1 Overview

The reset generation module (MC_RGM) centralizes the different reset sources and manages the reset sequence of the chip. It provides a register interface and the reset sequencer. Various registers are available to monitor and control the chip reset sequence. The reset sequencer is a state machine which controls the different phases (PHASE0, PHASE1, PHASE2, PHASE3, and IDLE) of the reset sequence and controls the reset signals generated in the system.

Figure 972 shows the MC_RGM block diagram.

Figure 972. MC_RGM block diagram



47.1.2 Features

The MC_RGM contains the functionality for the following features:

- 'Destructive' resets management
- 'Functional' resets management
- Signalling of reset events after each reset sequence (reset status flags)
- Conversion of reset events to SAFE mode or interrupt request events
- Short reset sequence configuration
- Configurable escalation of recurring 'functional' resets to 'destructive' reset
- Configurable escalation of recurring 'destructive' resets to keep chip in reset state until next power-on reset

47.1.3 Reset sources

The different reset sources are organized into two families: ‘destructive’ and ‘functional’.

- A ‘destructive’ reset source is associated with an event related to a critical, usually hardware, error or dysfunction. When a ‘destructive’ reset event occurs, the full reset sequence is applied to the chip starting from PHASE0. This resets the full chip ensuring a safe start-up state for both digital and analog modules, and the memory content must be considered to be unknown except in the software ‘destructive’ reset and external power-on reset cases, which do preserve the system memory content if the chip is not configured to execute a self-test on power-on, ‘destructive’, and external resets. ‘Destructive’ resets are:
  - Power-on reset
  - External power-on reset
  - Software ‘destructive’ reset
  - FCCU failure to react reset
  - STCU2 unrecoverable fault
  - ‘Functional’ reset escalation
  - Flash initialization failure
  - SSCM ’destructive’ reset request
  - JTAG ‘destructive’ reset request
  - Temperature sensor ‘destructive’ reset
  - Voltage out of range ‘destructive’ reset
- A ‘functional’ reset source is associated with an event related to a less-critical, usually non-hardware, error or dysfunction. When a ‘functional’ reset event occurs, a partial reset sequence is applied to the chip starting from PHASE1. In this case, most digital modules are reset normally, while the state of analog modules or specific digital modules (for example, debug modules, flash modules) as well as the system memory content is preserved, except on an external reset in the case where the chip is configured to execute a self-test on power-on, ‘destructive’, and external resets. ‘Functional’ resets are:
  - ESR1 external reset
  - Start-up self test completed
  - Software ‘functional’ reset
  - FCCU hard reaction
  - FCCU soft reaction
  - JTAG ‘functional’ reset
  - Temperature sensor ‘functional’ reset
  - Voltage out of range ‘functional’ reset

When a reset is triggered, the MC_RGM state machine is activated and proceeds through the different phases (PHASEn states). Each phase is associated with a particular chip reset being provided to the system. A phase is completed when all corresponding phase completion gates from either the system or internal to the MC_RGM are acknowledged. The chip reset associated with the phase is then released, and the state machine proceeds to the next phase up to entering the IDLE phase. During this entire process, the MC_ME state machine is held in RESET mode. Only at the end of the reset sequence, when the IDLE phase is reached, does the MC_ME enter the DRUN mode.

Alternatively, it is possible for software to configure some reset source events to be converted from a reset to either a SAFE mode request issued to the MC_ME or to an interrupt issued to the core (see *Section 47.3.2.2: Destructive Event Reset Disable Register (RGM_DERD)* and *Section 47.3.2.3: Destructive Event Alternate Request Register (RGM_DEAR)* for 'destructive' resets. See *Section 47.3.2.6: Functional Event Reset Disable Register (RGM_FERD)* and *Section 47.3.2.7: Functional Event Alternate Request Register (RGM_FEAR)* for 'functional' resets).

47.2 External signal description

The MC_RGM has no external interface signal.

47.3 Memory map and register definition

47.3.1 Memory map

Table 934. MC_RGM memory map

Address	Name	Description	Size	Access			Location
				User	Supervisor	Test	
0x000	RGM_DES	'Destructive' Event Status	word	read	read/write(1)	read/write(1)	*Section 47.3.2.1*
0x010	RGM_DERD	'Destructive' Event Reset Disable	word	read	read/write(2)	read/write(2)	*Section 47.3.2.2*
0x020	RGM_DEAR	'Destructive' Event Alternate Request	word	read	read/write	read/write	*Section 47.3.2.3*
0x030	RGM_DBRE	'Destructive' Bidirectional Reset Enable	word	read	read	read	*Section 47.3.2.4*
0x300	RGM_FES	'Functional' Event Status	word	read	read/write(1)	read/write(1)	*Section 47.3.2.5*
0x310	RGM_FERD	'Functional' Event Reset Disable	word	read(2)	read/write(2)	read/write(2)	*Section 47.3.2.6*
0x320	RGM_FEAR	'Functional' Event Alternate Request	word	read	read/write	read/write	*Section 47.3.2.7*
0x330	RGM_FBRE	'Functional' Bidirectional Reset Enable	word	read	read/write	read/write	*Section 47.3.2.8*
0x340	RGM_FESS	'Functional' Event Short Sequence	word	read	read/write	read/write	*Section 47.3.2.9*

Table 934. MC_RGM memory map (continued)

Address	Name	Description	Size	Access			Location
				User	Supervisor	Test	
0x604	RGM_FRET	'Functional' Reset Threshold	byte	read	read/write	read/write	*Section 47.3.2.10*
0x608	RGM_DRET	'Destructive' Reset Threshold	byte	read	read/write	read/write	*Section 47.3.2.11*
0x610	RGM_PRST0	Peripheral Reset 0	word	read	read	read	*Section 47.3.2.11*
0x614	RGM_PRST1	Peripheral Reset 1	word	read	read	read	*Section 47.3.2.13*
0x618	RGM_PRST2	Peripheral Reset 2	word	read	read	read	*Section 47.3.2.14*
0x61C	RGM_PRST3	Peripheral Reset 3	word	read	read	read	*Section 47.3.2.15*
0x620	RGM_PRST4	Peripheral Reset 4	word	read	read	read	*Section 47.3.2.16*
0x624	RGM_PRST5	Peripheral Reset 5	word	read	read	read	*Section 47.3.2.17*
0x628	RGM_PRST6	Peripheral Reset 6	word	read	read	read	*Section 47.3.2.18*
0x62C	RGM_PRST7	Peripheral Reset 7	word	read	read	read	*Section 47.3.2.19*
0x630	RGM_PSTAT0	Peripheral Reset Status 0	word	read	read	read	*Section 47.3.2.20*
0x634	RGM_PSTAT1	Peripheral Reset Status 1	word	read	read	read	*Section 47.3.2.21*
0x638	RGM_PSTAT2	Peripheral Reset Status 2	word	read	read	read	*Section 47.3.2.22*
0x63C	RGM_PSTAT3	Peripheral Reset Status 3	word	read	read	read	*Section 47.3.2.23*
0x640	RGM_PSTAT4	Peripheral Reset Status 4	word	read	read	read	*Section 47.3.2.24*
0x644	RGM_PSTAT5	Peripheral Reset Status 5	word	read	read	read	*Section 47.3.2.25*
0x648	RGM_PSTAT6	Peripheral Reset Status 6	word	read	read	read	*Section 47.3.2.26*
0x64C	RGM_PSTAT7	Peripheral Reset Status 7	word	read	read	read	*Section 47.3.2.27*

1. individual bits cleared on writing '1'
2. write once: '0' = enable, '1' = disable.

Note: Any access to unused registers as well as write accesses to read-only registers:
- does not change register content
- causes a transfer error

47.3.2 Register descriptions

Unless otherwise noted, all registers may be accessed as 32-bit words, 16-bit half-words, or 8-bit bytes. The bytes are ordered according to big endian. For example, the RGM_DES[8:15] register bits may be accessed as a word at address offset 0x000, as a half-word at address offset 0x002, or as a byte at address offset 0x003.

Some fields may be read-only, and their reset value of '1' or '0' and the corresponding behavior cannot be changed.

47.3.2.1 Destructive Event Status Register (RGM_DES)

This register contains the status of the 'destructive' reset sources, including those configured to not generate a reset sequence. It can be accessed in read/write on either supervisor mode or test mode. It can be accessed in read only in user mode. Register bits are cleared on write '1'. This register is reset only on power-on.

Address: 0x000 Access: User read, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	F_VOR_DEST(1)	F_TSR_DEST	0	0	0	0	0	0	0
W								w1c	w1c							
POR	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	F_SSCM	0	0	0	F_JTAG_DEST	F_FIF	F_EDR	0	0	F_SUF	F_FFRR	F_SOFT_DEST	0	F_PORST	F_POR
W		w1c				w1c	w1c	w1c			w1c	w1c	w1c		w1c	w1c
POR	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Figure 973. Destructive Event Status Register (RGM_DES)

1. HVD6 occurrence will not raise the F_VOR_DEST flag

Table 935. RGM_DES register field descriptions

Field	Description
7 F_VOR_DEST	Flag for voltage out of range ‘destructive’ reset 0 No voltage out of range ‘destructive’ reset event has occurred since either the last clear or the last power-on reset assertion 1 A voltage out of range ‘destructive’ reset event has occurred
8 F_TSR_DEST	Flag for temperature sensor ‘destructive’ reset 0 No temperature sensor ‘destructive’ reset event has occurred since the last clear 1 A temperature sensor ‘destructive’ reset event has occurred
17 F_SSCM	Flag for SSCM ’destructive’ reset request 0 No SSCM ’destructive’ reset request event has occurred since either the last clear or the last power-on reset assertion 1 An SSCM ’destructive’ reset request event has occurred
21 F_JTAG_DEST	Flag for JTAG ‘destructive’ reset request 0 No JTAG ‘destructive’ reset request event has occurred since the last clear 1 A JTAG ‘destructive’ reset request event has occurred
22 F_FIF	Flag for flash initialization failure 0 No flash initialization failure event has occurred since the last clear 1 A flash initialization failure event has occurred
23 F_EDR	Flag for ‘functional’ reset escalation 0 No ‘functional’ reset escalation event has occurred since the last clear 1 A ‘functional’ reset escalation event has occurred
26 F_SUF	Flag for STCU unrecoverable fault 0 No STCU unrecoverable fault event has occurred since either the last clear or the last power-on reset assertion 1 An STCU unrecoverable fault event has occurred
27 F_FFRR	Flag for FCCU failure to react 0 No FCCU failure to react event has occurred since either the last clear or the last power-on reset assertion 1 An FCCU failure to react event has occurred
28 F_SOFT_DEST	Flag for software ’destructive’ reset(1) 0 No software ’destructive’ reset event has occurred since either the last clear or the last power-on reset assertion 1 An software ’destructive’ reset event has occurred
30 F_PORST	Flag for external power-on reset 0 No external power-on reset event has occurred since either the last clear or the last power-on reset assertion 1 An external power-on reset event has occurred
31 F_POR	Flag for power-on reset 0 No power-on reset event has occurred since either the last clear or the last power-on reset assertion 1 A power-on reset event has occurred

1. The software reset is generated by the Mode Entry module (see *Chapter 52: Mode Entry Module (MC_ME)*)

47.3.2.2 Destructive Event Reset Disable Register (RGM_DERD)

This register provides dedicated bits to disable particular ‘destructive’ reset sources. When a ‘destructive’ reset source is disabled, the associated ‘destructive’ event will trigger either a SAFE mode request or an interrupt request (see *Section 47.3.2.3: Destructive Event Alternate Request Register (RGM_DEAR)*). Supervisor and test modes allow for read/write access, whereas read access only is allowed in user mode. Each byte can be written only once after a ‘destructive’ or power-on reset. This register is reset only on power-on.

Caution: It is important to clear the RGM_DES register before setting any of the bits in the RGM_DERD register to ‘1’. Otherwise a redundant SAFE mode request or interrupt request may occur.

Address: 0x010 Access: User read, Supervisor read, Test read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	D_VOR_DEST	D_TSR_DEST	0	0	0	0	0	0	0
W																
POR	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	D_SSCM	0	0	0	D_JTAG_DEST	D_FIF	D_EDR	0	0	D_SUF	D_FFRR	D_SOFT_DEST	0	D_PORST	D_POR
W																
POR	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 974. Destructive Event Reset Disable Register (RGM_DERD)

Table 936. RGM_DERD register field descriptions

Field	Description
7 D_VOR_DEST	Disable voltage out of range ‘destructive’ reset 0 A voltage out of range ‘destructive’ reset event triggers a reset sequence
8 D_TSR_DEST	Disable temperature sensor ‘destructive’ reset 0 A temperature sensor ‘destructive’ reset event triggers a reset sequence 1 A temperature sensor ‘destructive’ reset event generates either a SAFE mode or an interrupt request depending on the value of RGM_DEAR.AR_TSR_DEST
17 D_SSCM	Disable SSCM ’destructive’ reset request 0 A SSCM ’destructive’ reset request event triggers a reset sequence 1 A SSCM ’destructive’ reset request event generates either a SAFE mode or an interrupt request depending on the value of RGM_DEAR.AR_SSCM
21 D_JTAG_DEST	Disable JTAG ‘destructive’ reset request 0 A JTAG ‘destructive’ reset request event triggers a reset sequence

Table 936. RGM_DERD register field descriptions (continued)

Field	Description
22 D_FIF	Disable flash initialization failure 0 A flash initialization failure event triggers a reset sequence
23 D_EDR	Disable ‘functional’ reset escalation 0 A ‘functional’ reset escalation event triggers a reset sequence
26 D_SUF	Disable STCU unrecoverable fault 0 A STCU unrecoverable fault event triggers a reset sequence
27 D_FFRR	Disable FCCU failure to react 0 A FCCU failure to react event triggers a reset sequence
28 D_SOFT_DEST	Disable software ’destructive’ reset(1) 0 A software ’destructive’ reset event triggers a reset sequence
30 D_PORST	Disable external power-on reset 0 A external power-on reset event triggers a reset sequence 1 A external power-on reset event generates either a SAFE mode or an interrupt request depending on the value of RGM_DEAR.AR_PORST
31 D_POR	Disable power-on reset 0 A power-on reset event triggers a reset sequence

1. The software reset is generated by the Mode Entry module (see Chapter *Mode Entry Module (MC_ME)*)

47.3.2.3 Destructive Event Alternate Request Register (RGM_DEAR)

This register defines an alternate request to be generated when a reset on a ‘destructive’ event has been disabled. The alternate request can be either a SAFE mode request to MC_ME or an interrupt request to the system. Supervisor and test modes allow for read/write access, whereas read access only is allowed in user mode. This register is reset only on power-on.

Address: 0x020 Access: User read, Supervisor read, Test read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	AR_TSR_DEST	0	0	0	0	0	0	0
W																
POR	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	AR_PORST	0
W																
POR	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 975. Destructive Event Alternate Request Register (RGM_DEAR)

Table 937. RGM_DEAR register field descriptions

Field	Description
8 AR_TSR_DEST	Alternate Request for temperature sensor ‘destructive’ reset 0 Generate a SAFE mode request on a temperature sensor ‘destructive’ reset event if the reset is disabled 1 Generate an interrupt request on a temperature sensor ‘destructive’ reset event if the reset is disabled
30 AR_PORST	Alternate Request for external power-on reset 0 Generate a SAFE mode request on a external power-on reset event if the reset is disabled 1 Generate an interrupt request on a external power-on reset event if the reset is disabled

47.3.2.4 Destructive Bidirectional Reset Enable Register (RGM_DBRE)

This register enables the generation of an external reset on ‘destructive’ reset. It can be accessed in read only mode. This register is reset only on power-on.

Address: 0x030 Access: User read, Supervisor read, Test read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	BE_VOR_DEST	BE_TSR_DEST	0	0	0	0	0	0	0
W																
POR	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	BE_SSCM	0	0	0	BE_JTAG_DEST	BE_FIF	BE_EDR	0	0	BE_SUF	BE_FFRR	BE_SOFT_DEST	0	BE_PORST	BE_POR
W																
POR	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 976. Destructive Bidirectional Reset Enable Register (RGM_DBRE)

Table 938. RGM_DBRE register field descriptions

Field	Description
7 BE_VOR_DEST	Bidirectional Reset Enable for voltage out of range ‘destructive’ reset 0 PAD_47 is asserted on voltage out of range ‘destructive’ reset event
8 BE_TSR_DEST	Bidirectional Reset Enable for temperature sensor ‘destructive’ reset 0 PAD_47 is asserted on temperature sensor ‘destructive’ reset event

Table 938. RGM_DBRE register field descriptions (continued)

Field	Description
17 BE_SSCM	Bidirectional Reset Enable for SSCM 'destructive' reset request 0 PAD_47 is asserted on SSCM 'destructive' reset request event
21 BE_JTAG_DEST	Bidirectional Reset Enable for JTAG 'destructive' reset request 0 PAD_47 is asserted on JTAG 'destructive' reset request event
22 BE_FIF	Bidirectional Reset Enable for flash initialization failure 0 PAD_47 is asserted on flash initialization failure event
23 BE_EDR	Bidirectional Reset Enable for 'functional' reset escalation 0 PAD_47 is asserted on 'functional' reset escalation event
26 BE_SUF	Bidirectional Reset Enable for STCU unrecoverable fault 0 PAD_47 is asserted on STCU unrecoverable fault event
27 BE_FFRR	Bidirectional Reset Enable for FCCU failure to react 0 PAD_47 is asserted on FCCU failure to react event
28 BE_SOFT_DEST	Bidirectional Reset Enable for software 'destructive' reset 0 PAD_47 is asserted on software 'destructive' reset event
30 BE_PORST	Bidirectional Reset Enable for external power-on reset 0 PAD_47 is asserted on external power-on reset event
31 BE_POR	Bidirectional Reset Enable for power-on reset 0 PAD_47 is asserted on power-on reset event

47.3.2.5 Functional Event Status Register (RGM_FES)

This register contains the status of the 'functional' reset sources, including those configured to not generate a reset sequence. Supervisor and test modes allow for read/write access, whereas read access only is allowed in user mode. Register bits are cleared on write '1' if the triggering event has already been cleared at the source. This register is reset only on power-on.

Note: If a 'functional' reset source is configured to generate a SAFE mode request or an interrupt request, software needs to clear the event in the source module at least three system clock cycles before it clears the associated RGM_FES status bit in order to avoid multiple SAFE mode requests or interrupts for the same event. In order to avoid having to count cycles, it is good practice for software to check whether the RGM_FES register has been properly cleared and clear it again if this is not the case.

Address: 0x300 Access: User read, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	F_VOR_FUNC(1)	F_TSR_FUNC	0	0	0	0	0	0	0
W								w1c	w1c							
POR	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	F_JTAG_FUNC	0	0	0	F_FCCU_SOFT	F_FCCU_HARD	0	F_SOFT_FUNC	F_ST_DONE	F_WAKEUP	0
W						w1c				w1c	w1c		w1c	w1c	w1c	
POR	0	0	0	0	0	0	0	0	0	0	0	0	0	0(1)	0	0

1. Reset value of flag is 1 after reset if STCU is selected to run and 0 if STCU is bypassed

Figure 977. Functional Event Status Register (RGM_FES)

1. HVD6 occurrence will not raise the F_VOR_FUNC flag

Table 939. Functional Event Status Register (RGM_FES) field descriptions

Field	Description
7 F_VOR_FUNC	Flag for Voltage out of range functional reset 0 No Voltage out of range functional reset event has occurred since either the last clear or the last power-on reset assertion. 1 A Voltage out of range functional reset event has occurred.
8 F_TSR_FUNC	Flag for temperature sensor functional reset 0 No temperature sensor functional reset event has occurred since either the last clear or the last power-on reset assertion. 1 A temperature sensor functional reset event has occurred.
21 F_JTAG_FUNC	Flag for JTAG ‘functional’ reset 0 No JTAG ‘functional’ reset event has occurred since either the last clear or the last power-on reset assertion. 1 A JTAG ‘functional’ reset event has occurred.
25 F_FCCU_SOFT	Flag for soft reaction request 0 No soft reaction request event has occurred since either the last clear or the last power-on reset assertion. 1 A soft reaction request event has occurred.

Table 939. Functional Event Status Register (RGM_FES) field descriptions (continued)

Field	Description
26 F_FCCU_HARD	Flag for hard reaction request 0 No hard reaction request event has occurred since either the last clear or the last power-on reset assertion. 1 A hard reaction request event has occurred.
28 F_SOFT_FUNC	Flag for Software 'functional' reset(1) or online selftest completion 0 No Software 'functional' reset event has occurred since either the last clear or the last power-on reset assertion. 1 A Software 'functional' reset event has occurred or software triggered selftest has completed.
29 F_ST_DONE	Flag for Self-test 'functional' reset 0 No Self-test 'functional' reset event has occurred since either the last clear or the last power-on reset assertion. 1 A Self-test 'functional' reset event has occurred.
30 F_WAKEUP	Flag for Wake-up 'functional' reset(2) 0 No Wake-up 'functional' reset event has occurred since either the last clear or the last power-on reset assertion. 1 A Wake-up 'functional' reset event has occurred.

1. The software reset is generated by the Mode Entry module (see *Chapter 52: Mode Entry Module (MC_ME)*)
2. The software reset is generated by the NMI

47.3.2.6 Functional Event Reset Disable Register (RGM_FERD)

This register provides dedicated bits to disable 'functional' reset sources.When a 'functional' reset source is disabled, the associated 'functional' event will trigger either a SAFE mode request or an interrupt request (see *Section 47.3.2.7: Functional Event Alternate Request Register (RGM_FEAR)*). Supervisor and test modes allow for read/write access, whereas read access only is allowed in user mode. Each byte can be written only once after power-on reset. This register is reset only on power-on and any 'destructive' reset.

Caution: It is important to clear the RGM_FES register before setting any of the bits in the RGM_FERD register to '1'. Otherwise a redundant SAFE mode request or interrupt request may occur.

Address: 0x310 Access: User read, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	D_VOR_FUNC	D_TSR_FUNC	0	0	0	0	0	0	0
W																
POR	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	D_JTAG_FUNC	0	0	0	D_FCCU_SOFT	D_FCCU_HARD	0	D_SOFT_FUNC	D_ST_DONE	D_WAKEUP	0
W																
POR	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 978. Functional Event Reset Disable Register (RGM_FERD)

Table 940. RGM_FERD register field descriptions

Field	Description
7 D_VOR_FUNC	Disable Voltage out of range functional reset 0 A Voltage out of range functional reset event triggers a reset sequence 1 A Voltage out of range functional reset event generates either a SAFE mode or an interrupt request depending on the value of RGM_FEAR.AR_VOR_FUNC
8 D_TSR_FUNC	Disable temperature sensor functional reset 0 A temperature sensor functional reset event triggers a reset sequence 1 A temperature sensor functional reset event generates either a SAFE mode or an interrupt request depending on the value of RGM_FEAR.AR_TSR_FUNC
21 D_JTAG_FUNC	Disable JTAG ‘functional’ reset 0 A JTAG ‘functional’ reset event triggers a reset sequence
25 D_FCCU_SOFT	Disable soft reaction request 0 A soft reaction request event triggers a reset sequence 1 A soft reaction request event generates a SAFE mode request
26 D_FCCU_HARD	Disable hard reaction request 0 A hard reaction request event triggers a reset sequence
28 D_SOFT_FUNC	Disable Software ’functional’ reset or online selftest completion 0 A Software ’functional’ reset event triggers a reset sequence 1 A Software ’functional’ reset event generates either a SAFE mode or an interrupt request depending on the value of RGM_FEAR.AR_SOFT_FUNC

Table 940. RGM_FERD register field descriptions (continued)

Field	Description
29 D_ST_DONE	Disable Self-test 'functional' reset 0 A Self-test 'functional' reset event triggers a reset sequence 1 A Self-test 'functional' reset event generates either a SAFE mode or an interrupt request depending on the value of RGM_FEAR.AR_ST_DONE
30 D_WAKEUP	Disable Wake-up 'functional' reset 0 A Wake-up 'functional' reset event triggers a reset sequence 1 A Wake-up 'functional' reset event generates either a SAFE mode or an interrupt request depending on the value of RGM_FEAR.AR_WAKEUP

47.3.2.7 Functional Event Alternate Request Register (RGM_FEAR)

This register defines an alternate request to be generated when a reset on a 'functional' event has been disabled. The alternate request can be either a SAFE mode request to MC_ME or an interrupt request to the system. Supervisor and test modes allow for read/write access, whereas read access only is allowed in user mode. This register is reset only on power-on and any 'destructive' reset.

Address: 0x320 Access: User read, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	AR_VOR_FUNC	AR_TSR_FUNC	0	0	0	0	0	0	0
W																
'Dest' Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	AR_FCCU_SOFT	0	0	0	0	AR_WAKEUP	0
W																
'Dest' Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 979. Functional Event Alternate Request Register (RGM_FEAR)

Table 941. RGM_FEAR register field descriptions

Field	Description
7 AR_VOR_FUNC	Alternate Request for Voltage out of range functional reset 0 Generate a SAFE mode request on a Voltage out of range functional reset event if the reset is disabled 1 Generate an interrupt request on a Voltage out of range functional reset event if the reset is disabled
8 AR_TSR_FUNC	Alternate Request for temperature sensor functional reset 0 Generate a SAFE mode request on a temperature sensor functional reset event if the reset is disabled 1 Generate an interrupt request on a temperature sensor functional reset event if the reset is disabled
25 AR_FCCU_SOFT	Alternate Request for soft reaction request 0 Generate a SAFE mode request on a soft reaction request event if the reset is disabled
30 AR_WAKEUP	Alternate Request for Wake-up 'functional' reset 0 Generate a SAFE mode request on a Wake-up 'functional' reset event if the reset is disabled 1 Generate an interrupt request on a Wake-up 'functional' reset event if the reset is disabled

47.3.2.8 Functional Bidirectional Reset Enable Register (RGM_FBRE)

This register enables the generation of an external reset on 'functional' reset. It can be accessed in read/write in either supervisor mode or test mode. It can be accessed in read in user mode.

This register is reset only on power-on and any 'destructive' reset.

Address: 0x330 Access: User read, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	BE_VOR_FUNC	BE_TSR_FUNC	0	0	0	0	0	0	0
W																
POR	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	BE_JTAG_FUNC	0	0	0	BE_FCCU_SOFT	BE_FCCU_HARD	0	BE_SOFT_FUNC	BE_ST_DONE	BE_WAKEUP	0
W																
POR	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 980. Functional Bidirectional Reset Enable Register (RGM_FBRE)

Table 942. Functional Event Status Register (RGM_FBRE) field descriptions

Field	Description
7 BE_VOR_FUNC	Bidirectional Reset Enable for Voltage out of range functional reset 0 PAD_47 is asserted on Voltage out of range functional reset event. 1 PAD_47 is not asserted on Voltage out of range functional reset event.
8 BE_TSR_FUNC	Bidirectional Reset Enable for temperature sensor functional reset 0 PAD_47 is asserted on temperature sensor functional reset event. 1 PAD_47 is not asserted on temperature sensor functional reset event.
21 BE_JTAG_FUNC	Bidirectional Reset Enable for JTAG ‘functional’ reset 0 PAD_47 is asserted on JTAG ‘functional’ reset event. 1 PAD_47 is not asserted on JTAG ‘functional’ reset event.
25 BE_FCCU_SOFT	Bidirectional Reset Enable for soft reaction request 0 PAD_47 is asserted on soft reaction request event. 1 PAD_47 is not asserted on soft reaction request event.
26 BE_FCCU_HARD	Bidirectional Reset Enable for hard reaction request 0 PAD_47 is asserted on hard reaction request event. 1 PAD_47 is not asserted on hard reaction request event.
28 BE_SOFT_FUNC	Bidirectional Reset Enable for Software ’functional’ reset or online selftest completion 0 PAD_47 is asserted on Software ’functional’ reset event. 1 PAD_47 is not asserted on Software ’functional’ reset event.
29 BE_ST_DONE	Bidirectional Reset Enable for Self-test ’functional’ reset 0 PAD_47 is asserted on Self-test ’functional’ reset event.
30 BE_WAKEUP	Bidirectional Reset Enable for Wake-up ’functional’ reset 0 PAD_47 is asserted on Wake-up ’functional’ reset event. 1 PAD_47 is not asserted on Wake-up ’functional’ reset event.

47.3.2.9 Functional Event Short Sequence Register (RGM_FESS)

This register defines which reset sequence is activated when a ‘functional’ reset sequence is triggered. The ‘functional’ reset sequence can either start from PHASE1 or from PHASE3, skipping PHASE1 and PHASE2; this could be useful for fast reset sequence, for example to skip flash reset.Supervisor and test modes allow for read/write access, whereas read access only is allowed in user mode. This register is reset only on power-on and any ‘destructive’ reset.

Address: 0x340 Access: User read, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	SS_VOR_FUNC	SS_TSR_FUNC	0	0	0	0	0	0	0
W																
POR	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	SS_JTAG_FUNC	0	0	0	SS_FCCU_SOFT	SS_FCCU_HARD	0	SS_SOFT_FUNC	SS_ST_DONE	SS_WAKEUP	0
W																
POR	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0

Figure 981. Functional Event Short Sequence Register (RGM_FESS)

Table 943. RGM_FESS register field descriptions

Field	Description
7 SS_VOR_FUNC	Short Sequence for Voltage out of range functional reset 0 The reset sequence triggered by a Voltage out of range functional reset event starts from PHASE1 1 The reset sequence triggered by a Voltage out of range functional reset event starts from PHASE3
8 SS_TSR_FUNC	Short Sequence for temperature sensor functional reset 0 The reset sequence triggered by a temperature sensor functional reset event starts from PHASE1 1 The reset sequence triggered by a temperature sensor functional reset event starts from PHASE3
21 SS_JTAG_FUNC	Short Sequence forJTAG ‘functional’ reset 0 The reset sequence triggered by a JTAG ‘functional’ reset event starts from PHASE1 1 The reset sequence triggered by a JTAG ‘functional’ reset event starts from PHASE3
25 SS_FCCU_SOFT	Short Sequence for soft reaction request 1 The reset sequence triggered by a soft reaction request event starts from PHASE3
26 SS_FCCU_HARD	Short Sequence for hard reaction request 0 The reset sequence triggered by a hard reaction request event starts from PHASE1

Table 943. RGM_FESS register field descriptions (continued)

Field	Description
28 SS_SOFT_FUNC	Short Sequence for Software 'functional' reset or online selftest completion 0 The reset sequence triggered by a Software 'functional' reset event starts from PHASE1 1 The reset sequence triggered by a Software 'functional' reset event starts from PHASE3
29 SS_ST_DONE	Short Sequence for Self-test 'functional' reset 0 The reset sequence triggered by a Self-test 'functional' reset event starts from PHASE1
30 SS_WAKEUP	Short Sequence for Wake-up 'functional' reset 0 The reset sequence triggered by a Wake-up 'functional' reset event starts from PHASE1 1 The reset sequence triggered by a Wake-up 'functional' reset event starts from PHASE3

47.3.2.10 Functional Reset Escalation Threshold Register (RGM_FRET)

This register sets the threshold for 'functional' reset escalation to a 'destructive' reset. Supervisor and test modes allow for read/write access, whereas read access only is allowed in user mode.

Writing a non-zero value to the FRET field will enable the 'functional' reset escalation function. Writing any value to this register will reset the 'functional' reset counter. See *Section 47.4.5* for details on the 'functional' reset escalation function. This register is reset only on power-on and any 'destructive' reset events.

Address: 0x604 Access: User read, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7
R	0	0	0	0	FRET			
W								
'Dest' Reset	0	0	0	0	0	1	0	0

Figure 982. Functional Reset Escalation Threshold Register (RGM_FRET)

Table 944. RGM_FRET register field descriptions

Field	Description
4:7 FRET	'Functional' Reset Escalation Threshold 0000 'functional' reset escalation function is disabled 0001 Number of 'functional' resets which will cause a 'destructive' reset if the RGM_FRET register is not written to beforehand 0010 Number of 'functional' resets which will cause a 'destructive' reset if the RGM_FRET register is not written to beforehand 0011 Number of 'functional' resets which will cause a 'destructive' reset if the RGM_FRET register is not written to beforehand 0100 Number of 'functional' resets which will cause a 'destructive' reset if the RGM_FRET register is not written to beforehand 0101 Number of 'functional' resets which will cause a 'destructive' reset if the RGM_FRET register is not written to beforehand 0110 Number of 'functional' resets which will cause a 'destructive' reset if the RGM_FRET register is not written to beforehand 0111 Number of 'functional' resets which will cause a 'destructive' reset if the RGM_FRET register is not written to beforehand 1000 Number of 'functional' resets which will cause a 'destructive' reset if the RGM_FRET register is not written to beforehand 1001 Number of 'functional' resets which will cause a 'destructive' reset if the RGM_FRET register is not written to beforehand 1010 Number of 'functional' resets which will cause a 'destructive' reset if the RGM_FRET register is not written to beforehand 1011 Number of 'functional' resets which will cause a 'destructive' reset if the RGM_FRET register is not written to beforehand 1100 Number of 'functional' resets which will cause a 'destructive' reset if the RGM_FRET register is not written to beforehand 1101 Number of 'functional' resets which will cause a 'destructive' reset if the RGM_FRET register is not written to beforehand 1110 Number of 'functional' resets which will cause a 'destructive' reset if the RGM_FRET register is not written to beforehand 1111 Number of 'functional' resets which will cause a 'destructive' reset if the RGM_FRET register is not written to beforehand

47.3.2.11 Destructive Reset Escalation Threshold Register (RGM_DRET)

This register sets the threshold for 'destructive' reset escalation keeping the chip in the reset state until the next power-on reset triggers a new reset sequence. Supervisor and test modes allow for read/write access, whereas read access only is allowed in user mode.

Writing a non-zero value to the DRET field will enable the 'destructive' reset escalation function. Writing any value to this register will reset the 'destructive' reset counter. See *Section 47.4.6* for details on the 'destructive' reset escalation function. This register is reset only on power-on.

Address: 0x608 Access: User read, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7
R	0	0	0	0	DRET			
W								
POR	0	0	0	0	1	0	0	0

Figure 983. Destructive Reset Escalation Threshold Register (RGM_DRET)

Table 945. RGM_DRET register field descriptions

Field	Description
4:7 DRET	‘Destructive’ Reset Escalation Threshold 0000 ‘destructive’ reset escalation function is disabled 0001 Number of ‘destructive’ resets which will keep the chip in the reset state until the next power-on reset triggers a new reset sequence if the RGM_DRET register is not written to beforehand 0010 Number of ‘destructive’ resets which will keep the chip in the reset state until the next power-on reset triggers a new reset sequence if the RGM_DRET register is not written to beforehand 0011 Number of ‘destructive’ resets which will keep the chip in the reset state until the next power-on reset triggers a new reset sequence if the RGM_DRET register is not written to beforehand 0100 Number of ‘destructive’ resets which will keep the chip in the reset state until the next power-on reset triggers a new reset sequence if the RGM_DRET register is not written to beforehand 0101 Number of ‘destructive’ resets which will keep the chip in the reset state until the next power-on reset triggers a new reset sequence if the RGM_DRET register is not written to beforehand 0110 Number of ‘destructive’ resets which will keep the chip in the reset state until the next power-on reset triggers a new reset sequence if the RGM_DRET register is not written to beforehand 0111 Number of ‘destructive’ resets which will keep the chip in the reset state until the next power-on reset triggers a new reset sequence if the RGM_DRET register is not written to beforehand 1000 Number of ‘destructive’ resets which will keep the chip in the reset state until the next power-on reset triggers a new reset sequence if the RGM_DRET register is not written to beforehand 1001 Number of ‘destructive’ resets which will keep the chip in the reset state until the next power-on reset triggers a new reset sequence if the RGM_DRET register is not written to beforehand 1010 Number of ‘destructive’ resets which will keep the chip in the reset state until the next power-on reset triggers a new reset sequence if the RGM_DRET register is not written to beforehand 1011 Number of ‘destructive’ resets which will keep the chip in the reset state until the next power-on reset triggers a new reset sequence if the RGM_DRET register is not written to beforehand 1100 Number of ‘destructive’ resets which will keep the chip in the reset state until the next power-on reset triggers a new reset sequence if the RGM_DRET register is not written to beforehand 1101 Number of ‘destructive’ resets which will keep the chip in the reset state until the next power-on reset triggers a new reset sequence if the RGM_DRET register is not written to beforehand 1110 Number of ‘destructive’ resets which will keep the chip in the reset state until the next power-on reset triggers a new reset sequence if the RGM_DRET register is not written to beforehand 1111 Number of ‘destructive’ resets which will keep the chip in the reset state until the next power-on reset triggers a new reset sequence if the RGM_DRET register is not written to beforehand

47.3.2.12 Peripheral Reset Register 0 (RGM_PRST0)

This register provides individual resets for various peripherals. Supervisor and test modes allow for read/write access, whereas read access only is allowed in user mode.Please refer to *Table 946* for details.

Address: 0x610 Access: User read, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	PIT_RTC_0_RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	SIUL_RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 984. Peripheral Reset Register 0 (RGM_PRST0)

Table 946. RGM_PRST0 register field descriptions

Field	Description
1 PIT_RTC_0_RST	PIT_RTC_0 reset Writing a ‘1’ to this bit resets the PIT_RTC_0 peripheral. Writing a ‘0’ to this bit releases the reset to the PIT_RTC_0 peripheral if the bit’s current value is ‘1’, otherwise it has no effect. 0 No forced reset of peripheral 1 Forced reset of peripheral
16 SIUL_RST	SIUL reset Writing a ‘1’ to this bit resets the SIUL peripheral. Writing a ‘0’ to this bit releases the reset to the SIUL peripheral if the bit’s current value is ‘1’, otherwise it has no effect. 0 No forced reset of peripheral 1 Forced reset of peripheral

47.3.2.13 Peripheral Reset Register 1 (RGM_PRST1)

This register provides individual resets for various peripherals. Supervisor and test modes allow for read/write access, whereas read access only is allowed in user mode. Please refer to *Table 947* for details.

Address: 0x614 Access: User read, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	CCCU_RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	FCCU_RST	0	0	CRC_0_RST	0	DMAMUX_0_RST	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 985. Peripheral Reset Register 1 (RGM_PRST1)

Table 947. RGM_PRST1 field descriptions

Field	Description
1 CCCU_RST	CCCU reset Writing a ‘1’ to this bit resets the CCCU peripheral. Writing a ‘0’ to this bit releases the reset to the CCCU peripheral if the bit’s current value is ‘1’, otherwise it has no effect. 0 No forced reset of peripheral 1 Forced reset of peripheral
22 FCCU_RST	FCCU reset Writing a ‘1’ to this bit resets the FCCU peripheral. Writing a ‘0’ to this bit releases the reset to the FCCU peripheral if the bit’s current value is ‘1’, otherwise it has no effect. 0 No forced reset of peripheral 1 Forced reset of peripheral
25 CRC_0_RST	CRC_0 reset Writing a ‘1’ to this bit resets the CRC_0 peripheral. Writing a ‘0’ to this bit releases the reset to the CRC_0 peripheral if the bit’s current value is ‘1’, otherwise it has no effect. 0 No forced reset of peripheral 1 Forced reset of peripheral
27 DMAMUX_0_RST	DMAMUX_0 reset Writing a ‘1’ to this bit resets the DMAMUX_0 peripheral. Writing a ‘0’ to this bit releases the reset to the DMAMUX_0 peripheral if the bit’s current value is ‘1’, otherwise it has no effect. 0 No forced reset of peripheral 1 Forced reset of peripheral

47.3.2.14 Peripheral Reset Register 2 (RGM_PRST2)

This register provides individual resets for various peripherals. Supervisor and test modes allow for read/write access, whereas read access only is allowed in user mode. Please refer to *Table 948* for details.

Address: 0x618 Access: User read, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	LINFlexD_0_RST	LINFlexD_2_RST	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	MCAN_0_2_RAM_RST(1)	0	0	0	MCAN_0_RST(1)	MCAN_2_RST(1)	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 986. Peripheral Reset Register 2 (RGM_PRST2)

1. RGM is able to reset CAN RAM controller and not CAN RAM content. CAN RAM content cannot be accessed if corresponding CAN RAM controller is kept under reset by RGM bit.

Table 948. RGM_PRST2 register field descriptions

Field	Description
3 LINFlexD_0_RST	LINFlexD_0 reset Writing a ‘1’ to this bit resets the LINFlexD_0 peripheral. Writing a ‘0’ to this bit releases the reset to the LINFlexD_0 peripheral if the bit’s current value is ‘1’, otherwise it has no effect. 0 No forced reset of peripheral 1 Forced reset of peripheral
4 LINFlexD_2_RST	LINFlexD_2 reset Writing a ‘1’ to this bit resets the LINFlexD_2 peripheral. Writing a ‘0’ to this bit releases the reset to the LINFlexD_2 peripheral if the bit’s current value is ‘1’, otherwise it has no effect. 0 No forced reset of peripheral 1 Forced reset of peripheral
21 MCAN_0_2_RAM_RST	MCAN_0_2_RAM reset Writing a ‘1’ to this bit resets the MCAN_0_2_RAM peripheral. Writing a ‘0’ to this bit releases the reset to the MCAN_0_2_RAM peripheral if the bit’s current value is ‘1’, otherwise it has no effect. 0 No forced reset of peripheral 1 Forced reset of peripheral

Table 948. RGM_PRST2 register field descriptions (continued)

Field	Description
25 MCAN_0_RST	MCAN_0 reset Writing a '1' to this bit resets the MCAN_0 peripheral. Writing a '0' to this bit releases the reset to the MCAN_0 peripheral if the bit's current value is '1', otherwise it has no effect. 0 No forced reset of peripheral 1 Forced reset of peripheral
26 MCAN_2_RST	MCAN_2 reset Writing a '1' to this bit resets the MCAN_2 peripheral. Writing a '0' to this bit releases the reset to the MCAN_2 peripheral if the bit's current value is '1', otherwise it has no effect. 0 No forced reset of peripheral 1 Forced reset of peripheral

47.3.2.15 Peripheral Reset Register 3 (RGM_PRST3)

This register provides individual resets for various peripherals. Supervisor and test modes allow for read/write access, whereas read access only is allowed in user mode. Please refer to *Table 949* for details.

Address: 0x61C Access: User read, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	ADCSAR_dig_0_RST	ADCSAR_dig_1_RST	ADCSAR_dig_2_RST	0	ADCSAR_seq_0_RST	ADCSAR_seq_1_RST	ADCSAR_seq_2_RST	0	0	0	0	0	0	0	0	ADCSAR_dig_b_RST
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	ADCSAR_seq_b_RST	0	0	0	FLEXRAY_RST	0	0	SENT_0_RST	0	0	0	0	DSPI_0_RST	DSPI_1_RST	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 987. Peripheral Reset Register 3 (RGM_PRST3)

Table 949. RGM_PRST3 register field descriptions

Field	Description
0 ADCSAR_dig_0_RST	ADCSAR_dig_0 reset Writing a ‘1’ to this bit resets the ADCSAR_dig_0 peripheral. Writing a ‘0’ to this bit releases the reset to the ADCSAR_dig_0 peripheral if the bit’s current value is ‘1’, otherwise it has no effect. **Note:** This peripheral is coupled with RGM_PRST3.ADCSAR_seq_0_RST therefore they have to be identically set. 0 No forced reset of peripheral 1 Forced reset of peripheral
1 ADCSAR_dig_1_RST	ADCSAR_dig_1 reset Writing a ‘1’ to this bit resets the ADCSAR_dig_1 peripheral. Writing a ‘0’ to this bit releases the reset to the ADCSAR_dig_1 peripheral if the bit’s current value is ‘1’, otherwise it has no effect. **Note:** This peripheral is coupled with RGM_PRST3.ADCSAR_seq_1_RST therefore they have to be identically set. 0 No forced reset of peripheral 1 Forced reset of peripheral
2 ADCSAR_dig_2_RST	ADCSAR_dig_2 reset Writing a ‘1’ to this bit resets the ADCSAR_dig_2 peripheral. Writing a ‘0’ to this bit releases the reset to the ADCSAR_dig_2 peripheral if the bit’s current value is ‘1’, otherwise it has no effect. **Note:** This peripheral is coupled with RGM_PRST3.ADCSAR_seq_2_RST therefore they have to be identically set. 0 No forced reset of peripheral 1 Forced reset of peripheral
4 ADCSAR_seq_0_RST	ADCSAR_seq_0 reset Writing a ‘1’ to this bit resets the ADCSAR_seq_0 peripheral. Writing a ‘0’ to this bit releases the reset to the ADCSAR_seq_0 peripheral if the bit’s current value is ‘1’, otherwise it has no effect. 0 No forced reset of peripheral 1 Forced reset of peripheral
5 ADCSAR_seq_1_RST	ADCSAR_seq_1 reset Writing a ‘1’ to this bit resets the ADCSAR_seq_1 peripheral. Writing a ‘0’ to this bit releases the reset to the ADCSAR_seq_1 peripheral if the bit’s current value is ‘1’, otherwise it has no effect. 0 No forced reset of peripheral 1 Forced reset of peripheral
6 ADCSAR_seq_2_RST	ADCSAR_seq_2 reset Writing a ‘1’ to this bit resets the ADCSAR_seq_2 peripheral. Writing a ‘0’ to this bit releases the reset to the ADCSAR_seq_2 peripheral if the bit’s current value is ‘1’, otherwise it has no effect. 0 No forced reset of peripheral 1 Forced reset of peripheral

Table 949. RGM_PRST3 register field descriptions (continued)

Field	Description
15 ADCSAR_dig_b_RST	ADCSAR_dig_b reset Writing a ‘1’ to this bit resets the ADCSAR_dig_b peripheral. Writing a ‘0’ to this bit releases the reset to the ADCSAR_dig_b peripheral if the bit’s current value is ‘1’, otherwise it has no effect. **Note:** This peripheral is coupled with RGM_PRST3.ADCSAR_seq_b_RST therefore they have to be identically set. 0 No forced reset of peripheral 1 Forced reset of peripheral
16 ADCSAR_seq_b_RST	ADCSAR_seq_b reset Writing a ‘1’ to this bit resets the ADCSAR_seq_b peripheral. Writing a ‘0’ to this bit releases the reset to the ADCSAR_seq_b peripheral if the bit’s current value is ‘1’, otherwise it has no effect. 0 No forced reset of peripheral 1 Forced reset of peripheral
20 FLEXRAY_RST	FLEXRAY reset Writing a ‘1’ to this bit resets the FLEXRAY peripheral. Writing a ‘0’ to this bit releases the reset to the FLEXRAY peripheral if the bit’s current value is ‘1’, otherwise it has no effect. 0 No forced reset of peripheral 1 Forced reset of peripheral
23 SENT_0_RST	SENT_0 reset Writing a ‘1’ to this bit resets the SENT_0 peripheral. Writing a ‘0’ to this bit releases the reset to the SENT_0 peripheral if the bit’s current value is ‘1’, otherwise it has no effect. 0 No forced reset of peripheral 1 Forced reset of peripheral
28 DSPI_0_RST	DSPI_0 reset Writing a ‘1’ to this bit resets the DSPI_0 peripheral. Writing a ‘0’ to this bit releases the reset to the DSPI_0 peripheral if the bit’s current value is ‘1’, otherwise it has no effect. 0 No forced reset of peripheral 1 Forced reset of peripheral
29 DSPI_1_RST	DSPI_1 reset Writing a ‘1’ to this bit resets the DSPI_1 peripheral. Writing a ‘0’ to this bit releases the reset to the DSPI_1 peripheral if the bit’s current value is ‘1’, otherwise it has no effect. 0 No forced reset of peripheral 1 Forced reset of peripheral

47.3.2.16 Peripheral Reset Register 4 (RGM_PRST4)

This register provides individual resets for various peripherals. Supervisor and test modes allow for read/write access, whereas read access only is allowed in user mode. Please refer to *Table 950* for details.

Address: 0x620 Access: User read, Supervisor read/write, Test read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	PWM_SYNCHR_RST	0	FLEXPWM_2_RST	FLEXPWM_3_RST	FLEXPWM_0_RST	FLEXPWM_1_RST	0	CTU_1_RST	0	0	0	0	ETIMER_2_RST	ETIMER_3_RST	ETIMER_0_RST	ETIMER_1_RST
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 988. Peripheral Reset Register 4 (RGM_PRST4)

Table 950. RGM_PRST4 register field descriptions

Field	Description
16 PWM_SYNCHR_RST	PWM_SYNCHR reset Writing a ‘1’ to this bit resets the PWM_SYNCHR peripheral. Writing a ‘0’ to this bit releases the reset to the PWM_SYNCHR peripheral if the bit’s current value is ‘1’, otherwise it has no effect. 0 No forced reset of peripheral 1 Forced reset of peripheral
18 FLEXPWM_2_RST	FLEXPWM_2 reset Writing a ‘1’ to this bit resets the FLEXPWM_2 peripheral. Writing a ‘0’ to this bit releases the reset to the FLEXPWM_2 peripheral if the bit’s current value is ‘1’, otherwise it has no effect. 0 No forced reset of peripheral 1 Forced reset of peripheral
19 FLEXPWM_3_RST	FLEXPWM_3 reset Writing a ‘1’ to this bit resets the FLEXPWM_3 peripheral. Writing a ‘0’ to this bit releases the reset to the FLEXPWM_3 peripheral if the bit’s current value is ‘1’, otherwise it has no effect. 0 No forced reset of peripheral 1 Forced reset of peripheral
20 FLEXPWM_0_RST	FLEXPWM_0 reset Writing a ‘1’ to this bit resets the FLEXPWM_0 peripheral. Writing a ‘0’ to this bit releases the reset to the FLEXPWM_0 peripheral if the bit’s current value is ‘1’, otherwise it has no effect. 0 No forced reset of peripheral 1 Forced reset of peripheral

Table 950. RGM_PRST4 register field descriptions (continued)

Field	Description
21 FLEXPWM_1_RST	FLEXPWM_1 reset Writing a '1' to this bit resets the FLEXPWM_1 peripheral. Writing a '0' to this bit releases the reset to the FLEXPWM_1 peripheral if the bit's current value is '1', otherwise it has no effect. 0 No forced reset of peripheral 1 Forced reset of peripheral
23 CTU_1_RST	CTU_1 reset Writing a '1' to this bit resets the CTU_1 peripheral. Writing a '0' to this bit releases the reset to the CTU_1 peripheral if the bit's current value is '1', otherwise it has no effect. 0 No forced reset of peripheral 1 Forced reset of peripheral
28 ETIMER_2_RST	ETIMER_2 reset Writing a '1' to this bit resets the ETIMER_2 peripheral. Writing a '0' to this bit releases the reset to the ETIMER_2 peripheral if the bit's current value is '1', otherwise it has no effect. 0 No forced reset of peripheral 1 Forced reset of peripheral
29 ETIMER_3_RST	ETIMER_3 reset Writing a '1' to this bit resets the ETIMER_3 peripheral. Writing a '0' to this bit releases the reset to the ETIMER_3 peripheral if the bit's current value is '1', otherwise it has no effect. 0 No forced reset of peripheral 1 Forced reset of peripheral
30 ETIMER_0_RST	ETIMER_0 reset Writing a '1' to this bit resets the ETIMER_0 peripheral. Writing a '0' to this bit releases the reset to the ETIMER_0 peripheral if the bit's current value is '1', otherwise it has no effect. 0 No forced reset of peripheral 1 Forced reset of peripheral
31 ETIMER_1_RST	ETIMER_1 reset Writing a '1' to this bit resets the ETIMER_1 peripheral. Writing a '0' to this bit releases the reset to the ETIMER_1 peripheral if the bit's current value is '1', otherwise it has no effect. 0 No forced reset of peripheral 1 Forced reset of peripheral

47.3.2.17 Peripheral Reset Register 5 (RGM_PRST5)

This register provides individual resets for various peripherals. Supervisor and test modes allow for read/write access, whereas read access only is allowed in user mode. Please refer to *Table 951* for details.

Address: 0x624 Access: User read, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	CCCU_1_RST	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

Figure 989. Peripheral Reset Register 5 (RGM_PRST5)

Table 951. RGM_PRST5 register field descriptions

Field	Description
1 CCCU_1_RST	CCCU_1 reset Writing a ‘1’ to this bit resets the CCCU_1 peripheral. Writing a ‘0’ to this bit releases the reset to the CCCU_1 peripheral if the bit’s current value is ‘1’, otherwise it has no effect. 0 No forced reset of peripheral 1 Forced reset of peripheral

47.3.2.18 Peripheral Reset Register 6 (RGM_PRST6)

This register provides individual resets for various peripherals. Supervisor and test modes allow for read/write access, whereas read access only is allowed in user mode. Please refer to *Table 952* for details.

Address: 0x628 Access: User read, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	LINFlexD_1_RST	LINFlexD_3_RST	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	MCAN1_RAM_RST(1)	0	0	0	MCAN_1_RST	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 990. Peripheral Reset Register 6 (RGM_PRST6)

1. RGM is able to reset CAN RAM controller and not CAN RAM content. CAN RAM content cannot be accessed if corresponding CAN RAM controller is kept under reset by RGM bit.

Table 952. RGM_PRST6 register field descriptions

Field	Description
3 LINFlexD_1_RST	LINFlexD_1 reset Writing a ‘1’ to this bit resets the LINFlexD_1 peripheral. Writing a ‘0’ to this bit releases the reset to the LINFlexD_1 peripheral if the bit’s current value is ‘1’, otherwise it has no effect. 0 No forced reset of peripheral 1 Forced reset of peripheral
4 LINFlexD_3_RST	LINFlexD_3 reset Writing a ‘1’ to this bit resets the LINFlexD_3 peripheral. Writing a ‘0’ to this bit releases the reset to the LINFlexD_3 peripheral if the bit’s current value is ‘1’, otherwise it has no effect. 0 No forced reset of peripheral 1 Forced reset of peripheral

Table 952. RGM_PRST6 register field descriptions (continued)

Field	Description
21 MCAN1_RAM_RST	MCAN1_RAM reset Writing a '1' to this bit resets the MCAN1_RAM peripheral. Writing a '0' to this bit releases the reset to the MCAN1_RAM peripheral if the bit's current value is '1', otherwise it has no effect. 0 No forced reset of peripheral 1 Forced reset of peripheral
25 MCAN_1_RST	MCAN_1 reset Writing a '1' to this bit resets the MCAN_1 peripheral. Writing a '0' to this bit releases the reset to the MCAN_1 peripheral if the bit's current value is '1', otherwise it has no effect. 0 No forced reset of peripheral 1 Forced reset of peripheral

47.3.2.19 Peripheral Reset Register 7 (RGM_PRST7)

This register provides individual resets for various peripherals. Supervisor and test modes allow for read/write access, whereas read access only is allowed in user mode. Please refer to *Table 953* for details.

Address: 0x62C Access: User read, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	ADCSAR_dig_3_RST	ADCSAR_dig_4_RST	ADCSAR_dig_5_RST	0	ADCSAR_seq_3_RST	ADCSAR_seq_4_RST	ADCSAR_seq_5_RST	0	0	0	0	CTU_0_RST	0	0	0	ADCSAR_dig_b1_RST
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	ADCSAR_seq_b1_RST	0	0	0	0	0	0	SENT_1_RST	0	0	0	0	DSPI_2_RST	DSPI_3_RST	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 991. Peripheral Reset Register 7 (RGM_PRST7)

Table 953. RGM_PRST7 register field descriptions

Field	Description
0 ADCSAR_dig_3_RST	ADCSAR_dig_3 reset Writing a ‘1’ to this bit resets the ADCSAR_dig_3 peripheral. Writing a ‘0’ to this bit releases the reset to the ADCSAR_dig_3 peripheral if the bit’s current value is ‘1’, otherwise it has no effect. **Note:** This peripheral is coupled with RGM_PRST7.ADCSAR_seq_3_RST therefore they have to be identically set. 0 No forced reset of peripheral 1 Forced reset of peripheral
1 ADCSAR_dig_4_RST	ADCSAR_dig_4 reset Writing a ‘1’ to this bit resets the ADCSAR_dig_4 peripheral. Writing a ‘0’ to this bit releases the reset to the ADCSAR_dig_4 peripheral if the bit’s current value is ‘1’, otherwise it has no effect. **Note:** This peripheral is coupled with RGM_PRST7.ADCSAR_seq_4_RST therefore they have to be identically set. 0 No forced reset of peripheral 1 Forced reset of peripheral
2 ADCSAR_dig_5_RST	ADCSAR_dig_5 reset Writing a ‘1’ to this bit resets the ADCSAR_dig_5 peripheral. Writing a ‘0’ to this bit releases the reset to the ADCSAR_dig_5 peripheral if the bit’s current value is ‘1’, otherwise it has no effect. **Note:** This peripheral is coupled with RGM_PRST7.ADCSAR_seq_5_RST therefore they have to be identically set. 0 No forced reset of peripheral 1 Forced reset of peripheral
4 ADCSAR_seq_3_RST	ADCSAR_seq_3 reset Writing a ‘1’ to this bit resets the CTU peripheral. Writing a ‘0’ to this bit releases the reset to the ADCSAR_seq_3 peripheral if the bit’s current value is ‘1’, otherwise it has no effect. 0 No forced reset of peripheral 1 Forced reset of peripheral
5 ADCSAR_seq_4_RST	ADCSAR_seq_4 reset Writing a ‘1’ to this bit resets the ADCSAR_seq_4 peripheral. Writing a ‘0’ to this bit releases the reset to the ADCSAR_seq_4 peripheral if the bit’s current value is ‘1’, otherwise it has no effect. 0 No forced reset of peripheral 1 Forced reset of peripheral
6 ADCSAR_seq_5_RST	ADCSAR_seq_5 reset Writing a ‘1’ to this bit resets the ADCSAR_seq_5 peripheral. Writing a ‘0’ to this bit releases the reset to the ADCSAR_seq_5 peripheral if the bit’s current value is ‘1’, otherwise it has no effect. 0 No forced reset of peripheral 1 Forced reset of peripheral

Table 953. RGM_PRST7 register field descriptions (continued)

Field	Description
11 CTU_0_RST	CTU reset Writing a ‘1’ to this bit resets the CTU peripheral. Writing a ‘0’ to this bit releases the reset to the CTU peripheral if the bit’s current value is ‘1’, otherwise it has no effect. 0 No forced reset of peripheral 1 Forced reset of peripheral
15 ADCSAR_dig_b1_RST	ADCSAR_dig_b1 reset Writing a ‘1’ to this bit resets the ADCSAR_dig_b1 peripheral. Writing a ‘0’ to this bit releases the reset to the ADCSAR_dig_b1 peripheral if the bit’s current value is ‘1’, otherwise it has no effect. **Note:** This peripheral is coupled with RGM_PRST7.ADCSAR_seq_b1_RST therefore they have to be identically set. 0 No forced reset of peripheral 1 Forced reset of peripheral
16 ADCSAR_seq_b1_RST	ADCSAR_seq_b1 reset Writing a ‘1’ to this bit resets the ADCSAR_seq_b1 peripheral. Writing a ‘0’ to this bit releases the reset to the ADCSAR_seq_b1 peripheral if the bit’s current value is ‘1’, otherwise it has no effect. 0 No forced reset of peripheral 1 Forced reset of peripheral
23 SENT_1_RST	SENT_1 reset Writing a ‘1’ to this bit resets the SENT_1 peripheral. Writing a ‘0’ to this bit releases the reset to the SENT_1 peripheral if the bit’s current value is ‘1’, otherwise it has no effect. 0 No forced reset of peripheral 1 Forced reset of peripheral
28 DSPI_2_RST	DSPI_2 reset Writing a ‘1’ to this bit resets the DSPI_2 peripheral. Writing a ‘0’ to this bit releases the reset to the DSPI_2 peripheral if the bit’s current value is ‘1’, otherwise it has no effect. 0 No forced reset of peripheral 1 Forced reset of peripheral
29 DSPI_3_RST	DSPI_3 reset Writing a ‘1’ to this bit resets the DSPI_3 peripheral. Writing a ‘0’ to this bit releases the reset to the DSPI_3 peripheral if the bit’s current value is ‘1’, otherwise it has no effect. 0 No forced reset of peripheral 1 Forced reset of peripheral

Caution: Use the ***RGM_PRSTn*** registers with care. See *Section 47.4.7* for the proper sequence to prevent unexpected chip behavior.

47.3.2.20 Peripheral Reset Status Register 0 (RGM_PSTAT0)

This register provides the current reset status for various peripherals. It can be accessed in read only in all modes. See *Table 954* for details.

Address: 0x630 Access: User read, Supervisor read, Test read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	PIT_RTC_0_STAT	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	SIUL_STAT	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 992. Peripheral Reset Status Register 0 (RGM_PSTAT0)

Table 954. RGM_PSTAT0 register field descriptions

Field	Description
1 PIT_RTC_0_STAT	PIT_RTC_0 reset status 0 peripheral is not in reset 1 peripheral is in reset
16 SIUL_STAT	SIUL reset status 0 peripheral is not in reset 1 peripheral is in reset

47.3.2.21 Peripheral Reset Status Register 1 (RGM_PSTAT1)

This register provides the current reset status for various peripherals. It can be accessed in read only in all modes. See *Table 955* for details.

Address: 0x634 Access: User read, Supervisor read, Test read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	CCCU_STAT	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	FCCU_STAT	0	0	CRC_0_STAT	0	DMAMUX_0_STAT	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 993. Peripheral Reset Status Register 1 (RGM_PSTAT1)

Table 955. RGM_PSTAT1 register field descriptions

Field	Description
1 CCCU_STAT	CCCU reset status 0 peripheral is not in reset 1 peripheral is in reset
22 FCCU_STAT	FCCU reset status 0 peripheral is not in reset 1 peripheral is in reset
25 CRC_0_STAT	CRC_0 reset status 0 peripheral is not in reset 1 peripheral is in reset
27 DMAMUX_0_STAT	DMAMUX_0 reset status 0 peripheral is not in reset 1 peripheral is in reset

47.3.2.22 Peripheral Reset Status Register 2 (RGM_PSTAT2)

This register provides the current reset status for various peripherals. It can be accessed in read only in all modes. See *Table 956* for details.

Address: 0x638 Access: User read, Supervisor read, Test read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	LINFlexD_0_STAT	LINFlexD_2_STAT	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	MCAN_0_2_RAM_STAT	0	0	0	MCAN_0_STAT	MCAN_2_STAT	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 994. Peripheral Reset Status Register 2 (RGM_PSTAT2)

Table 956. RGM_PSTAT2 register field descriptions

Field	Description
3 LINFlexD_0_STAT	LINFlexD_0 reset status 0 peripheral is not in reset 1 peripheral is in reset
4 LINFlexD_2_STAT	LINFlexD_2 reset status 0 peripheral is not in reset 1 peripheral is in reset
21 MCAN_0_2_RAM_STAT	MCAN_0_2_RAM reset status 0 peripheral is not in reset 1 peripheral is in reset
25 MCAN_0_STAT	MCAN_0 reset status 0 peripheral is not in reset 1 peripheral is in reset
26 MCAN_2_STAT	MCAN_2 reset status 0 peripheral is not in reset 1 peripheral is in reset

47.3.2.23 Peripheral Reset Status Register 3 (RGM_PSTAT3)

This register provides the current reset status for various peripherals. It can be accessed in read only in all modes. See *Table 957* for details.

Address: 0x63C Access: User read, Supervisor read, Test read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	ADCSAR_dig_0_STAT	ADCSAR_dig_1_STAT	ADCSAR_dig_2_STAT	0	ADCSAR_seq_0_STAT	ADCSAR_seq_1_STAT	ADCSAR_seq_2_STAT	0	0	0	0	0	0	0	0	ADCSAR_dig_b_STAT
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	ADCSAR_seq_b_STAT	0	0	0	FLEXRAY_STAT	0	0	SENT_0_STAT	0	0	0	0	DSPI_0_STAT	DSPI_1_STAT	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 995. Peripheral Reset Status Register 3 (RGM_PSTAT3)

Table 957. RGM_PSTAT3 register field descriptions

Field	Description
0 ADCSAR_dig_0_STAT	ADCSAR_dig_0 reset status 0 peripheral is not in reset 1 peripheral is in reset
1 ADCSAR_dig_1_STAT	ADCSAR_dig_1 reset status 0 peripheral is not in reset 1 peripheral is in reset
2 ADCSAR_dig_2_STAT	ADCSAR_dig_2 reset status 0 peripheral is not in reset 1 peripheral is in reset
4 ADCSAR_seq_0_STAT	ADCSAR_seq_0 reset status 0 peripheral is not in reset 1 peripheral is in reset
5 ADCSAR_seq_1_STAT	ADCSAR_seq_1 reset status 0 peripheral is not in reset 1 peripheral is in reset
6 ADCSAR_seq_2_STAT	ADCSAR_seq_2 reset status 0 peripheral is not in reset 1 peripheral is in reset
15 ADCSAR_dig_b_STAT	ADCSAR_dig_b reset status 0 peripheral is not in reset 1 peripheral is in reset

Table 957. RGM_PSTAT3 register field descriptions (continued)

Field	Description
16 ADCSAR_seq_b_STAT	ADCSAR_seq_b reset status 0 peripheral is not in reset 1 peripheral is in reset
20 FLEXRAY_STAT	FLEXRAY reset status 0 peripheral is not in reset 1 peripheral is in reset
23 SENT_0_STAT	SENT_0 reset status 0 peripheral is not in reset 1 peripheral is in reset
28 DSPI_0_STAT	DSPI_0 reset status 0 peripheral is not in reset 1 peripheral is in reset
29 DSPI_1_STAT	DSPI_1 reset status 0 peripheral is not in reset 1 peripheral is in reset

47.3.2.24 Peripheral Reset Status Register 4 (RGM_PSTAT4)

This register provides the current reset status for various peripherals. It can be accessed in read only in all modes. See *Table 958* for details.

Address: 0x640 Access: User read, Supervisor read, Test read

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	PWM_SYNCHR_STAT	0	FLEXPWM_2_STAT	FLEXPWM_3_STAT	FLEXPWM_0_STAT	FLEXPWM_1_STAT	0	CTU_1_STAT	0	0	0	0	ETIMER_2_STAT	ETIMER_3_STAT	ETIMER_0_STAT	ETIMER_1_STAT
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 996. Peripheral Reset Status Register 4 (RGM_PSTAT4)

Table 958. RGM_PSTAT4 register field descriptions

Field	Description
16 PWM_SYNCHR_STAT	PWM_SYNCHR reset status 0 peripheral is not in reset 1 peripheral is in reset
18 FLEXPWM_2_STAT	FLEXPWM_2 reset status 0 peripheral is not in reset 1 peripheral is in reset
19 FLEXPWM_3_STAT	FLEXPWM_3 reset status 0 peripheral is not in reset 1 peripheral is in reset
20 FLEXPWM_0_STAT	FLEXPWM_0 reset status 0 peripheral is not in reset 1 peripheral is in reset
21 FLEXPWM_1_STAT	FLEXPWM_1 reset status 0 peripheral is not in reset 1 peripheral is in reset
23 CTU_1_STAT	CTU_1 reset status 0 peripheral is not in reset 1 peripheral is in reset
28 ETIMER_2_STAT	ETIMER_2 reset status 0 peripheral is not in reset 1 peripheral is in reset
29 ETIMER_3_STAT	ETIMER_3 reset status 0 peripheral is not in reset 1 peripheral is in reset
30 ETIMER_0_STAT	ETIMER_0 reset status 0 peripheral is not in reset 1 peripheral is in reset
31 ETIMER_1_STAT	ETIMER_1 reset status 0 peripheral is not in reset 1 peripheral is in reset

47.3.2.25 Peripheral Reset Status Register 5 (RGM_PSTAT5)

This register provides the current reset status for various peripherals. It can be accessed in read only in all modes. See *Table 959* for details.

Address: 0x644 Access: User read, Supervisor read, Test read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	CCCU_1_STAT	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

Figure 997. Peripheral Reset Status Register 5 (RGM_PSTAT5)

Table 959. RGM_PSTAT5 register field descriptions

Field	Description
1 CCCU_1_STAT	CCCU_1 reset status 0 peripheral is not in reset 1 peripheral is in reset

47.3.2.26 Peripheral Reset Status Register 6 (RGM_PSTAT6)

This register provides the current reset status for various peripherals. It can be accessed in read only in all modes. See *Table 960* for details.

Address: 0x648 Access: User read, Supervisor read, Test read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	LINFlexD_1_STAT	LINFlexD_3_STAT	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	MCAN1_RAM_STAT	0	0	0	MCAN_1_STAT	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 998. Peripheral Reset Status Register 6 (RGM_PSTAT6)

Table 960. RGM_PSTAT6) register field descriptions

Field	Description
3 LINFlexD_1_STAT	LINFlexD_1 reset status 0 peripheral is not in reset 1 peripheral is in reset
4 LINFlexD_3_STAT	LINFlexD_3 reset status 0 peripheral is not in reset 1 peripheral is in reset
21 MCAN1_RAM_STAT	MCAN1_RAM reset status 0 peripheral is not in reset 1 peripheral is in reset
25 MCAN_1_STAT	MCAN_1 reset status 0 peripheral is not in reset 1 peripheral is in reset

47.3.2.27 Peripheral Reset Status Register 7 (RGM_PSTAT7)

This register provides the current reset status for various peripherals. It can be accessed in read only in all modes. See *Table 961* for details.

Address: 0x64C Access: User read, Supervisor read, Test read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	ADCSAR_dig_3_STAT	ADCSAR_dig_4_STAT	ADCSAR_dig_5_STAT	0	ADCSAR_seq_3_STAT	ADCSAR_seq_4_STAT	ADCSAR_seq_5_STAT	0	0	0	0	CTU_0_STAT	0	0	0	ADCSAR_dig_b1_STAT
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	ADCSAR_seq_b1_STAT	0	0	0	0	0	0	SENT_1_STAT	0	0	0	0	DSPI_2_STAT	DSPI_3_STAT	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 999. Peripheral Reset Status Register 7 (RGM_PSTAT7)

Table 961. RGM_PSTAT7 register field descriptions

Field	Description
0 ADCSAR_dig_3_STAT	ADCSAR_dig_3 reset status 0 peripheral is not in reset 1 peripheral is in reset
1 ADCSAR_dig_4_STAT	ADCSAR_dig_4 reset status 0 peripheral is not in reset 1 peripheral is in reset
2 ADCSAR_dig_5_STAT	ADCSAR_dig_5 reset status 0 peripheral is not in reset 1 peripheral is in reset
4 ADCSAR_seq_3_STAT	ADCSAR_seq_3 reset status 0 peripheral is not in reset 1 peripheral is in reset
5 ADCSAR_seq_4_STAT	ADCSAR_seq_4 reset status 0 peripheral is not in reset 1 peripheral is in reset
6 ADCSAR_seq_5_STAT	ADCSAR_seq_5 reset status 0 peripheral is not in reset 1 peripheral is in reset
11 CTU_0_STAT	CTU_0 reset status 0 peripheral is not in reset 1 peripheral is in reset

Table 961. RGM_PSTAT7 register field descriptions (continued)

Field	Description
15 ADCSAR_dig_b1_STAT	ADCSAR_dig_b1 reset status 0 peripheral is not in reset 1 peripheral is in reset
16 ADCSAR_seq_b1_STAT	ADCSAR_seq_b1 reset status 0 peripheral is not in reset 1 peripheral is in reset
23 SENT_1_STAT	SENT_1 reset status 0 peripheral is not in reset 1 peripheral is in reset
28 DSPI_2_STAT	DSPI_2 reset status 0 peripheral is not in reset 1 peripheral is in reset
29 DSPI_3_STAT	DSPI_3 reset status 0 peripheral is not in reset 1 peripheral is in reset

47.4 Functional description

47.4.1 Reset state machine

The main role of the MC_RGM is the generation of the reset sequence which ensures that the correct parts of the chip are reset based on the reset source event. This is summarized in *Table 962*.

Table 962. MC_RGM reset implications

Source	What is reset	External reset assertion	Boot mode capture
Power-on reset	All	No	Yes
'Destructive' resets	All except some clock/reset management	No	Yes
External reset	All except some clock/reset management and debug	No	Yes
'Functional' resets	All except some clock/reset management, STCU, FCCU, and debug	No	Programmable (1)
Shortened 'functional' resets(2)	Flip-flops except some clock/reset management, STCU, FCCU, and debug	No	Programmable

1. the boot mode is captured if the external reset is asserted.
2. the short sequence is enabled via the RGM_FESS register.

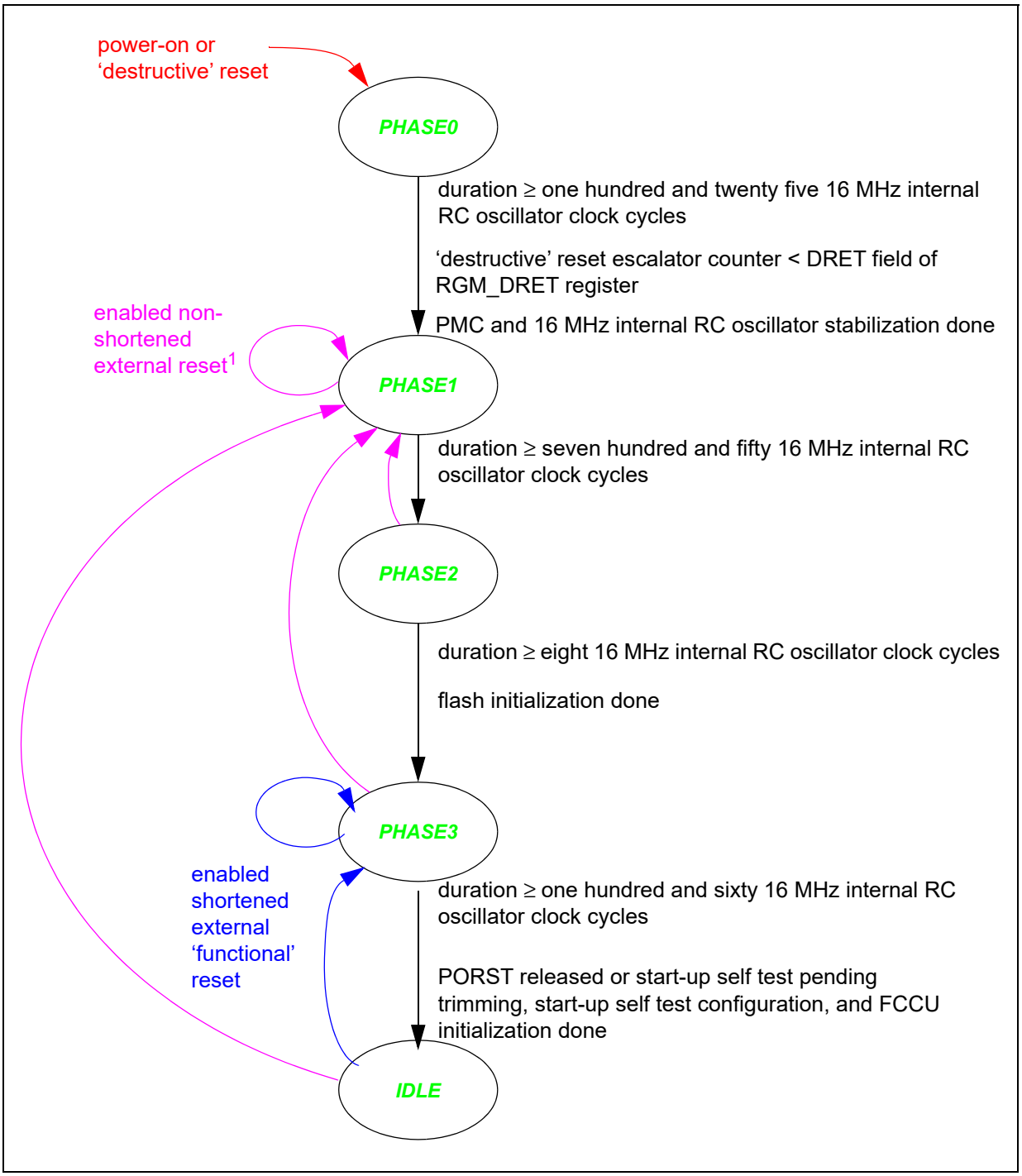
Note: JTAG logic has its own independent reset control and is not controlled by the MC_RGM in any way.

The reset sequence is comprised of five phases managed by a state machine, which ensures that all phases are correctly processed through waiting for a minimum duration and

until all processes that need to occur during that phase have been completed before proceeding to the next phase.

The state machine used to produce the reset sequence is shown in *Figure 1000*.

Figure 1000. MC_RGM State Machine



47.4.1.1 PHASE0 phase

This phase is entered immediately from any phase on a power-on or enabled ‘destructive’ reset event. The reset state machine exits PHASE0 and enters PHASE1 on verification of the following:

- all enabled ‘destructive’ resets have been processed
- all processes that need to be done in PHASE0 are completed
  - PMC and 16 MHz internal RC oscillator stabilization (Power management Controller)
- a minimum of one hundred and twenty five 16 MHz internal RC oscillator clock cycles have elapsed since power-up completion and the last enabled ‘destructive’ reset event
- the ‘destructive’ reset escalator counter has not reached the value in the DRET field of the RGM_DRET register

47.4.1.2 PHASE1 phase

This phase is entered either on exit from PHASE0 or immediately from PHASE2, PHASE3, or IDLE on a non-masked external or ‘functional’ reset event if it has not been configured to trigger a ‘short’ sequence. The reset state machine exits PHASE1 and enters PHASE2 on verification of the following:

- all enabled, non-shortened ‘functional’ resets have been processed
- a minimum of seven hundred and fifty 16 MHz internal RC oscillator clock cycles have elapsed since the last enabled external or non-shortened ‘functional’ reset event

47.4.1.3 PHASE2 phase

This phase is entered on exit from PHASE1. The reset state machine exits PHASE2 and enters PHASE3 on verification of the following:

- all processes that need to be done in PHASE2 are completed
  - flash initialization
- a minimum of eight 16 MHz internal RC oscillator clock cycles have elapsed since entering PHASE2

47.4.1.4 PHASE3 phase

This phase is entered either on exit from PHASE2 or immediately from IDLE on an enabled, shortened ‘functional’ reset event. The reset state machine exits PHASE3 and enters IDLE on verification of the following:

- all processes that need to be done in PHASE3 are completed
  - trimming, start-up self test configuration, and FCCU initialization
- a minimum of one hundred and sixty 16 MHz internal RC oscillator clock cycles have elapsed since the last enabled, shortened ‘functional’ reset event
- PORST is not asserted, except if a start-up self test is to be executed on exit of the current reset sequence

47.4.1.5 IDLE phase

This is the final phase and is entered on exit from PHASE3. When this phase is reached, the MC_RGM releases control of the system to the platform and waits for new reset events that can trigger a reset sequence.

47.4.2 ‘Destructive’ resets

A ‘destructive’ reset indicates that an event has occurred after which critical register or memory content can no longer be guaranteed. Exceptions to the latter are the software ‘destructive’ reset and the external power-on reset which do allow the memory content to be preserved if the chip is configured to not execute a self test on power-on, ‘destructive’, and external resets.

The status flag associated with a given ‘destructive’ reset event (RGM_DES.F_<destructive reset> bit) is set when the ‘destructive’ reset is asserted and the power-on reset is not asserted. It is possible for multiple status bits to be set simultaneously, and it is software’s responsibility to determine which reset source is the most critical for the application.

A given ‘destructive’ reset can be optionally disabled by writing bit RGM_DERD.D_<destructive reset>.

Note: The RGM_DERD register can be written only once between two ‘destructive’ reset events.

The chip’s low-voltage detector threshold ensures that, when 'destructive' voltage out of range (LVD or HVD, or both) is enabled, the supply is sufficient to have the ‘destructive’ event correctly propagated through the digital logic. Therefore, if a given ‘destructive’ reset is enabled, the MC_RGM ensures that the associated reset event will be correctly triggered to the full system. However, if the given ‘destructive’ reset is disabled and the voltage goes below the digital functional threshold, functionality can no longer be ensured, and the reset may or may not be asserted.

An enabled ‘destructive’ reset will trigger a reset sequence starting from the beginning of PHASE0.

47.4.2.1 External power-on reset—PORST

The external power-on reset pin PORST is a special case ‘destructive’ reset. Detection of a falling edge on PORST will start the reset sequence.

In addition, the reset sequence is not exited into application until PORST is no longer driven low from off-chip. In the case the chip is configured to execute a self-test on power-on, ‘destructive’, and external resets, this is the exit of PHASE3 after self-test execution has completed.

47.4.3 ‘Functional’ resets

A ‘functional’ reset indicates that an event has occurred after which it can be guaranteed that critical register and memory content is still intact.

The status flag associated with a given ‘functional’ reset event (RGM_FES.F_<functional reset> bit) is set when the ‘functional’ reset is asserted and the power-on reset is not asserted. It is possible for multiple status bits to be set simultaneously, and it is software’s responsibility to determine which reset source is the most critical for the application.

A given ‘functional’ reset can be optionally disabled by software writing bit RGM_FERD.D_<functional reset>.

Note: The RGM_FERD register can be written only once between two power-on reset events.

An enabled ‘functional’ reset will normally trigger a reset sequence starting from the beginning of PHASE1. Nevertheless, the RGM_FESS register enables the further

configuring of the reset sequence triggered by a ‘functional’ reset. When RGM_FESS.SS_<functional reset> is set, the associated ‘functional’ reset will trigger a reset sequence starting directly from the beginning of PHASE3, skipping PHASE1 and PHASE2. This can be useful especially in case a ‘functional’ reset should not reset the flash module.

47.4.4 Alternate event generation

The MC_RGM provides alternative events to be generated on reset source assertion. When a reset source is asserted, the MC_RGM normally enters the reset sequence. Alternatively, it is possible for some reset source events to be converted from a reset to either a SAFE mode request issued to the MC_ME or to an interrupt request issued to the core.

Alternate event selection for a given reset source is made via the RGM_DERD/FERD and RGM_DEAR/FEAR registers as shown in *Table 963*.

Table 963. MC_RGM Alternate Event Selection

RGM_DERD/FERD bit value	RGM_DEAR/FEAR bit value	Generated event
0	X	Reset
1	0	SAFE mode request
1	1	Interrupt request

The alternate event is cleared by deasserting the source of the request (that is at the reset source that caused the alternate request) and also clearing the appropriate RGM_DES/FES status bit.

Note: SAFE mode requests are generated regardless of whether the system clock is running. Interrupt requests are generated with the system clock, so if the system clock is disabled at the time of the event, the interrupt request is not asserted until the system clock is re-enabled.

If a masked ‘destructive’ reset event which is configured to generate a SAFE mode/interrupt request occurs during PHASE0, it is ignored and the MC_RGM does not send any safe mode/interrupt request to the MC_ME.

If a masked ‘functional’ reset event which is configured to generate a SAFE mode/interrupt request occurs during PHASE1, it is ignored, and the MC_RGM will not send any safe mode/interrupt request to the MC_ME.

47.4.5 ‘Functional’ reset escalation

‘Functional’ reset escalation can be used to generate a ‘destructive’ reset if a number of ‘functional’ or external resets has occurred between software writes to the RGM_FRET register. This function is enabled by writing a non-zero value to the FRET field of this register.

Once ‘functional’ reset escalation has been enabled, the MC_RGM increases a counter on each ‘functional’ or external reset which causes a reset sequence to be initiated (such as, entrance into PHASE1 or PHASE3 from the IDLE phase). This counter is cleared on a write of any value to the RGM_FRET register and on any power-on or ‘destructive’ reset. If the counter reaches the value in the RGM_FRET register’s FRET field, the MC_RGM starts a reset sequence at PHASE0 and asserts the F_EDR bit in the RGM_DES register.

47.4.6 'Destructive' reset escalation

'Destructive' reset escalation can be used to keep the chip in the reset state until a new power-on triggers a reset sequence if a number of 'destructive' resets has occurred between software writes to the RGM_DRET register. This function is enabled by writing a non-zero value to the DRET field of this register.

Once 'destructive' reset escalation has been enabled, the MC_RGM increases a counter on each 'destructive' reset which causes a reset sequence to be initiated (such as, entrance into PHASE0 from the IDLE phase) or an ongoing reset sequence to restart (such as, entrance into PHASE0 from PHASE1, PHASE2, or PHASE3). This counter is cleared on a write of any value to the RGM_DRET register and on any power-on reset. If the counter reaches the value in the RGM_DRET register's DRET field, the MC_RGM enters reset PHASE0 and stays there until the next power-on reset occurs.

47.4.7 Individual peripheral resets

A peripheral may be reset individually without resetting the rest of the chip by setting the appropriate bit in the RGM_PRSTn registers. However, in order to prevent unexpected behavior by the chip, the following sequence must be observed by software.

1. Disable the peripheral that is to be reset via the MC_ME (see the MC_ME chapter for details).
2. Wait for the mode change to complete.
3. Set the bit corresponding to the peripheral in the RGM_PRSTn registers.
4. Wait at least two peripheral clock periods.
5. Clear the bit corresponding to the peripheral in the RGM_PRSTn registers.

The peripheral can be kept in reset indefinitely by not executing step 5. After the completion of step 5, the peripheral can be enabled again via the MC_ME.

48 Boot Assist Flash (BAF)

48.1 Introduction

The BAF code is programmed by ST. The two main tasks of BAF are to provide a serial bootloader feature and to search for the application entry point.
For details regarding how the BAF code is executed, refer to chapter *Chapter 8: Reset and boot*.

The BAF is executed via the IRCOSC as clock source.

The MCU is booted through a collaboration of several blocks, hardware and firmware. The first boot phases are performed by a state machine inside the System Status and Configuration Module (SSCM). Once completed, the SSCM sends a reset vector to the HW boot core of the device pointing into the Boot Assist Flash (BAF).

The BAF code then checks the life cycle of the device. If it is FAIL_ANALYSIS, the BAF enters a loop in which it services the watchdog. Otherwise, it searches for a boot header and boots the application code in internal flash memory.

If no boot header is found in internal flash memory, it downloads application code serially using the LINFlexD module or the M_CAN module. The package pins used by LINFlexD are the same pins used by the M_CAN module. Hence, the external PHY can be either LIN or CAN (see *Section 48.3.6* for configuration details).

48.2 Memory map

For the BAF location in flash, refer to the Embedded memories chapter. The BAF block base address is 0x0040_4000. It is one time programmable (OTP) and is programmed during factory test. The address space and memory used by the BAF code are shown in *Table 964*.

Note: The BAF Flash block is assigned as OTP and it is protected from programming by a bit in the PASS LOCK0 registers; but the BAF Flash block can further be protected from programming by an external hardware pin. If the Hardware Interlock feature is enabled, while this pin is logic '0' the BAF Flash block can not be programmed.

Table 964. BAF memory organization

BAF Component	Address
BAF image header	0x0040_4000
BAF code entry point	0x0040_4100
BAF data area and stack	5000_0000h–0x5000_03FFh

48.2.1 BAF image header

BAF image contains a 256-byte header starting from address 0x0040_4000. The BAF header is explained in *Table 965*.

Table 965. BAF image header

Address offset	Size (bits)	Field description
0x0000	32	SSCM boot record tag
0x0004	32	BAF start address
0x0008	32	BAF version number
0x000C–0x00FF	Reserved	

48.2.1.1 BAF image version

The BAF version is a 32-bit field in the image header starting at address 0x0040_4000. The 32-bit field is explained in *Table 966*.

Table 966. BAF image version

Field	Description
0:7	Major Number: This field contains the major version number for the BAF image
8:15	Minor Number: This field contains the minor version number for the BAF image
16:31	Reserved

48.3 Functional description

A functional description of the BAF modes is contained in the following sections.

48.3.1 Boot modes

Depending on the state and the content of its flash memory, the device can enter one of these different boot modes.

- Application code execution

  If the internal flash contains application code with a valid boot header, the application is executed.

- Serial boot

  If no valid boot header is found and the current life cycle phase of the device is less than Infield, an attempt is made to download the application by a serial protocol using the LINFlexD module or M_CAN module. The downloaded code is then executed by the boot CPU.

- Static mode

  The device enters power safe mode and waits for an external reset.

48.3.2 Device initialization

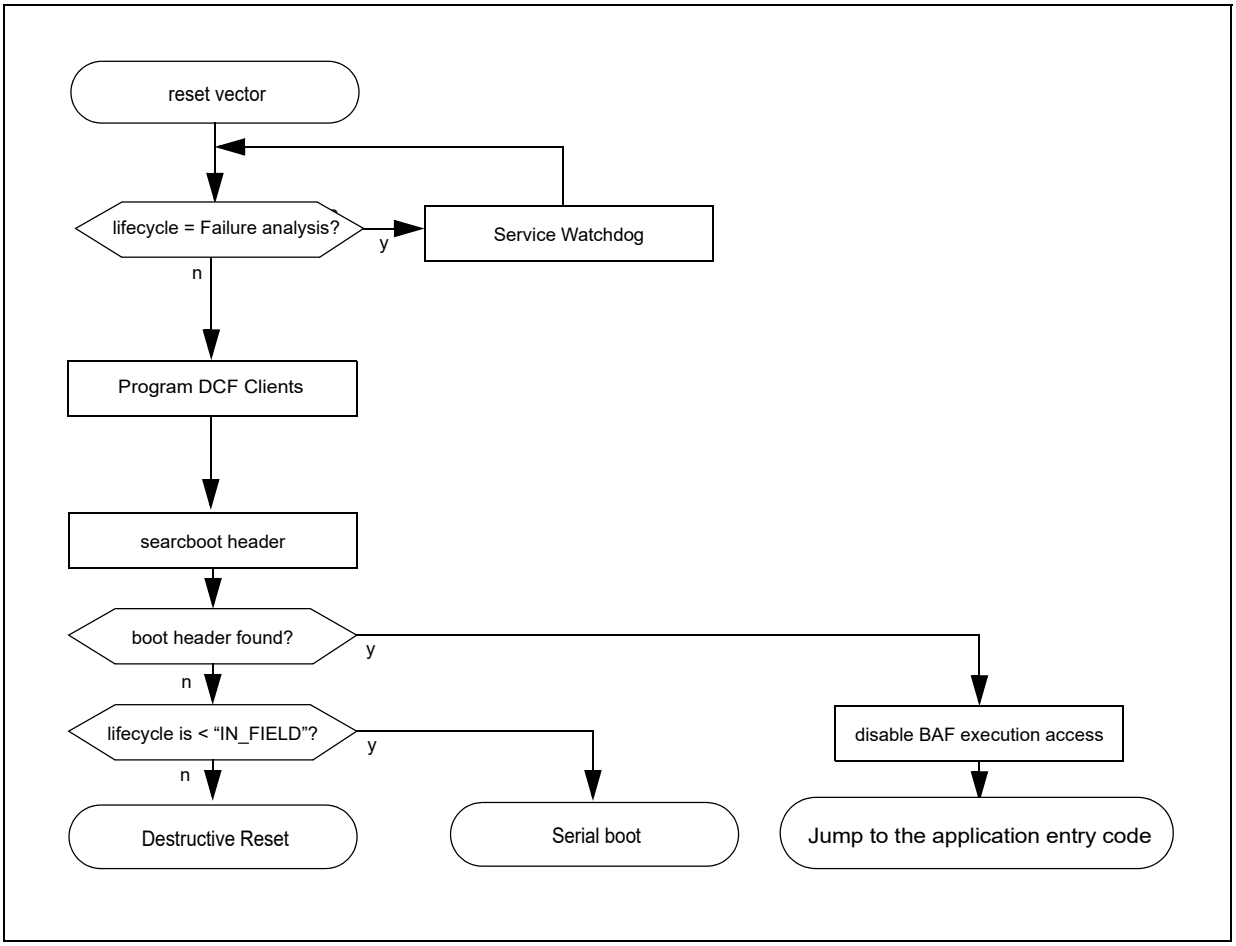
The BAF performs a minimal initialization of core registers and clocks required to enter the above described boot modes, then attempts to interact in the following way:

- BAF sets up the exception vector table to catch machine check exceptions, which may occur if there are any uncorrectable ECC errors. In the event of an uncorrectable ECC error, the BAF issues a destructive reset from the machine check exception handler if the exception is due to the reading of instructions from the BAF region. If an exception occurs during reading of the Boot header, the BAF ignores the ECC error and jumps to next boot header. If an ECC error occurs during reading of System RAM, the BAF will still issue a destructive reset. BAF programs the Mode Entry module's MC_ME_MCTL[TARGET_MODE] to issue a destructive reset.
- BAF uses memory of the system RAM as in *Table 964: BAF memory organization* for its stack, data and any code that executes from volatile memory. During the BAF start-up procedure, it initializes this section of memory so that there is no ECC error when BAF accesses this memory area.
- The memory area of the BAF is protected from execution for functional safety considerations. This is achieved by disabling BAF execution in PFLASH control register3 (PFLASHC_PFCR3[BAF_DIS]). Finally a mode change (from DRUN to DRUN again) is requested from Mode Entry module to activate the new core configuration. At this point the BAF has finished its operation.

48.3.3 Flow of control

Figure 1001 shows the overall flow of control in the BAF. The individual steps are explained in the following sections.

Figure 1001. Flow of control



48.3.4 Initialization of BAF DCF clients

48.3.4.1 ‘Soft’ DCF clients

DCF clients are 32-bit hardware registers that are written by the SSCM during reset with data that is stored in DCF record format in UTEST region of flash memory. These DCF records are used to configure module registers, and select device operating modes.

The BAF extends the DCF record concept by implementing soft DCF clients. The 64-bit DCF records include:

- 32-bits of data
- 15-bits chip select field (selects the module which implements the DCF clients)
- 15-bits address field (selects a specific DCF client within the selected module)
- Parity bit (not used with UTEST DCF records)
- Stop bit (to indicate the end of DCF record has been reached)

As part of its execution flow, BAF searches through the list of DCF records in UTEST flash to determine if any of the records are to be processed by BAF.

The search starts at one of two addresses: either the start address of DCF records in UTEST (DCF Start Record), or the address specified in UTEST (Soft DCF Record Start

Address). If either of the two least significant bits of the 32-bit address stored at the Soft DCF Record Start Address is set, the default start address at the DCF Start Record is used. If the two least significant bits are clear, this 32-bit value is assumed to be the start address of the search.

The default setting of the two least significant bits is 11b (erased value of flash memory is 0xFFFF_FFFF). A user may intentionally program a dummy address with the least two significant bits set in order to disable the search for an alternate address (as the UTEST block is OTP). Valid search start address must start on a 4-byte boundary.

The DCF record search finishes when a DCF record is read with a STOP bit set. The STOP bit may be set because that memory location is not programmed yet (so the list can be added to) or it can be intentionally programmed STOP record to indicate that no more records are added to the list. BAF does not execute the DCF record with STOP bit set. BAF parses the STOP bit before parsing the rest of the records.

BAF uses Chip Select 14 (CS14) for BAF DCF clients. For records with CS14 asserted, the BAF uses the address field (address[14:0]) in DCF records to determine how to process the 32-bits of data. The soft DCF clients programmed by BAF is listed in *Table 967*.

A group of four Soft DCF clients is used to provide a mechanism to write configuration information to address locations. The first DCF client receive 32-bit address information. The next three DCF Clients receive the data that is to be written to the address contained in the first client. The three data clients allow 32-, 16- and 8-bit data to be written to the address in the first client. These four clients can be accessed repeatedly to write whole string of data to memory.

Another soft DCF client as stated in *Table 967* provides a simple callback feature. The soft DCF record data contains the start address of a subroutine that the BAF calls before it starts searching for the host boot header and exits to customer application code. The callback routine can reside within the BAF or UTEST blocks only. The callback DCF client can be accessed repeatedly in order to execute a callback sequence.

The serial boot DCF client provides a mechanism using which application can inform BAF to perform serial boot or boot from internal flash. Details can be viewed in *Section 48.3.4.5*.

Table 967. BAF DCF client list

Address (hex)	Name	Label	Description
0x0000–0x01FF	Reserved		
0x0200	Address	ADDR	The address to which DATAX is written
0x0201	32-bit Data	DATA32	The 32-bit data that is written to ADDR
0x0202	16-bit Data	DATA16	The 16-bit data that is written to ADDR
0x0203	8-bit Data	DATA8	The 8-bit data that is written to ADDR
0x0204–0x020F	Reserved		
0x0210	Callback Address	CALLBACK	The address of a callback function
0x0211	Serial boot Callback Address	SER_BOOT_CBACK	The address of serial boot callback function
0x0212–0x0213	Reserved		Reserved for future use.
0x0214	Reserved		Reserved for future use.
0x0215–0x7FFF	Reserved		Reserved for future use.

Note: BAF boots with a watchdog timeout window set to 16ms. Parsing more and more DCF Records may increase boot time. But it should be taken care that number of DCF records should not be very high as it can cause watchdog window timeout.

48.3.4.2 BAF DCF client - ADDRESS

BAF parses the DCF records. If BAF finds a DCF record with CS14 set and Address field as 0x0200 while parsing, BAF programs this DCF client. This client is written with a 32-bit address from the Data section of DCF record. This 32-bit address is used in conjunction with any one of the DATA32, DATA16 or DATA8 DCF Data clients. BAF does not immediately use the ADDRESS information, but it is stored and used later in conjunction with a DATAX DCF client (DATA32, DATA16 or DATA8). Then BAF writes the data to the address in memory specified by this ADDRESS DCF client.

The user can create a series of data writes to memory by programming a series of DCF records consisting of pairs of records. The first DCF record of the pair writes to ADDR to provide the address. The second DCF record of the pair writes to the DATA client (DATAX) to provide the data. Subsequent pairs of DCF records can be programmed to write to more memory locations. The number of pairs of DCF records that can be programmed is only limited by the amount UTEST space set aside for DCF records.

DCF Client Address 0x0200

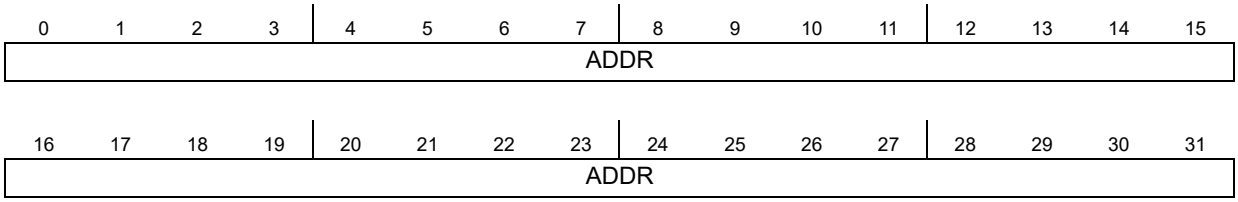


Figure 1002. BAF DCF ADDR client configuration

Table 968. BAF DCF ADDR client description

Field	Description
0:31 ADDR	32-Bit Address DCF client This ADDRESS value is used in conjunction with a DATAX value. The BAF writes DATAX value to ADDRESS value in the memory space of the MCU.

48.3.4.3 BAF DCF client - DATAX

Caution: While using DATAX DCF records, it should be taken in account that BAF does not take care of ECC scheme and can generate ECC, if write is performed in memory locations which are protected by ECC and are not pre-initialized as required.

48.3.4.3.1 DATA32 – 32-bit data DCF client

BAF parses the DCF records. If BAF finds a record with CS14 set and address field set as 0x0201 while parsing, BAF programs this 32-bit DCF client. BAF takes the memory address provided by the ADDR client, masks the least significant 2 bits to ‘0’ (Only 32-bit aligned writes are supported) and writes this 32-bit data to the address.

DCF Client Address 0x0201

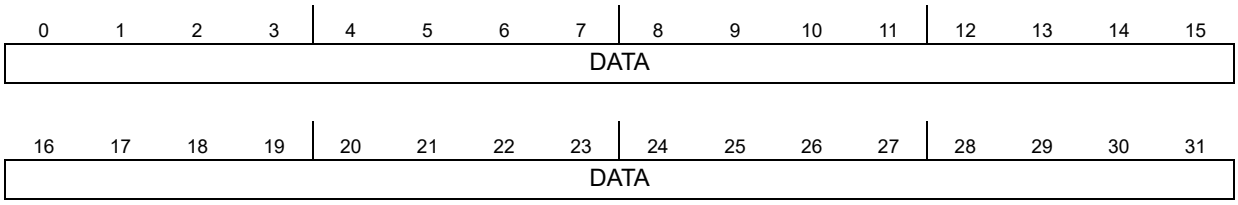


Figure 1003. DATA32 DCF client configuration

Table 969. DATA32 DCF client configuration

Field	Description
0:31 DATA	32-Bit DCF data

48.3.4.3.2 DATA16 – 16-bit data DCF client

BAF parses the DCF records. If BAF finds a record with CS14 set and address field set as 0x0202 while parsing, BAF programs this 16-bit DCF client. BAF takes the memory address provided by the ADDR client, masks the least significant bits to '0' (Only 16-bit aligned writes are supported) and writes least significant 16-bit data to the address. Data in the 16 most significant bits are ignored.

DCF Client Address 0x0202

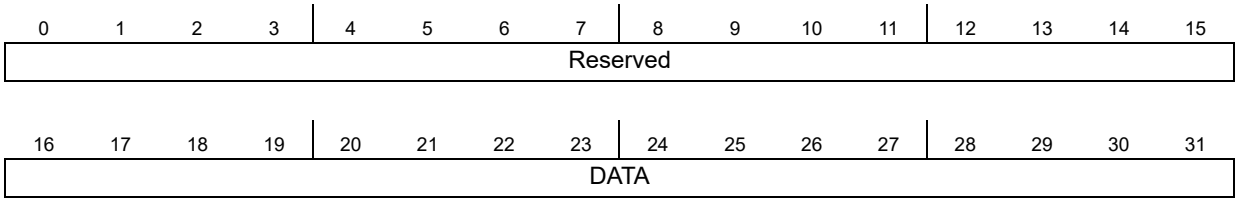


Figure 1004. DATA16 DCF Client Configuration

Table 970. DATA16 DCF client configuration

Field	Description
16:31 DATA	16-Bit DCF data

48.3.4.3.3 DATA8 – 8-bit DCF client

BAF parses the DCF records. If BAF finds a record with CS14 set and address field set as 0x0203 while parsing, BAF programs this 8-bit DCF client. BAF takes the memory address provided by the ADDR client, and writes least significant 8-bit data to the address. Data in the 24 most significant bits are ignored.

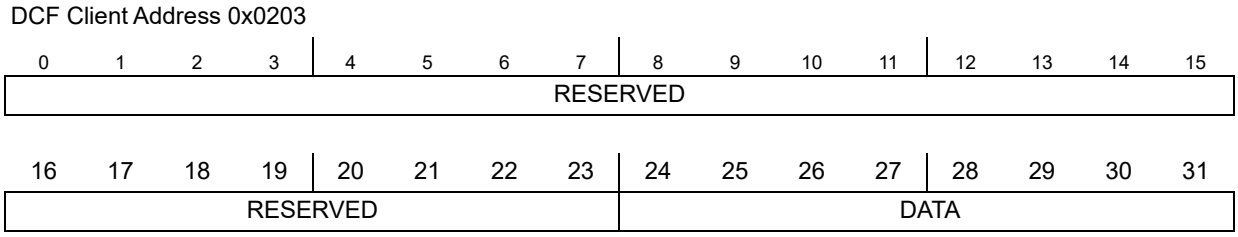


Figure 1005. DATA8 DCF Client Configuration

Table 971. DATA8 DCF client configuration

Field	Description
24:31 DATA	8-Bit DCF data

48.3.4.4 CALLBACK – BAF callback DCF client

BAF parses the DCF records. If BAF finds a record with CS14 set and address field as 0x0210 while parsing, BAF programs this callback DCF client. This client is written with a 32-bit start address of a function which is invoked by BAF. The BAF code checks the start address is within the UTEST or within the BAF flash memory blocks, and then passes program flow to this address. If the address is found to be outside the UTEST and BAF flash memory blocks, no function call is executed. The callback is aborted and BAF moves on to the next DCF records. The function that is called is expected to comply with Embedded Application Binary Interface (EABI) for PowerPC found in Power Architecture web site and to terminate in an orderly manner. An incorrectly written function causes unpredictable operations in BAF. No parameters are passed to this function and no return value is expected.

Caution: Excessively long callback functions could cause the watchdog timer to time-out and cause a reset.

BAF executes the callback function as the DCF records are parsed. Therefore, several callback functions maybe executed by writing several DCF records that write to the callback DCF clients. The callback functions are executed in the same order as the DCF records.

48.3.4.5 SER_BOOT_CBACK – BAF serial boot DCF client

BAF parses the DCF records. If BAF finds a record with CS14 set and address field as 0x0211 while parsing, BAF programs this serial boot DCF client. This DCF client is written with a 32 bit start address of function which is invoked by BAF. The BAF code checks the start address is within UTEST or within the BAF flash memory block, and then passes program flow to this address. If the address is found to be outside the UTEST and BAF flash memory block, no function call is executed. The DCF client execution is aborted and BAF moves on to the next DCF record. The function that is called is expected to comply with Embedded Application Binary Interface (EABI) for PowerPC found in ower Architecture web site and to terminate in orderly manner. An incorrectly written function causes unpredictable operations in BAF. No parameters are passed to this function.

Warning: No parameters are passed to the function invoked by the BAF.Excessively long callback functions could cause the watchdog timer to time-out and cause a reset.

The serial boot callback function returns a char value. The returned value informs BAF if serial boot or normal internal flash boot should be executed (see *Table 972* for details regarding return value).

Table 972. Serial boot callback function return value definition

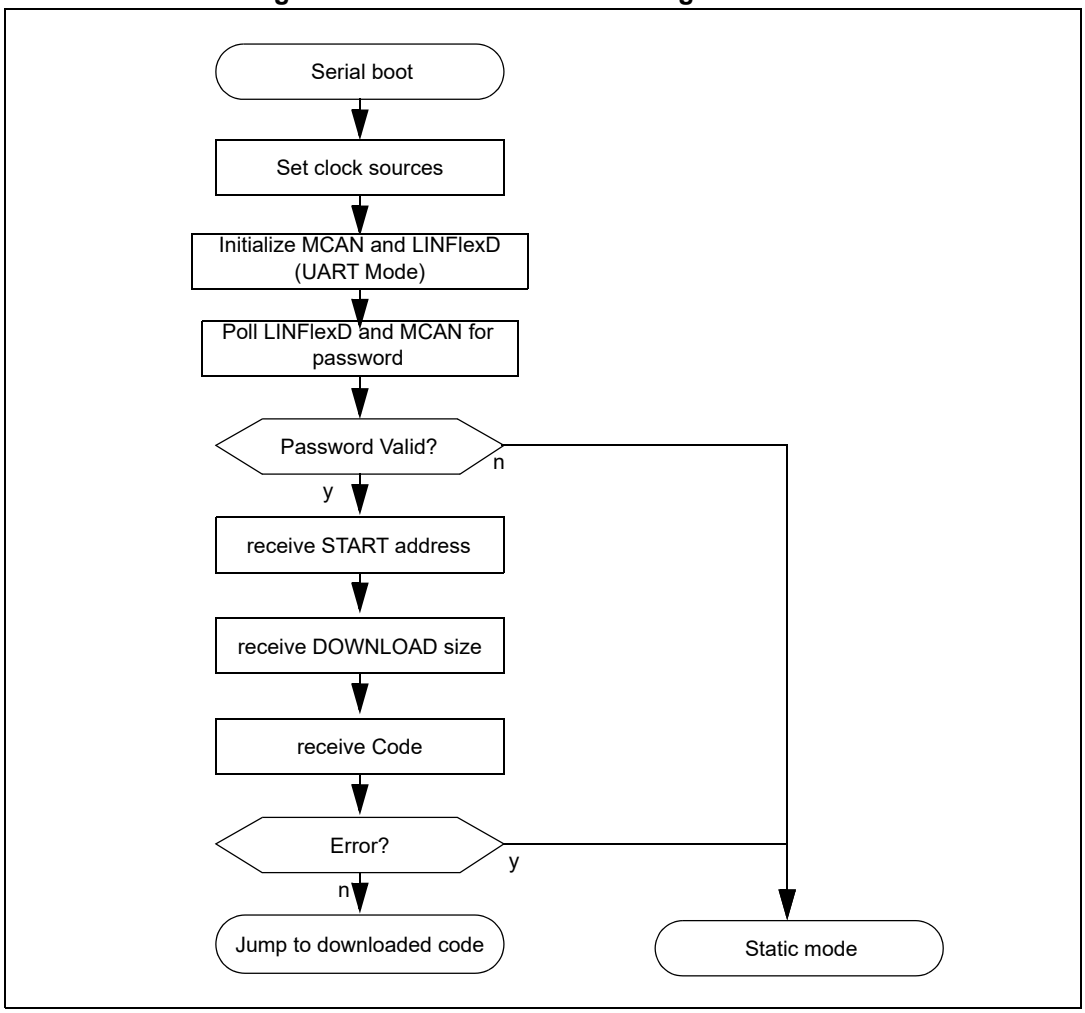
Return values	Description
0x00	Boot from internal flash using standard search for boot record (see 'Search boot header and boot options' in *Chapter 7: Device configuration*).
0x01	Serial boot using both M_CAN and LINFlexD (in UART mode) (see 'Search boot header and boot options' in *Chapter 7: Device configuration*). **Note:** Serial boot is conditioned by life cycle as described in *Figure 1001*

48.3.5 Optionally perform a serial boot

If no boot header is found in one of the locations mentioned above, then BAF detemines the Life Cycle statusof the device. If the the Life Cycle is lower than In Field, it attempts a serial boot. Otherwise, the boot has failed and the BAF issues a destructive reset.

f the customer selects serial boot using serial boot using serial boot DCF client, then BAF determines the Life Cycle status of the device. If the Life Cycle is lower than Failure Analysis, it attempts a serial boot.

Figure 1006. Flow of control during serial boot



The clock source in the MC_CGM is configured and the PADs are set up for LINFlexD and M_CAN signals, then the code is downloaded using LINFlexD or M_CAN in Serial Boot mode. In the absence of errors causing the device to be placed in Static mode, the device state is restored to resemble the out-of-reset state and the downloaded code is executed.

Serial boot allows the download of application to internal SRAM via LINFlexD or M_CAN interface. After downloading the application image to SRAM, the BAF jumps to the start address provided as part of download protocol mentioned below and executes the downloaded application.

At the start of the process both LINFlexD and M_CAN are configured in passive listening mode, connected to the same pair of pads. As soon as either interface receives a valid message, the receiving interface is configured in full Tx and Rx mode. The non receiving interface is switched back to reset state and doesn't participate further in serial boot.

If a CAN message of 8 bytes with ID 0x11 is received on the M_CAN controller first, the BAF program

- Disables LINFlexD.
- Configures LINFlexD RXD input mux back to reset state.
- Transitions to CAN serial download protocol routine.

If a CAN message not containing 8 bytes or NOT with ID 0x11 is received by M_CAN controller or a M_CAN controller generates a CAN error, it is probable that the tool is sending a LINFlexD frame. Under such circumstances BAF program

- Ignores the CAN message, if a message arrived.
- Clear any pending CAN error.
- Continue monitoring for valid CAN messages.

If any byte from LINFlexD is received first, the BAF program:

- Checks if the first byte is 0xFE of the password. If not continue polling. The data may be a valid CAN message.
- If it is first byte of the password (0xFE) then disable the M_CAN controller and restore to its reset state.
- Configures the M_CAN pads to their reset state.
- Transition to LINFlexD serial boot (UART mode).

48.3.5.1 Serial boot mode protocol

From a high-level perspective, the download protocol over M_CAN and LIN interfaces follows steps below:

1. Host transmits the 64-bit password to MCU.
2. Host transmits start address, size of downloaded code in bytes, NO_ECHO bit and VLE bit to MCU.

   NO_ECHO bit signifies if actual download data to be echoed or not. If NO_ECHO bit is set, then data will not be echoed otherwise if NO_ECHO bit is clear, then data will be echoed.

Note: Since this device supports only VLE code and no Book E code, the VLE bit is used only for backward compatibility.

3. Host transmits a sequence of data bytes that are to be executed.
4. The boot CPU executes code from start address provided in step 2.

Each step must be completed before the next step starts.

The exact protocol between LINFlexD and M_CAN are slightly different.

48.3.5.2 LINFlexD download protocol

The communication is performed in half-duplex mode, any transmission from the host is followed by the MCU transmission.

1. The host sends data to the MCU and starts waiting.
2. The MCU echoes the received 8-byte of password, 4-byte of load address and 4-byte of VLE-bit, NO_ECHO bit and 30-bit of download size to the host. The MCU echoes actual download data depending upon NO_ECHO bit.
3. The host verifies whether the echo is correct.
   – If the data is correct, the host can send the next byte of data.
   – If data is not correct, it is assumed the MCU has not correctly received the data and the only practical option is for the Host to reset the MCU and start again.

A more detailed description of these steps follows.

All multi-byte data structures are sent with MSB first.

The tool sends data in the following order:

1. 8 bytes that represent the password.
2. 4 bytes that represent the start address where the code is to be downloaded.
3. 4 bytes that represent the number of bytes to be downloaded, NO_ECHO bit and the VLE bit.
4. *x* bytes of code that is downloaded to SRAM and executed (*x* must match the value specified in step 3.).

48.3.5.3 M_CAN download protocol

The M_CAN download protocol uses very similar structure to the LINFlexD protocol, but it makes use of the message ID's provided by M_CAN.

The tool sends data packed into standard CAN message in the following manner:

1. A message with 0x11 as the ID and an 8-byte length to send the password. The MCU transmits the same message, but with an ID of 0x1.
2. A message with 0x12 as the ID and and 8-byte length to send the start address, size of the code and VLE mode bit.The MCU transmits back the same data but with an ID of 0x2.
3. A message with 0x13 as the ID is used to send data that is to be downloaded to SRAM. The MCU transmit the same data with an ID of 0x3.

48.3.5.4 Download 64-bit password and password check

The first 64 bits received represent the password. The MCU only supports public passwords. The received password is always compared with the public password 0xFEED_FACE_CAFE_BEEF.

- If the correct password is received, BAF continues its task.
- If an incorrect password is supplied, BAF puts the device into static mode(low power safe mode).
- If 64 bits of password data is not received after approximately 30s, BAF causes a destructive reset of the MCU.

48.3.5.5 Download start address, VLE bit, NO_ECHO bit and code size

After the password, the next 8 bytes received by the MCU contains a 32-bit start address, the VLE mode bit, NO_ECHO bit and 30-bit code length as shown in *Figure 1007* and *Figure 1008*.

The VLE bit (Variable Length Encoding) is used to indicate for which instruction set the downloaded code is compiled. The UART serial boot protocol supports download of VLE Code (VLE = 1) and Book E Code (VLE = 0).

The MCU only supports VLE code. If the BAF detects that the host is sending Book E code, the BAF puts the device into static mode.

The NO_ECHO bit is used to indicate if download data should be echoed by the MCU. If the NO_ECHO bit is set (for example, NO_ECHO = 1), then only 8 bytes of password, 4 bytes of download address, and 4 bytes of download size are echoed, but download data is not echoed. If the NO_ECHO bit is clear, then every byte sent to the MCU must be echoed.

The start address defines where the received data is stored and where the MCU branches after the download is finished. The three LSB bits of the start address are ignored by BAF, such that the load address is 64-bit double-word aligned.

The length determines how many data bytes has to be loaded.

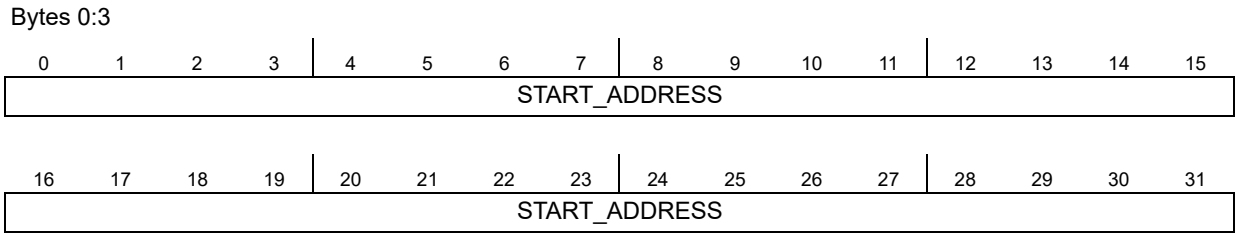


Figure 1007. Start address

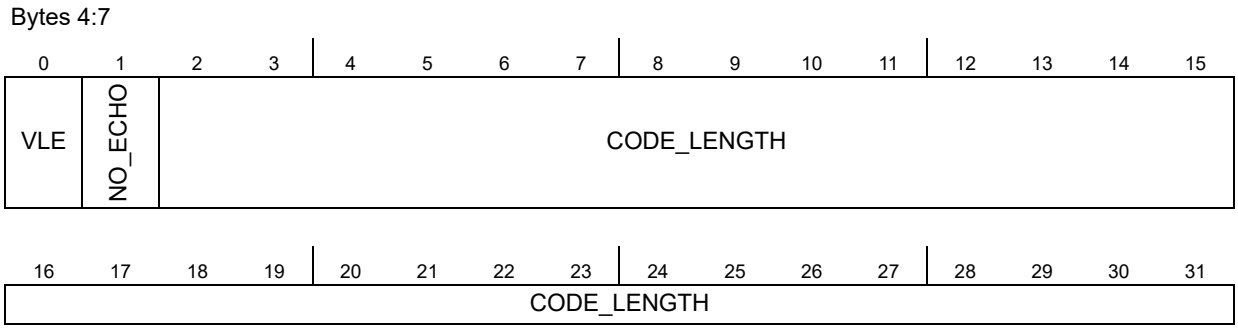


Figure 1008. VLE bit, NO_ECHO bit and code size

48.3.5.6 Download data

Each byte of data received is downloaded into device SRAM starting from the address specified in the previous protocol step. It is not verified whether the provided address is a valid address in SRAM or is writable.

The address increments until a number of bytes of data received matches the code length specified in the previous protocol step.

Since the SRAM is protected by 64-bit wide Error Correction Code (ECC), BAF always writes to SRAM grouped in 64-bit double-words. If the last byte does not fall into a 64-Bit bounder, the BAF fills it with zero's and renders the data aligned on a 64-bit bounder. After the last data is written to SRAM, the BAF writes a dummy double-word (0x0000_0000_0000_0000) to the next address location. This is written to avoid possible ECC error during core prefetch.

48.3.5.7 Execute code

The BAF program waits for the last echo message transmission to be completed. Then it restores the initial MCU configuration and jumps to the loaded code at Start Address which was received in step 2 of the protocol. At this point the BAF has finished its tasks and the MCU is controlled by new code executing from SRAM.

48.3.6 Serial boot configuration

48.3.6.1 LINFlexD configuration

Boot using UART protocol is implemented by LINFlexD module.

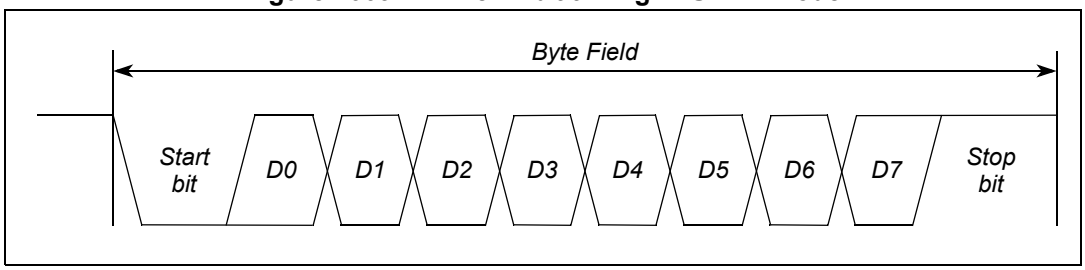
Note: See the pin configuration table in Chapter 7: Device configuration for pins used by the LINFlexD module.

When Serial Boot mode is started the LINFlexD_Rx is configured as an input and multiplexed to the LINFlexD module. The LINFlexD_Tx remains in a passive input state until the first byte of protocol is received. It is then configured as an output and multiplexed to LINFlexD module. The application may not be using the Serial Boot Mode function, and may be using the pin used by LINFlexD_Tx as an input, which may have some external hardware driving the pin. Inadvertently driving this pin as an output would cause contention and possible damage.

The LINFlexD module operates in half-duplex mode. The interface is either transmitting a byte or receiving a byte, but never both at the same time. It is expected that LINFlexD pins are connected to a physical layer bus driver/receiver (PHY). Many PHYs enforce half-duplex mode by using a single data channel for both transmitting and receiving data . The side effect of this is that all data transmitted on Tx pin is also received on Rx pin. To prevent the LINFlexD module receiving its own transmission, the Rx pin is blocked during Tx pin transmission.

The LINFlexD controller is configured to operate with an 8-bit data frame without parity bit and 1 stop bit.

Figure 1009. LINFlexD bit timing in UART mode

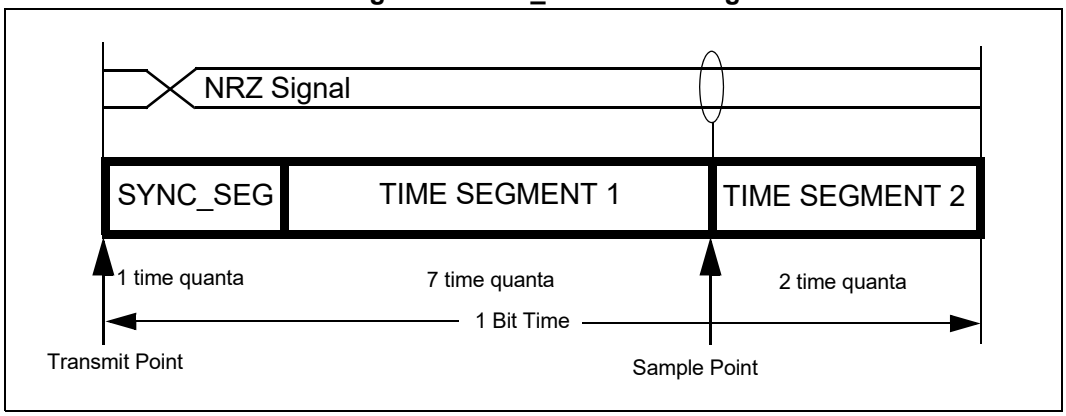


48.3.6.2 M_CAN configuration

The M_CAN controller is configured to operate at a baud rate equal to crystal frequency divided by 40, using the standard 11-bit identifier format detailed in CAN 2.0A specification (see *Table 973* for baud rates).

The Bit timing is configured as shown in *Figure 1010*

Figure 1010. M_CAN Bit Timing



48.3.6.3 Baud rates

The LINFlexD baud rate is set by frequency of external crystal oscillator/833.333 . *Table 973* below shows some possible baud rates.

The M_CAN baud rate is set by the frequency of the external crystal/40.

Table 973. LinFlexD_0 and M_CAN_1 baud rates

Frequency	LINFlexD baud rate	M_CAN baud rate
8 MHz	9,600 baud	200 kbaud
12 MHz	14,400 baud	300 kbaud
16 MHz	19,200 baud	400 kbaud
20 MHz	24,000 baud	500 kbaud
40 MHz	48,000 baud	1000 kbaud

48.3.6.4 LINFlexD and M_CAN pin configuration

Please see *Chapter 7: Device configuration* for pin details.

48.3.6.5 Protocol summary

Table 974 summarizes the LINFlexD protocol and BAF action during this boot mode.

Table 974. LINFlexD serial boot mode download protocol

Protocol steps	Host message sent	BAF response message	Action
1	64-bit password (MSB first)	64-bit password (MSB first)	Password checked for validity and compared with stored password.
2	32-bit store address	32-bit store address	Load address is stored for future use.

Table 974. LINFlexD serial boot mode download protocol (continued)

Protocol steps	Host message sent	BAF response message	Action
3	VLE bit + NO_ECHO bit + 30 bits of download data size in bytes (MSB first)	VLE bit + NO_ECHO bit + 30 bits of download data size in bytes (MSB first)	Size of download is stored for future use. Verify VLE bit.
4	8 bits of raw binary data	If NO_ECHO bit is set, data is not echoed, otherwise data is echoed.	8 x 8 bits of data are packed into 64-bit double-words. These double-words are stored in SRAM starting from the Load address. The Load address increments until the number of data received and stored matches the size as specified in the previous steps.
5	None	None	Branch to downloaded code

Table 975 shows the M_CAN serial boot protocol.

Table 975. M_CAN serial boot mode download protocol

Protocol steps	Host message sent	BAF response message	Action
1	CAN ID 0x011 + 64-bit password	CAN ID 0x001 + 64-bit password	Password checked against fixed password (0xFEED_FACE_CAFE_BEEF). Watchdog timer is refreshed if the password check is successful.
2	CAN ID 0x012 + 32-bit store address + 32-bit (1 VLE bit, 1 NO_ECHO bit and 30 of download data size in bytes)	CAN ID 0x002 + 32-bit store address + 32-bit (1 VLE bit, 1 NO_ECHO bit and 30 of download data size in bytes)	Load address and size of download are stored for future use.
3	CAN ID 0x013 + 8 to 64-bits of raw binary data	VLE bit + 31 bit number of bytes (MSB first)	Each byte of data received is store in MCU memory, starting at the address specified in the previous step and incrementing until the amount of data received and stored, matched the size as specified in the previous step.
4	None	None	The BAF returns IO pins and CAN module to their reset state, then branches to the first address the data was stored to (As specified in step 2)
5	None	None	Branch to downloaded code

48.4 Resources accessed by BAF code execution

The BAF code execution uses several IPs to achieve the main functions.

- Magic carpet (clocks)

Several clock configurations are made depending on the boot mode of the BAF. All settings are restored before execution of the application code (in flash memory or received via serial download).

- LINFlexD and M_CAN

The LINFlexD module is only used if normal boot header search fails and a serial download boot is attempted, or from the SER_BOOT_CBACK callback function. All settings are restored before the application code in system RAM is executed.

- PFLASH

PFLASH module is used to disable BAF execution. Remapping of the external overlay memory is configured automatically under certain conditions.

- Interrupts

No interrupts/exceptions are used or enabled, except the Machine Check exception, to handle ECC errors.

- SSCM

Used to read some register bits (mirrors of DCF)

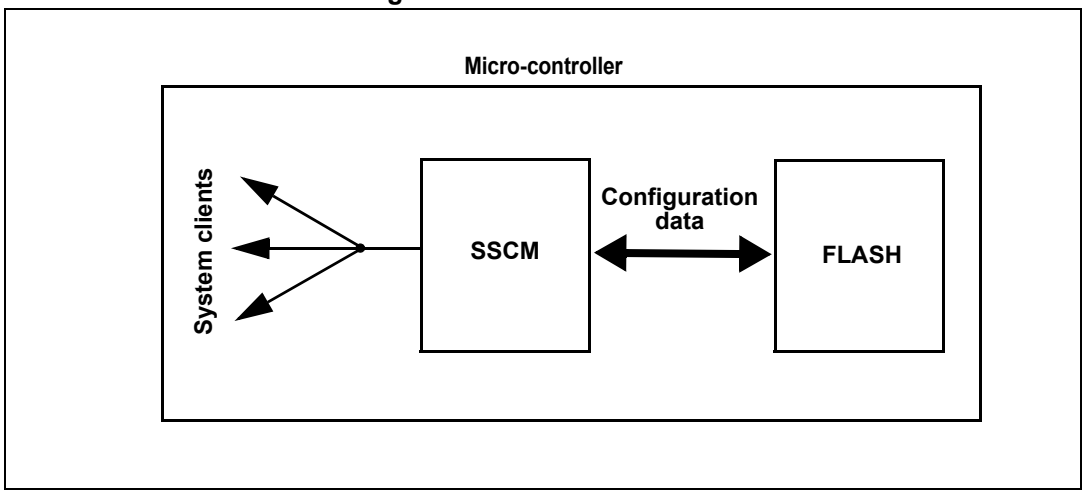
49 System Status and Configuration Module (SSCM)

49.1 Introduction

49.1.1 Overview

The System Status and Configuration Module (SSCM) is aimed to read the system configuration from the Flash and distribute to various clients inside the micro-controller, pictured in *Figure 1011*.

Figure 1011. SSCM overview



49.1.2 Features

The SSCM includes these distinct features:

- System Configuration and Status
  - Device Mode and System Status
  - Determine primary boot vector
  - Decodes the Device Life Cycle

49.1.3 Modes of operation

The SSCM operates identically in all system modes.

49.2 Memory map and register definition

This section provides a detailed description of all memory-mapped registers in the SSCM.

Table 976 shows the memory map for the SSCM. Note that all addresses are offsets; the absolute address may be calculated by adding the specified offset to the base address of the SSCM.

Note: All registers are accessible via 8-bit, 16-bit or 32-bit accesses. However, 16-bit accesses must be aligned to 16-bit boundaries, and 32-bit accesses must be aligned to 32-bit boundaries. As an example, the STATUS register is accessible by a 16-bit READ/WRITE to

address Base + 0x0002, but performing a 16-bit access to Base + 0x0003 is illegal.

Table 976. SSCM memory map

Address offset	Register	Access	Reset value	Section
0x0000	SSCM System Status (STATUS)	R/W	0xyyy0(1)	*Section 49.2.1.1*
0x0002	SSCM System Memory and ID Register (MEMCONFIG)	R	0xyyyy(1)	*Section 49.2.1.2*
0x0004–0x0005	Reserved			
0x0006	SSCM Error Configuration Register (ERROR)	R/W	0x0000	*Section 49.2.1.3*
0x0008–0x0009	Reserved(2)			
0x000A–0x001C	Reserved			
0x0020–0x0023	Reserved			
0x0024–0x0027	Reserved			
0x0028	Processor Start Address Register (PSA)	R	0xyyyy_yyyy(1)	*Section 49.2.1.4*
0x002C–0x002F	Reserved			
0x0030–0x0033	Reserved			
0x0034	Life Cycle Status Register (LCSTAT)	R	0x0000_0y0y(1)	*Section 49.2.1.5*
0x0038–0x004F	Reserved			

1. See *Section 49.2.1: Register descriptions* for reset value details.
2. No register abort on this location.

49.2.1 Register descriptions

The following registers are available in the SSCM. Those bits that are shaded out are reserved for future use. To optimize future compatibility, these bits should be masked out during any read/write operations to avoid conflict with future revisions.

49.2.1.1 System Status Register (STATUS)

The System Status register reflects the current state of the system.

Offset: 0x0000 Access: Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	CER	0	0	NXEN	1	0	0	BMODE			VLE	0	0	0	0
W		w1c														
Reset	0	0	0	-(2)	-(2)	1	0	0	-(2)			-(1)	0	0	0	0

1. Reset value depends on the associated option bits in flash memory.
2. Reset value depends on the device status after leaving reset.

Figure 1012. System Status Register (STATUS)

Table 977. STATUS field descriptions

Field	Description
1 CER	Configuration Error This field indicates that the SSCM has detected a configuration error while loading DCF clients (with PARITY Enabled). This error may have been detected during the reset sequence. 0 No configuration problem detected by the SSCM 1 Device configuration is not correct If a non-permanent (transient) error is detected, this bit can be cleared by writing a 1. If the error is permanent, the bit will reappear immediately.
4 NXEN	Processor 0 Nexus enable status. 0 Processor 0 Nexus disabled. 1 Processor 0 Nexus enabled
8:10 BMODE	Device Boot Mode 000 Reserved 001 Reserved 010 Reserved 011 User Code Boot (BAF not executed) 100 Reserved 101 BAF Code Boot (BAF executed) 110 Reserved 111 Reserved This field is only updated during reset.
11 VLE	Variable Length Instruction Mode When booting in User Code Boot mode, this field indicates that the code stored in the flash memory is using the VLE instruction set. The value of this field is determined by the RCHW field of the flash memory boot sector. 0 User Code Boot Location contains standard PPC code 1 User Code Boot Location contains VLE code

49.2.1.2 System Memory and ID Register (MEMCONFIG)

The System Memory and ID register is a read-only register which reflects the memory configuration of the system. It also contains the JTAG ID.

Offset: 0x0002 Access: Read Only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	JPIN										1	MREV				0
W																
Reset	0/1(1)	0/1(1)	0/1(1)	0/1(1)	0/1(1)	0/1(1)	0/1(1)	0/1(1)	0/1(1)	0/1(1)	1	0/1(1)	0/1(1)	0/1(1)	0/1(1)	0

1. Reset value reflects the JTAG ID of the device.

Figure 1013. System memory and ID register (MEMCONFIG)

Table 978. MEMCONFIG field descriptions

Field	Description
0:9 JPIN	JTAG Part ID Number
11:14 MREV	Minor Mask Revision

49.2.1.3 Error Configuration Register (ERROR)

The Error Configuration register is a read-write register that controls the error handling of the system.

Offset: 0x0006 Access: Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0(1)	PAE	RAE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1014. Error Configuration Register (ERROR)

Table 979. ERROR field descriptions

Field	Description
14 PAE	Peripheral Bus Abort Enable This bit enables bus aborts on: – Any access to a peripheral slot that is not used on the device. This feature is intended to aid in debugging when developing application code. – Illegal accesses to off-platform peripherals. On platform peripherals cannot be controlled by SSCM peripheral abort. 0 Illegal accesses to non-existing peripherals do not produce a Prefetch or Data Abort exception 1 Illegal accesses to non-existing peripherals produce a Prefetch or Data Abort exception
15 RAE	Register Bus Abort Enable This bit enables bus aborts on illegal accesses to off-platform peripherals. Illegal accesses are defined as reads or writes to reserved addresses within the address space for a particular peripheral. This feature is intended to aid in debugging when developing application code. 0 Illegal accesses to peripherals do not produce a Prefetch or Data Abort exception 1 Illegal accesses to peripherals produce a Prefetch or Data Abort exception **Note:** Transfers to peripheral bus resources may be aborted even before they reach the peripheral bus (in example, at the PBRIDGE level). In this case, the PAE and RAE register bits have no effect on the abort.

49.2.1.4 Processor Start Address Register (PSA)

Offset 0x0028 Access: Read Only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	SADR(1)															
W																
Reset	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	SADR(1)															
W																
Reset	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1

1. Reset value is the start address.

Figure 1015. Processor Start Address Register (PSA)

Table 980. PSA field descriptions

Field	Description
0:31 SADR	Start Address The boot processor will start executing application code from this address. In User Code Boot mode this indicates the location determined by the RCHW search. In BAF Code Boot mode this will show the start address of the BAF code. In the case of a serial download, this register will point to the BAM start location.

49.2.1.5 Life Cycle Status Register (LCSTAT)

Offset 0x0034 Access: Read Only

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	LC		
W																
Reset	0	0	0	0	0	0	0/1(1)	0/1(1)	0	0	0	0	0	0/1(2)	0/1(2)	0/1(2)

1. Reset value reflects the Test Life Cycle of the device.
2. Reset value is the Life Cycle of the device.

Figure 1016. Life Cycle Status Register (LCSTAT)

Table 981. LCSTAT field descriptions

Field	Description
29:31 LC	Life Cycle As determined by the SSCM by reading the LC slots. The encoding is: 000 FAIL_ANALYSIS 001 Reserved 010 PROD_OEM 011 CUST_DELIV 100 Reserved 101 Reserved 110 ST_PROD 111 IN_FIELD

49.3 Functional description

The primary purpose of the SSCM is to provide information about the current state and configuration of the system that may be useful for configuring application software and for debug of the system.

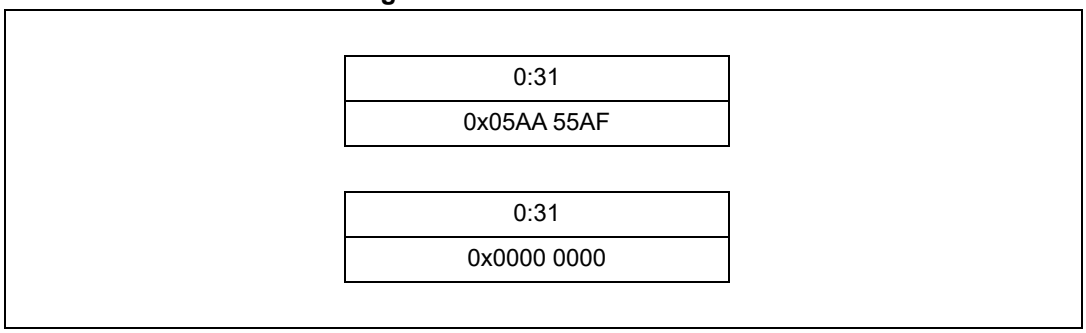
49.3.1 DCF mechanism

The DCF mechanism has been developed to handle settings of device parameters via OTP flash memory.

Starting at location UTEST_OFFSET the user can store a series of DCF records - the device will process these records during the system reset sequence before the CPU leaves reset. The first record needs to be a start record followed by an application-specific number of data records as required, and finally a stop record.

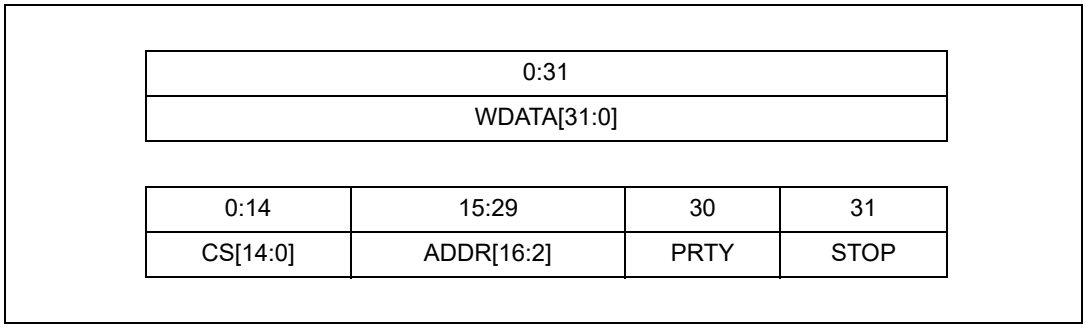
Each record is 64 bits in length. *Figure 1017* shows the DCF start record. The start record must be placed at UTEST_OFFSET to indicate to the device that the following data records should be processed.

Figure 1017. DCF start record



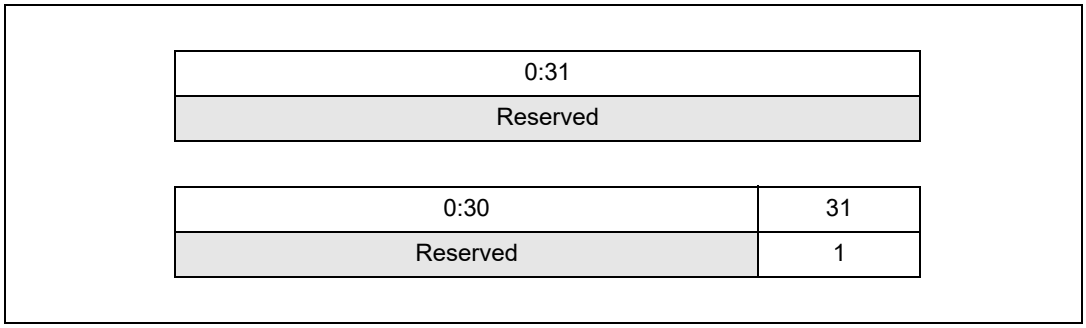
The data records are structured similar to a CPU write instruction - 15 bits of client select, 15 bits of address, 32 bits of payload data plus a STOP bit - See *Figure 1018*. In a data record the STOP bit must always be programmed to 0. Some clients support a parity bit.

Figure 1018. Format of a DCF data record



The stop record indicates that processing should stop. The general format of the stop record is shown in *Figure 1019*. Only the STOP bit needs to be 1 in order to form a stop record – All other bits are ignored. An unprogrammed location in UTEST flash will be interpreted as a stop record.

Figure 1019. DCF stop record



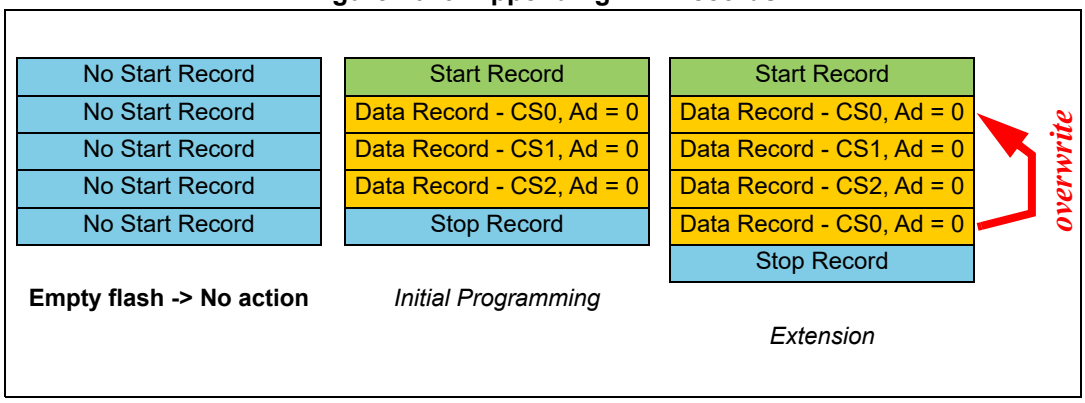
If *n* data records are to be stored in UTEST, the data structure must be as shown in *Table 982*.

Table 982. Series of DCF records in UTEST

ADDR offset	DATA			
0x00	0x05AA_55AF			
0x04	0x0000_0000			STOP=0
0x08	WDATA[31:0]			
0x0C	CS[14:0]	ADDR[16:2]	PRTY	STOP=0
0x10	WDATA[31:0]			
0x14	CS[14:0]	ADDR[16:2]	PRTY	STOP=0
...	...			
8n - 1 + 0x0	Reserved			
8n - 1 + 0x4	Reserved			1
8n + 0x0				
8n + 0x4				

There must never be an unprogrammed record in the data structure, as that would be interpreted as a stop record, so subsequent records would be ignored. This allows programming the records in several sessions, each time appending new records at the end of the list, as shown in *Figure 1020*.

Figure 1020. Appending DCF records



In this case, a record may overwrite a client's value which was already set by a previous record. However whether such an operation is allowed depends on the rules given for the client - some clients are write-once, only allow setting bits but not clearing, and so on.

The parity bit (PRTY) is required for some DCF records. The parity scheme used is even parity, so the number of 1s in the WDATA and PRTY fields needs to be even. (For example, if the WDATA field has the value 0x0000_0001 then the PRTY field needs to be set to '1', so that the total number of 1s is even).

49.3.2 Boot mode functionality

The device supports the following boot modes for the main boot core:

- User Code Boot mode (SSCM): the device will boot from the first bootable section of the flash memory main array (BOOT_ID = 0x5A).
- BAF Code Boot: the device will boot from UTEST NVM Flash Block, (interface with the HSM if available) and perform device setup (BOOT_ID = 0xA5).

If booting is not possible with the selected configuration (for example, no Boot ID is found in the selected boot location for SSCM and BAF, that is 0x5A and 0xA5 respectively) then the device will enter serial boot mode.

49.3.3 BAF configuration

In BAF Code Boot mode the boot core will start to execute the BAF code at the location provided in the "Embedded Flash Memory" chapter (TestFlash block memory map). The BMODE field in the SSCM status register will be set to the value 0b101. Details of the BAF operation are described in the BAF chapter.

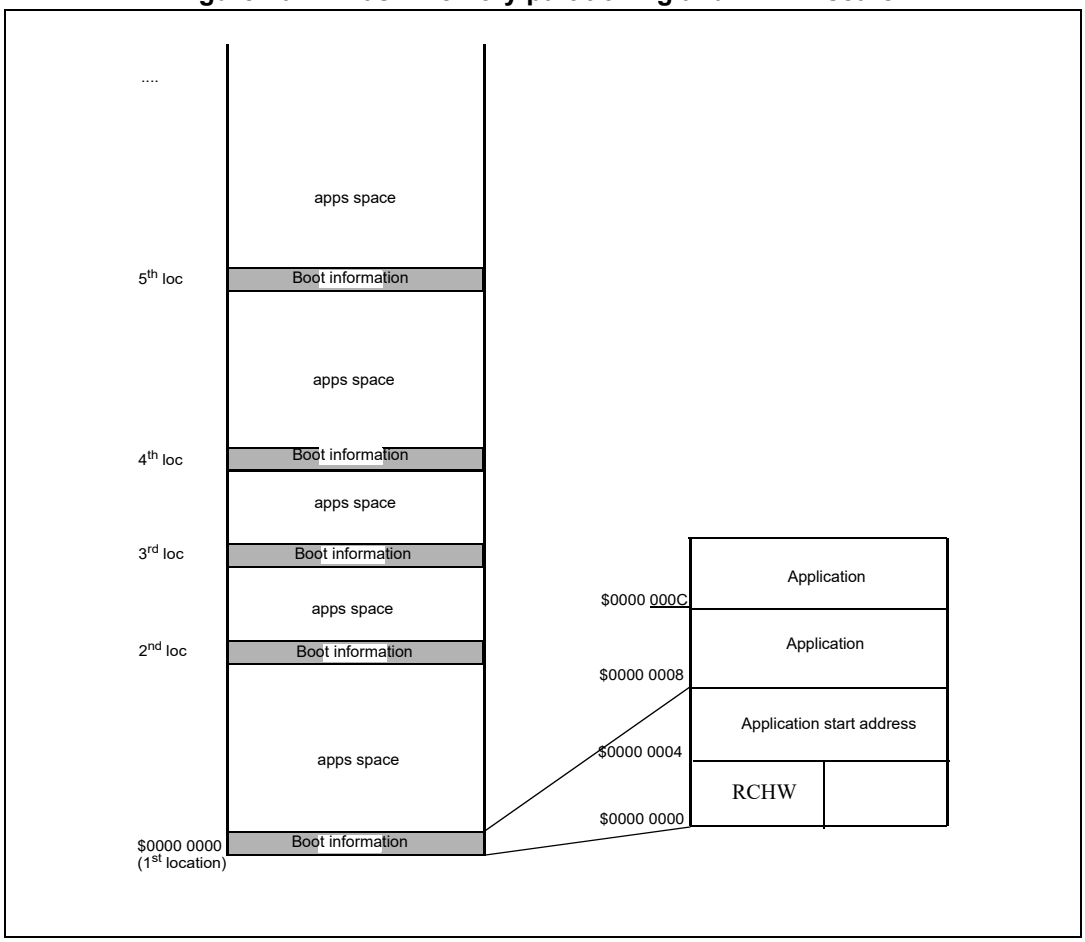
49.3.4 User code boot sector search

49.3.4.1 Potential boot sectors

As shown in *Figure 1021* in User Code Boot mode, several locations are searched for a valid boot ID. The lowest sector which starts with a valid boot ID will be used to boot the device.

The flash memory location that are searched for boot information are shown in the “Reset and Boot” chapter (section “HW boot record search (SSCM search)”).

Figure 1021. Flash memory partitioning and RCHW search



49.3.4.2 Reset Configuration Half-Word

For each boot sector in the flash memory, SSCM checks the content of the Reset Configuration Half-Word (RCHW) at offset 0x00. If the RCHW field BOOT_ID holds the value 0x5A then the sector is considered bootable. In addition, there is a flag which indicates that the code is VLE code. (If a device only supports VLE code for future compatibility the flag must always be set to ‘1’.) All other bits are reserved.

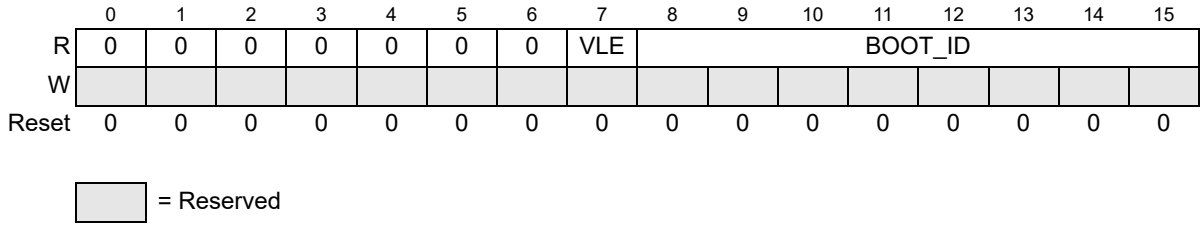


Figure 1022. Reset Configuration Half-Word

Once the device detects that it needs to boot from flash memory, and finds a valid BOOT_ID, it will boot from the application start address at offset 0x04 within the boot sector.

49.3.5 Life Cycle

The SSCM will determine the Life Cycle of the device by reading the Life Cycle slots from UTEST flash memory area. The read operation is done during the reset phase with normal timings and it is protected by both operating monitors and ECC check. In addition a set of sanity checks executed over the LC read data guarantee the integrity of the final LC value.

At the end of the reset phase, the LC can have one of the following values:

- ST Production
- Customer Delivery
- OEM Production
- In Field
- FA

The LC is written into 5 slots, 128 bits each, at fixed locations in UTEST flash memory block. Each LC slot is read in one single atomic operation and it is organized into two fields:

- The valid field
- The invalid field

Depending on the possible combination of the data programmed into these fields, each LC slot can have one of the possible 4 status:

Table 983. Life Cycle slots

Use case	Offset	LC slot	Valid field (64 bits)		Invalid field (64 bits)		LC slot status
1	0x00-0x10	1	Erased	Erased	Erased	Erased	Erased
2	0x00-0x10	1	Marked	Erased	Marked	Erased	Inactive
3	0x00-0x10	1	Marked	Erased	Erased	Erased	Active
4	0x00-0x10	1	Any other value				Illegal

“Marked” in this case means that the value has been programmed with the bit pattern 0x55AA_50AF_55AA_50AF, “Erased” is detected by the bit pattern 0xFFFF_FFFF_FFFF_FFFF.

Table 984 shows how the slots are arranged in memory.

Table 984. Life Cycle Slots in Memory

Offset	LC Slot word
0x00	Valid[31:0]
0x04	Valid[63:32]
0x08	Valid[31:0]
0x0C	Valid[63:32]
0x10	Invalid[31:0]
0x14	Invalid[63:32]
0x18	Invalid[31:0]
0x1C	Invalid[63:32]

The Life Cycle is determined as shown in *Table 985*. The priority of the entries is from top to bottom, so the first row which applies determines the resulting Life Cycle.

Table 985. Resulting Life Cycle

LC Slot 0 ST_PROD(1)	LC Slot 1 CUST_DEL(1)	LC Slot 2 OEM_PROD(1)	LC Slot 3 IN_FIELD(1)	LC Slot 4 FA(1)	Resulting LifeCycle
Active	Erased	Erased	Erased	Erased	ST Production
Inactive	**Active**	Erased	Erased	Erased	Customer Delivery
Inactive	*Inactive*	**Active**	Erased	Erased	OEM Production
Inactive	*Inactive*	*Inactive*	**Active**	Erased	In Field
Inactive	*Inactive*	*Inactive*	*Inactive*	**Active**	FA
Erased	Erased	Erased	Erased	Erased	System Reset

1. Erased = all slot bits at ‘1’

49.4 Initialization and application information

49.4.1 Reset

The reset state of each individual bit is shown within the Register Description section (see *Section 49.2.1: Register descriptions*).

49.5 Additional safety measures

49.5.1 Spurious reset protection

The SSCM implements protection against spurious module resets (for example, resets which only effect the SSCM but not other modules) by gating the state machines with the expected status of the MC_RGM. In case of a spurious reset the SSCM will not interfere with the flash memory bus or overwrite configuration registers. Status internal to the SSCM will be lost, however.

50 Power management controller digital interface (PMC_Dig)

50.1 Introduction

The Control PMU manages the PMU at the reset and during the run, it checks the LVD or HVD accuracy.

The PMC analog block interfaces the digital part of the device by the PMC_dig block .

This chapter describes the digital block of the PMU.

The PMC_Dig contains all of the registers and digital logic to generate all of the enable signals, trim control bits, POR reset generation, and test mode logic for the analog block.

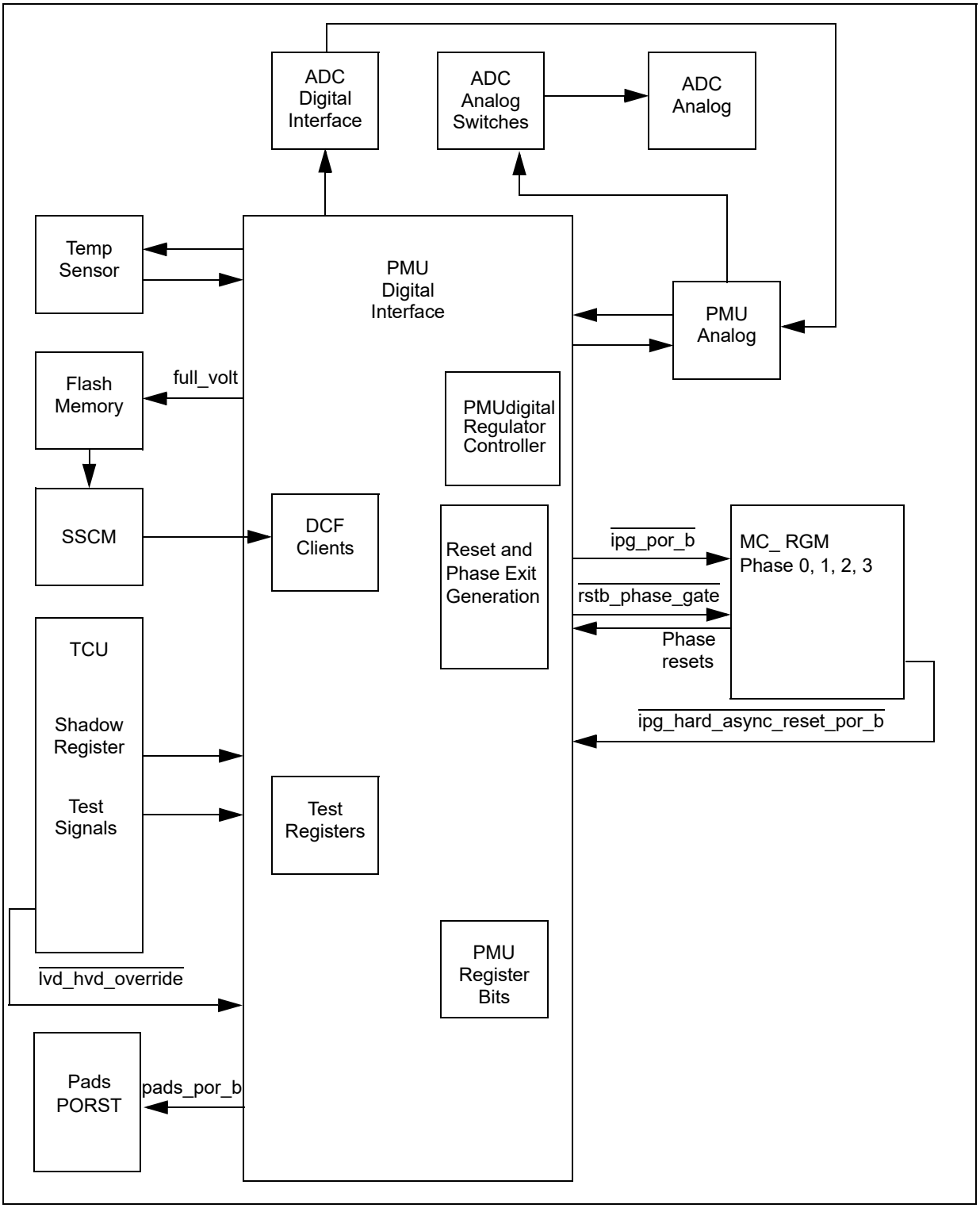
In addition, the PMU digital logic contains the enable signals, trim bits and other registers for the temperature sensor.

There are also several custom interfaces to various other blocks: The Reset Generation Module (MC_RGM) block receives the LVD values during POR to use for phase transition conditions.

There is also a custom interface to flash memory via the SSCM (and DCF client blocks) that is used to load the trim values.

The PMU digital block generates a transfer error event to the system if the selected address is greater than the highest address in the PMU digital memory map. A transfer error is not generated if an unused address within the PMU digital memory space is accessed. See in *Figure 1023* the PMUDig top level block diagram interfaced to the device modules.

Figure 1023. Digital PMU block diagram



50.2 Main features

- PMU management at a main reset
- Resets signal generation

50.2.1 Standard features

- IPS bus interface

50.3 IPS bus interface

The IPS bus interface is a slave bus used for configuration purposes via CPU. The following bus operations (contiguous byte enables) are supported.

- Word (32 bits) data write/read operations to any registers.
- Low and high half-words (16 bits, data[31:16] or data[15:0]) data write/read operations to any registers.
- Byte(8 bits, data[31:24] or data[23:16] or data[15:8] or data[7:0]) data write/read operations to any registers, except BIST time registers (BIST_TIME10, BIST_TIME32 and BIST_TIME65).
- Any other operation (free byte enables or other operations) must be avoided.

The PMC_DIG block generates a transfer error in the following cases.

- Any write/read access to the register addresses not mapped on the peripheral but included in the address space of the peripheral.
- Any write/read operation different from byte/hword/word (free byte enables or other operations) on each register.

The registers of the PMC_DIG block are accessible (read/write) in each access mode: user, supervisor or test.

About the Reset used among the register.

- The POR reset is used on EPR registers.
- The destructive reset is used on REE, RES and FEE registers.
- The functional reset is used IE, CTL_TD.

50.4 Memory Map and Registers

Table 986 shows the PMU memory map. The PMU includes many registers for configuring and monitoring LVD monitors, some other control registers.

Table 986. Power Management Controller Memory Map

Address	Register	Location
0x0000	Event Pending Register LV0 (EPR_LV0)	*Section 50.5.1*
0x0004	Reset Event Enable LV0 (REE_LV0)	*Section 50.5.2*
0x0008	Reset Event Select LV0 (RES_LV0)	*Section 50.5.3*
0x000C	Interrupt Enable LV0 (IE_LV0)	*Section 50.5.4*
0x0010	FCCU Event Enable LV0 (FEE_LV0)	*Section 50.5.5*

Table 986. Power Management Controller Memory Map (continued)

Address	Register	Location
0x0014–0x001F	Reserved	
0x0020	Event Pending Register LV1 (EPR_LV1)	*Section 50.5.6*
0x0024	Reset Event Enable LV1 (REE_LV1)	*Section 50.5.7*
0x0028	Reset Event Select LV1 (RES_LV1)	*Section 50.5.8*
0x002C	Interrupt Enable LV1 (IE_LV1)	*Section 50.5.9*
0x0030	FCCU Event Enable LV1 (FEE_LV1)	*Section 50.5.10*
0x0034–0x003F	Reserved	
0x0040	Event Pending Register HV0 (EPR_HV0)	*Section 50.5.11*
0x0044	Reset Event Enable HV0 (REE_HV0)	*Section 50.5.12*
0x0048	Reset Event Selection HV0 (RES_HV0)	*Section 50.5.13*
0x004C	Interrupt Enable HV0 (IE_HV0)	*Section 50.5.14*
0x0050	FCCU Event Enable HV0 (FEE_HV0)	*Section 50.5.15*
0x0054–0x005F	Reserved	
0x0060	Event Pending Register HV1 (EPR_HV1)	*Section 50.5.16*
0x0064	Reset Event Enable HV1 (REE_HV1)	*Section 50.5.17*
0x0068	Reset Event Selection HV1 (RES_HV1)	*Section 50.5.18*
0x006C	Interrupt Enable HV1 (IE_HV1)	*Section 50.5.19*
0x0070	FCCU Event Enable HV1 (FEE_HV1)	*Section 50.5.20*
0x0074–0x007F	Reserved	
0x0080	Supply Gauge Status register (GR_S)	*Section 50.5.21*
0x0084	Pending Gauge Status register (GS_P)	*Section 50.5.22*
0x0088	Global Interrupt Enable register (IE_G)	*Section 50.5.23*
0x008C–0x0103	Reserved	
0x0104	Voltage Selection of IO (VSIO)	*Section 50.5.24*
0x0108–0x023F	Reserved	
0x0240	IPS Regulator Control PMOS NMOS register (IREGU_CTL_PN)	*Section 50.5.25*
0x0244–0x02FF	Reserved	
0x0300	Temperature Event Pending register (EPR_TD)	*Section 50.5.26*
0x0304	Temperature Reset Event Enable register (REE_TD)	*Section 50.5.27*
0x0308	Temperature Reset Event Selection register (RES_TD)	*Section 50.5.28*
0x030C	Temperature Sensor Configuration register (CTL_TD)	*Section 50.5.29*
0x0310–0x0317	Reserved	
0x0318	Temperature FCCU Event Enable register (FEE_TD)	*Section 50.5.30*
0x031C–0x037F	Reserved	

Table 986. Power Management Controller Memory Map (continued)

Address	Register	Location
0x0368	IPS Regulator Monitor register (IREGU_MON)	*Section 50.5.31*
0x03A8	ESR0 ConFiGuration register (ESR0_CFG)	*Section 50.5.32*
0x03AC–0x03CB	Reserved	
0x03CC	IO User BIST Flags register (BIST_FLAGS_IO10)	*Section 50.5.33*
0x03D0	User BIST Flags register (BIST_FLAGS)	*Section 50.5.34*
0x03D4	User BIST Control register (BIST_CTRL)	*Section 50.5.35*
0x03D8	User BIST Time1 and Time0 register (BIST_TIME10)	*Section 50.5.36*
0x03DC	User BIST Time3 and Time2 register (BIST_TIME32)	*Section 50.5.37*
0x03E0	User BIST Time6 and Time5 register (BIST_TIME65)	*Section 50.5.38*
0x03E4	User BIST VD Under Test Monitor Register (BIST_DEBUG)	*Section 50.5.39*

Register ESR0_CFG can be accessed both by IPS and DCF with their corresponding IPS and DCF addresses respectively.

Table 987 shows how the 32-bit registers for LV0, LV1, HV0, and HV1 are organized.

Table 987. 32-bit registers LV0, LV1, HV0, HV1 arrangement table

Register class	BYTE 3	BYTE 2	BYTE 1	BYTE 0
LV0	VD3=lvd108_c	VD2=mvd098_fl_c	VD1=mvd085t_c	VD0=por041_c
LV1	VD7=uvd145_fl_c	VD6=hvd140_c	VD5=Reserved	VD4=Reserved
HV0	VD11=lvd290_adl1_adl0_fl_iol1_osc_c	VD10=mvd270_fl_c	VD9=mvd240t_c	VD8=por200_c
HV1	VD15 = uvd600_c	VD14=lvd400_adl1_adl0_iol1_aiol0	VD13=hvd400_c	VD12=Reserved

Where each byte VD# is so structured:

Bit7, ADL1, ADL0 , FLASH, IOL1, IOL0, Bit1, CORE.

Table 988 shows the correspondence between VD number and VD name, where HV means a 3.3V/5.0V supply and LV means a 1.2V supply.

Table 988. VD number versus VD name table

VD Number	VD Name	VD Meaning
VD0	POR041_C	LV detector asserts when PMU supply < 0.41V
VD1	MVD085t_C	LV detector asserts when PMU supply < 0.85V
VD2	MVD098_C	LV detector asserts when PMU supply < 0.98V
VD2	MVD098_FL	LV detector asserts when FLASH supply < 0.98V
VD3	LVD108_C	LV detector asserts when PMU supply < 1.08V

Table 988. VD number versus VD name table (continued)

VD Number	VD Name	VD Meaning
VD6	HVD140_C	LV detector asserts when PMU supply > 1.40V
VD7	UVD145_C	LV detector asserts when PMU supply > 1.45V
VD7	UVD145_FL	LV detector asserts when FLASH supply > 1.45V
VD8	POR200_C	HV detector asserts when PMU supply < 2.00V
VD9	MVD240t_C	HV detector asserts when PMU supply < 2.40V
VD10	MVD270_C	HV detector asserts when PMU supply < 2.70V
VD10	MVD270_FL	HV detector asserts when FLASH supply < 2.70V
VD11	LVD290_C	HV detector asserts when PMU supply < 2.90V
VD11	LVD290_XOSC	HV detector asserts when XOSC supply < 2.90V
VD11	LVD290_IOL1	HV detector asserts when IO1 supply < 2.90V
VD11	LVD290_FL	HV detector asserts when FLASH supply < 2.90V
VD11	LVD290_ADL0	HV detector asserts when ADC0 supply < 2.90V
VD11	LVD290_ADL1	HV detector asserts when ADC1 supply < 2.90V
VD13	HVD400_C	HV detector asserts when PMU supply > 4.00V
VD14	LVD400_IOL0	HV detector asserts when IO0 PMU supply < 4.00V
VD14	LVD400_IOL1	HV detector asserts when IO1 supply< 4.00V
VD14	LVD400_ADL0	HV detector asserts when ADC0 supply < 4.00V
VD14	LVD400_ADL1	HV detector asserts when ADC1 supply< 4.00V
VD15	UVD600_C	HV detector asserts when PMU supply > 6.00V

50.5 Register description

50.5.1 Event Pending Register (EPR_LV0)

This Event Pending Register indicates the present and past state of the voltage detect signalsin LOW VOLTAGE 0 range. If the voltage detect event has ever passed the trigger event since the last clearing event, the flag bit is set. In order to clear the flag, a one must be written to the flag bit that needs to be cleared.

Offset: 0x0000 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	LVD3_C	0	0	0	0	0	0	0	0
W								w1c								
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

Figure 1024. Event Pending Register (EPR_LV0)

Table 989. EPR_LV0 field descriptions

Field	Description
7 LVD3_C	LVD3_C flag This bit is the low-voltage status flag associated with LVD108_C VD, the voltage level detect for the low voltage Core 1.08V supply. It is asserted when the supply falls below its corresponding Voltage Detector 3 threshold, and clears when the supply rises above its corresponding LVD3 threshold and a one is written to this bit location. 0 Currently no occurrence 1 VD occurrence detected on the low voltage1.08V point supply

50.5.2 Reset Event Enable Register (REE_LV0)

This Reset Event Enable Register controls whether the voltage detect signal event causes a reset. If the desired flag bit is enabled, the voltage detect event causes a reset to be generated when the selected voltage passes the trigger event. These bits are loaded from the flash during the boot sequence. If the flash loaded bits are programmed to be enabled (set), these bits cannot be disabled (cleared).

Offset: 0x0004 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	REE3_C	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

Figure 1025. Reset Event Enable Register (REE_LV0)

Table 990. REE_VD3 field descriptions

Field	Description
7 REE3_C	REE3_C reset enable This bit defines whether LVD108_C VD, the voltage level detect for the low voltage CORE 1.08V supply of the cold point, generates a system reset. The REE3_C bit determines whether the reset is functional or destructive. 0 Disabled. LVD assertion on the supply of the cold point does not cause system reset 1 Enabled. LVD assertion on the supply of the cold point causes system reset

50.5.3 Reset Event Select Register (RES_LV0)

This Reset Event Select Register, in LOW VOLTAGE 0 range, controls whether the voltage detect signal event causes a destructive or functional reset. If the desired flag bit is enabled, the voltage detect event causes either a destructive or functional reset to be generated when the selected voltage passes the trigger event. These bits are loaded from the flash during the boot sequence.

Offset: 0x0008 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	RES3_C	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

Figure 1026. Reset Event Select Register (RES_LV0)

Table 991. RES_LV0 field descriptions

Field	Description
7 RES3_C	RES3_C reset event select This bit defines whether LVD108_C VD, the voltage level detect for the low voltage CORE 1.08 V supply of the cold point, generates a destructive reset or a functional reset. The RES3_C bit determines whether the reset event is enabled. 0 Destructive Reset Generated. LVD assertion on the supply of the cold point causes a destructive system reset 1 Functional Reset Generated. LVD assertion on the supply of the cold point causes a functional system reset

50.5.4 Interrupt Enable (IE_LV0)

This configuration register contains a set of LV0 interrupt Enable bits that correspond to each of the LV0 Event Pending Registers (EPR_LV0). These bits indicate whether an interrupt occurs when the voltage event is seen. A ‘0’ indicates that no interrupt is to be sent, and a ‘1’ indicates that an interrupt is to occur when the voltage detect event occurs. These

bits are replicated as a read-only monitor in the register (IE_G) at address 0088h. The IE_LV0 bits can be read or written at any time.

Offset: 0x000C Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	VD3IE_C	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

Figure 1027. Interrupt Enable Register (IE_LV0)

Table 992. IE_LV0 field descriptions

Field	Description
7 VD3IE_C	VD3IE_C Interrupt Enable This bit determines whether an interrupt is seen by the system when LVD108_C VD, the voltage level detect for the low voltage CORE 1.08V supply event occurs. 0 No interrupt occurs 1 An interrupt occurs

50.5.5 FCCU Event Enable Register (FEE_LV0)

This FCCU Event Enable Register LV0 controls whether the voltage detect signal events cause an event signal to be sent to the FCCU (Fault Control Unit). If the desired flag bit is enabled, the voltage detect event causes an FCCU event to be generated when the selected voltage passes the trigger event.

Offset: 0x0010 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	FEE3_C	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

Figure 1028. FCCU Event Enable Register (FEE_LV0)

Table 993. FEE_VD3 field descriptions

Field	Description
7 FEE3_C	FEE3_C FCCU event enable This bit defines whether an VD assertion of LVD108_C VD, the voltage level detect for the low voltage CORE 1.08V supply, generates an FCCU event. 0 Disabled. An VD assertion of LV0 on the supply of the CORE does not cause an FCCU event 1 Enabled. An VD assertion of LV0 on the supply of the CORE causes an FCCU event

50.5.6 Event Pending Register (EPR_LV1)

This Event Pending Register indicates the present and past state of the voltage detect signals, in LOW VOLTAGE1 range. If the voltage detect event has ever passed the trigger event since the last clearing event, the flag bit is set. In order to clear the flag, a '1' must be written to the flag bit that needs to be cleared.

Offset: 0x0020 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	HVD6_C
W																w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

Figure 1029. Event Pending Register (EPR_LV1)

Table 994. EPR_LV1 field descriptions

Field	Description
15 HVD6_C	UVD6_FL flag This bit is the Low Voltage status flag associated with HVD140_C VD, the voltage level detect for the low voltage CORE 1.40V supply. It is asserted when the supply falls below its corresponding Voltage Detector threshold, and clears when the supply rises above its corresponding HVD7 threshold and a '1' is written to this bit location. 0 Currently no occurrence 1 VD occurrence detected on the low voltage 1.45V point supply

50.5.7 Reset Event Enable Register (REE_LV1)

This Reset Event Enable Register controls, in LOW VOLTAGE 1 range, whether the voltage detect signal event causes a reset. If the desired flag bit is enabled, the voltage detect event causes a reset to be generated when the selected voltage passes the trigger event. These bits are loaded from the flash during the boot sequence. If the flash loaded bits are programmed to be enabled (set), these bits cannot be disabled (cleared).

Offset: 0x0024 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	REE6_C
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

Figure 1030. Reset Event Enable Register (REE_LV1)

Table 995. REE_LV1 field descriptions

Field	Description
15 REE6_C	REE6_C reset enable This bit defines whether a HVD140_C VD, the voltage level detect for the low voltage CORE 1.40V supply of the cold point, generates a system reset. The RES_LV1 bit determines whether the reset is functional or destructive. 0 Disabled. HVD assertion on the supply of the cold point does not cause system reset 1 Enabled. HVD assertion on the supply of the cold point causes system reset

50.5.8 Reset Event Select Register (RES_LV1)

This Reset Event Select Register controls, in LOW VOLTAGE 1 range, whether the voltage detect signal event causes a destructive or functional reset. If the desired flag bit is enabled, the voltage detect event causes either a destructive or functional reset to be generated when the selected voltage passes the trigger event. These bits are loaded from the flash during the boot sequence.

Offset: 0x0028 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	RES6_C
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

Figure 1031. Reset Event Select Register (RES_LV1)

Table 996. RES_LV1 field descriptions

Field	Description
15 RES6_C	RES6_C event select This bit defines whether a HVD140_C VD, the voltage level detect for the low voltage CORE 1.40V supply of the cold point, generates a destructive reset or a functional reset. The REE_VD3 bit determines whether the reset event is enabled. 0 Destructive Reset Generated. LVD assertion on the supply of the cold point causes a destructive system reset 1 Functional Reset Generated. LVD assertion on the supply of the cold point causes a functional system reset

50.5.9 Interrupt Enable (IE_LV1)

This configuration register contains a set of LV1 interrupt Enable bits that correspond to each of the LV1 Event Pending Registers (EPR_LV1). These bits indicate whether an interrupt occurs when the voltage event is seen. A ‘0’ indicates that no interrupt is to be sent, and a ‘1’ indicates that an interrupt is to occur when the voltage detect event occurs. These bits are replicated as a read-only monitor in the register (IE_G) at address 0088h. The IE_LV1 bits can be read or written at any time.

Offset: 0x002C Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	HVD6IE_C
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

Figure 1032. Interrupt Enable Register (IE_LV1)

Table 997. IE_LV1 field descriptions

Field	Description
15 HVD6IE_C	HVD6IE_C Interrupt Enable This bit determines whether an interrupt is seen by the system when HVD140_C VD, the voltage level detect for the low voltage CORE 1.40V supply the voltage event occurs. 0 No interrupt occurs 1 An interrupt occurs

50.5.10 FCCU Event Enable Register (FEE_LV1)

This FCCU Event Enable Register LV1 controls whether the voltage detect signal events cause an event signal to be sent to the FCCU (Fault Control Unit). If the desired flag bit is enabled, the voltage detect event causes an FCCU event to be generated when the selected voltage passes the trigger event.

Offset: 0x0030 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	FEE6_C
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

Figure 1033. FCCU Event Enable Register (FEE_LV1)

Table 998. FEE_LV1 field descriptions

Field	Description
15 FEE6_C	FEE6_C FCCU event enable This bit defines whether an VD assertion of HVD140_C VD, the voltage level detect for the low voltage CORE 1.40V supply, generates an FCCU event 0 Disabled. an VD assertion of LV1on the supply of the CORE does not cause an FCCU event 1 Enabled. an VD assertion of LV0 on the supply of the CORE causes an FCCU event

50.5.11 Event Pending Register (EPR_HV0)

This Event Pending Register indicates the present and past state of the voltage detect signalsin HIGH VOLTAGE 0 range. If the voltage detect event has ever passed the trigger event since the last clearing event, the flag bit is set. In order to clear the flag, a one must be written to the flag bit that needs to be cleared.

Offset: 0x0040 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	LVD11_ADL1	LVD11_ADL0	LVD11_FL	LVD11_IOL1	LVD11_OSC	0	LVD11_C	0	0	0	0	0	0	0	0
W		w1c	w1c	w1c	w1c	w1c		w1c								
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

Figure 1034. Event Pending Register (EPR_HV0)

Table 999. EPR_HV0 field descriptions

Field	Description
1 LVD11_ADL1	LVD11_ADL1 flag This bit is the high-voltage status flag associated with LVD11_ADL1, the voltage level detect for the low voltage ADC1 2.90V supply. It is asserted when the supply falls below its corresponding Voltage Detector 3 threshold, and clears when the supply rises above its corresponding LVD11 threshold and a one is written to this bit location. 0 Currently no occurrence 1 VD occurrence detected on the low voltage 2.90V point supply
2 LVD11_ADL0	LVD11_AD flag This bit is the high-voltage status flag associated with LVD11_ADL0, the voltage level detect for the low voltage ADC0 2.90V supply. It is asserted when the supply falls below its corresponding Voltage Detector 3 threshold, and clears when the supply rises above its corresponding LVD11 threshold and a one is written to this bit location. 0 Currently no occurrence 1 VD occurrence detected on the low voltage 2.90V point supply
3 LVD11_FL	LVD11_FL flag This bit is the high-voltage status flag associated with LVD290_FL VD, the voltage level detect for the high voltage assertion on the FLASH 2.90V supply. It is asserted when the supply falls below its corresponding Voltage Detector 11 threshold, and clears when the supply rises above its corresponding LVD11 threshold and a one is written to this bit location. 0 Currently no occurrence 1 VD occurrence detected on the low voltage 2.90V point supply

Table 999. EPR_HV0 field descriptions (continued)

Field	Description
4 LVD11_IOL1	LVD11_IOL1 flag This bit is the high-voltage status flag associated with LVD11_IOL1, the voltage level detect for the high voltage IO1 2.90V supply. It is asserted when the supply falls below its corresponding Voltage Detector 3 threshold, and clears when the supply rises above its corresponding LVD11 threshold and a one is written to this bit location. 0 Currently no occurrence 1 VD occurrence detected on the low voltage 2.90V point supply
5 LVD11_OSC	LVD11_OSC flag This bit is the high-voltage status flag associated with LVD11_OSC, the voltage level detect for the high voltage XOSC 2.90V supply. It is asserted when the supply falls below its corresponding Voltage Detector 3 threshold, and clears when the supply rises above its corresponding LVD11 threshold and a one is written to this bit location. 0 Currently no occurrence 1 VD occurrence detected on the low voltage 2.90V point supply
7 LVD11_C	LVD11_C flag This bit is the high-voltage status flag associated with LVD290_C VD, the voltage level detect for the high voltage Core/IO0 2.90V supply. It is asserted when the supply falls below its corresponding Voltage Detector 11 threshold, and clears when the supply rises above its corresponding LVD11 threshold and a one is written to this bit location. 0 Currently no occurrence 1 VD occurrence detected on the low voltage 2.90V point supply

50.5.12 Reset Event Enable Register (REE_HV0)

This Reset Event Enable Register controls whether the voltage detect signal event causes a reset. If the desired flag bit is enabled, the voltage detect event causes a reset to be generated when the selected voltage passes the trigger event. These bits are loaded from the flash during the boot sequence. If the flash loaded bits are programmed to be enabled (set), these bits cannot be disabled (cleared).

Offset: 0x0044 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	REE11_ADL1	REE11_ADL0	0	REE11_IOL1	REE11_OSC	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

Figure 1035. Reset Event Enable Register (REE_HV0)

Table 1000. REE_HV0 field descriptions

Field	Description
1 REE11_ADL1	LVD11_ADL1 reset enable This bit defines whether LVD290_ADL1 VD, the voltage level detect for the high voltage assertion on the ADC1 supply generates a system reset. The REE_LV0.LVD11_AC bit determines whether a system reset is enabled or not. 0 Disabled. HVD assertion on the supply of the cold point does not cause system reset 1 Enabled. HVD assertion on the supply of the cold point causes system reset
2 REE11_ADL0	LVD11_ADL0 reset enable This bit defines whether LVD290_ADL0 VD, the voltage level detect for the high voltage assertion on the ADC0 supply generates a system reset. The REE_LV0.LVD11_AC bit determines whether a system reset is enabled or not. 0 Disabled. LVD assertion on the supply of the cold point does not cause system reset 1 Enabled. LVD assertion on the supply of the cold point causes system reset
4 REE11_IOL1	LVD11_IOL1 reset enable This bit defines whether LVD290_IOL1 VD, the voltage level detect for the high voltage assertion on the IO1 supply generates a system reset. The REE_LV0.LVD11_AC bit determines whether a system reset is enabled or not. 0 Disabled. LVD assertion on the supply of the cold point does not cause system reset 1 Enabled. LVD assertion on the supply of the cold point causes system reset
5 REE11_OSC	LVD11_OSC reset enable This bit defines whether LVD290_OSC VD, the voltage level detect for the high voltage assertion on the XOSC supply generates a system reset. The REE_LV0.LVD11_OSC bit determines whether a system reset is enabled or not. 0 Disabled. LVD assertion on the supply of the cold point does not cause system reset 1 Enabled. LVD assertion on the supply of the cold point causes system reset

50.5.13 Reset Event Select Register (RES_HV0)

This Reset Event Select Register, in HIGH VOLTAGE 0 range, controls whether the voltage detect signal event causes a destructive or functional reset. If the desired flag bit is enabled, the voltage detect event causes either a destructive or functional reset to be generated when the selected voltage passes the trigger event. These bits are loaded from the flash during the boot sequence.

Offset: 0x0048 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	RES11_ADL1	RES11_ADL0	0	RES11_IOL1	RES11_OSC	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

Figure 1036. Reset Event Select Register (RES_HV0)

Table 1001. RES_HV0 field descriptions

Field	Description
1 RES11_ADL1	RES11_ADL1 reset event select This bit defines whether an LVD assertion on the ADC1 supply of the cold point generates a destructive reset or a functional reset. The RES_HV0.RES11_ADL1 bit determines whether the reset event is enabled. 0 Destructive Reset Generated. LVD assertion on the supply of the cold point causes a destructive system reset 1 Functional Reset Generated. LVD assertion on the supply of the cold point causes a functional system reset
2 RES11_ADL0	RES11_ADL0 reset event select This bit defines whether an LVD assertion on the ADC0 supply of the cold point generates a destructive reset or a functional reset. The RES_HV0.RES11_ADL0 bit determines whether the reset event is enabled. 0 Destructive Reset Generated. LVD assertion on the supply of the cold point causes a destructive system reset 1 Functional Reset Generated. LVD assertion on the supply of the cold point causes a functional system reset

Table 1001. RES_HV0 field descriptions (continued)

Field	Description
4 RES11_IOL1	RES11_IOL1 reset event select This bit defines whether an LVD assertion on the IO1 supply of the cold point generates a destructive reset or a functional reset. The RES_HV0.RES11_IOL1 bit determines whether the reset event is enabled. 0 Destructive Reset Generated. LVD assertion on the supply of the cold point causes a destructive system reset 1 Functional Reset Generated. LVD assertion on the supply of the cold point causes a functional system reset
5 RES11_OSC	RES11_OSC reset event select This bit defines whether an LVD assertion on the XOSC supply of the cold point generates a destructive reset or a functional reset. The RES_HV0.RES11_OSC bit determines whether the reset event is enabled. 0 Destructive Reset Generated. LVD assertion on the supply of the cold point causes a destructive system reset 1 Functional Reset Generated. LVD assertion on the supply of the cold point causes a functional system reset

50.5.14 Interrupt Enable (IE_HV0)

This configuration register contains a set of HV0 interrupt Enable bits that correspond to each of the HV0 Event Pending Registers (EPR_HV0). These bits indicate whether an interrupt occurs when the voltage event is seen. A ‘0’ indicates that no interrupt is to be sent, and a ‘1’ indicates that an interrupt is to occur when the voltage detect event occurs. These bits are replicated as a read-only monitor in the register (IE_G) at address 0088h. The IE_HV0 bits can be read or written at any time.

Offset: 0x004C Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	LVD11IE_ADL1	LVD11IE_ADL0	0	LVD11IE_IOL1	LVD11IE_OSC	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

Figure 1037. Interrupt Enable Register (IE_HV0)

Table 1002. IE_HV0 field descriptions

Field	Description
1 LVD11IE_ADL1	LVD11IE_ADL1 Interrupt Enable This bit determines whether an interrupt is seen by the system when LVD290_ADL1VD, the voltage level detect for the high voltage assertion on the ADC1 supply event occurs. 0 No interrupt occurs 1 An interrupt occurs
2 LVD11IE_ADL0	LVD11IE_ADL0 Interrupt Enable This bit determines whether an interrupt is seen by the system when LVD290_ADL0 VD, the voltage level detect for the high voltage assertion on the ADC0 supply event occurs. 0 No interrupt occurs 1 An interrupt occurs
4 LVD11IE_IOL1	LVD11IE_IOL1 Interrupt Enable This bit determines whether an interrupt is seen by the system when LVD290_IOL1 VD, the voltage level detect for the high voltage assertion on the IO1 supply event occurs. 0 No interrupt occurs 1 An interrupt occurs
5 LVD11IE_OSC	LVD11IE_OSC Interrupt Enable This bit determines whether an interrupt is seen by the system when LVD290_OSC VD, the voltage level detect for the high voltage assertion on the XOSC supply event occurs. 0 No interrupt occurs 1 An interrupt occurs

50.5.15 FCCU Event Enable Register (FEE_HV0)

This FCCU Event Enable Register HV0 controls whether the voltage detect signal events cause an event signal to be sent to the FCCU (Fault Control Unit). If the desired flag bit is enabled, the voltage detect event causes an FCCU event to be generated when the selected voltage passes the trigger event.

Offset: 0x0050 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	FEE11_ADL1	FEE11_ADL0	FEE11_FL	FEE11_IOL1	FEE11_OSC	0	FEE11_C	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

Figure 1038. FCCU Event Enable Register (FEE_HV0)

Table 1003. FEE_HV0 field descriptions

Field	Description
1 FEE11_ADL1	FEE11_ADL1 FCCU event enable This bit defines whether a LVD290_ADL1 VD, the voltage level detect for the high voltage assertion on the ADC1 2.90V supply assertion generates an FCCU event. 0 Disabled. VD assertion of HV0 on the supply of the ADC1 does not cause an FCCU event 1 Enabled. VD assertion of HV0 on the supply of the ADC1 causes an FCCU event
2 FEE11_ADL0	FEE11_ADL0 FCCU event enable This bit defines whether a LVD290_ADL0 VD, the voltage level detect for the high voltage assertion on the ADC0 2.90V supply assertion generates an FCCU event. 0 Disabled. VD assertion of HV0 on the supply of the ADC0 does not cause an FCCU event 1 Enabled. VD assertion of HV0 on the supply of the ADC0 causes an FCCU event
3 FEE11_FL	FEE11_FL FCCU event enable This bit defines whether a LVD290_FL VD, the voltage level detect for the high voltage assertion on the FLASH 2.90V supply assertion generates an FCCU event. 0 Disabled. VD assertion of HV0 on the supply of the FLASH does not cause an FCCU event 1 Enabled. VD assertion of HV0 on the supply of the FLASH causes an FCCU event
4 FEE11_IOL1	FEE11_IOL1 FCCU event enable This bit defines whether a LVD290_IOL1 VD, the voltage level detect for the high voltage assertion on the IO1 2.90V supply assertion generates an FCCU event. 0 Disabled. VD assertion of HV0 on the supply of the IO1 does not cause an FCCU event 1 Enabled. VD assertion of HV0 on the supply of the IO1 causes an FCCU event

Table 1003. FEE_HV0 field descriptions (continued)

Field	Description
5 FEE11_OSC	FEE11_OSC FCCU event enable This bit defines whether a LVD290_OSC VD, the voltage level detect for the high voltage assertion on the XOSC 2.90V supply assertion generates an FCCU event. 0 Disabled. VD assertion of HV0 on the supply of the XOSC does not cause an FCCU event 1 Enabled. VD assertion of HV0 on the supply of the XOSC causes an FCCU event
7 FEE11_C	FEE11_C FCCU event enable This bit defines whether a LVD290_C VD, the voltage level detect for the high voltage assertion on the CORE/PMU 2.90V supply assertion generates an FCCU event. 0 Disabled. an VD assertion of HV0 on the supply of the CORE does not cause an FCCU event 1 Enabled. VD assertion of HV0 on the supply of the CORE causes an FCCU event

50.5.16 Event Pending Register (EPR_HV1)

This Event Pending Register indicates the present and past state of the voltage detect signals, in HIGH VOLTAGE1 range. If the voltage detect event has ever passed the trigger event since the last clearing event, the flag bit is set. In order to clear the flag, a one must be written to the flag bit that needs to be cleared.

Offset: 0x0060 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	LVD14_ADL1	LVD14_ADL0	0	LVD14_IOL1	LVD14_IOL0	0	0
W										w1c	w1c		w1c	w1c		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	HVD13_C	0	0	0	0	0	0	0	0
W								w1c								
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1039. Event Pending Register (EPR_HV1)

Table 1004. EPR_HV1 field descriptions

Field	Description
9 LVD14_ADL1	LVD14_ADL1 flag This bit is the High voltage status flag associated with LVD400_ADL1 VD, the VD for the high voltage ADC1 4.00V supply. It is asserted when the supply falls below its corresponding Voltage Detector 14 threshold, and clears when the supply rises under its corresponding LVD14_ADL1 threshold and a one is written to this bit location. 0 Currently no occurrence 1 VD occurrence detected on the high voltage 4.00V point supply
10 LVD14_ADL0	LVD14_ADL0 flag This bit is the High voltage status flag associated with LVD400_ADL0 VD, the VD for the high voltage ADC0 4.00V supply. It is asserted when the supply falls below its corresponding Voltage Detector 14 threshold, and clears when the supply rises under its corresponding LVD14_ADL0 threshold and a one is written to this bit location. 0 Currently no occurrence 1 VD occurrence detected on the high voltage 4.00V point supply
12 LVD14_IOL1	LVD14_IOL1 flag This bit is the High voltage status flag associated with LVD400_IOL1 VD, the VD for the high voltage IO1 4.00V supply. It is asserted when the supply falls below its corresponding Voltage Detector 14 threshold, and clears when the supply rises under its corresponding LVD14_IOL1 threshold and a one is written to this bit location. 0 Currently no occurrence 1 VD occurrence detected on the high voltage 4.00V point supply
13 LVD14_IOL0	LVD14_IOL0 flag This bit is the High voltage status flag associated with LVD400_IOL0 VD, the VD for the high voltage IO0 PMU 4.00V supply. It is asserted when the supply falls below its corresponding Voltage Detector 14 threshold, and clears when the supply rises under its corresponding LVD14_IOL0 threshold and a one is written to this bit location. 0 Currently no occurrence 1 VD occurrence detected on the high voltage 4.00V point supply
23 HVD13_C	HVD13_C flag This bit is the High Voltage status flag associated with HVD400_C VD, the VD for the high voltage CORE 4.00V supply. It is asserted when the supply falls below its corresponding Voltage Detector 13 threshold, and clears when the supply rises above its corresponding HVD13 threshold and a one is written to this bit location. 0 Currently no occurrence 1 VD occurrence detected on the high voltage 4.00V point supply

50.5.17 Reset Event Enable Register (REE_HV1)

This Reset Event Enable Register controls, in HIGH VOLTAGE 1 range, whether the voltage detect signal event causes a reset. If the desired flag bit is enabled, the voltage detect event causes a reset to be generated when the selected voltage passes the trigger event. These bits are loaded from the flash during the boot sequence. If the flash loaded bits are programmed to be enabled (set), these bits cannot be disabled (cleared).

Offset: 0x0064 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	REE14_ADL1	REE14_ADL0	0	REE14_IOL1	REE14_IOL0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	REE13_C	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1040. Reset Event Enable Register (REE_HV1)

Table 1005. REE_LV1 field descriptions

Field	Description
9 REE14_ADL1	REE14_ADL1 reset enable This bit defines whether an LVD assertion on the high voltage supply of the cold point generates a system reset. The RES_HV1.REE14_ADL1 bit determines whether the reset is functional or destructive. 0 Disabled. LVD assertion on the supply of the cold point does not cause system reset 1 Enabled. LVD assertion on the supply of the cold point causes system reset
10 REE14_ADL0	REE14_ADL0 reset enable This bit defines whether an LVD assertion on the high voltage supply of the cold point generates a system reset. The RES_HV1.REE14_ADL0 bit determines whether the reset is functional or destructive. 0 Disabled. LVD assertion on the supply of the cold point does not cause system reset 1 Enabled. LVD assertion on the supply of the cold point causes system reset
12 REE14_IOL1	REE14_IOL1 reset enable This bit defines whether an LVD assertion on the high voltage supply of the cold point generates a system reset. The RES_HV1.REE14_IOL1 bit determines whether the reset is functional or destructive. 0 Disabled. LVD assertion on the supply of the cold point does not cause system reset 1 Enabled. LVD assertion on the supply of the cold point causes system reset

Table 1005. REE_LV1 field descriptions (continued)

Field	Description
13 REE14_IOL0	REE14_IOL0 reset enable This bit defines whether an LVD assertion on the high voltage supply of the cold point generates a system reset. The RES_HV1.REE14_IOL0 bit determines whether the reset is functional or destructive. 0 Disabled. LVD assertion on the supply of the cold point does not cause system reset 1 Enabled. LVD assertion on the supply of the cold point causes system reset
23 REE13_C	REE13_C reset enable This bit defines whether HVD400_C VD, the VD for the high voltage CORE 4.00V supply generates a system reset. The RES_HV1.HVD13_C bit determines whether the reset is functional or destructive. 0 Disabled. HVD assertion on the supply of the cold point does not cause system reset 1 Enabled. HVD assertion on the supply of the cold point causes system reset

50.5.18 Reset Event Select Register (RES_HV1)

This Reset Event Select Register controls, in HIGH VOLTAGE 1 range, whether the voltage detect signal event causes a destructive or functional reset. If the desired flag bit is enabled, the voltage detect event causes either a destructive or functional reset to be generated when the selected voltage passes the trigger event. These bits are loaded from the flash during the boot sequence.

Offset: 0x0068 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	RES14_ADL1	RES14_ADL0	0	RES14_IOL1	RES14_IOL0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	RES13_C	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1041. Reset Event Select Register (RES_HV1)

Table 1006. RES_HV1 field descriptions

Field	Description
9 RES14_ADL1	RES14_ADL0 reset event select This bit defines whether a LVD400_ADL1 VD, the VD for the high voltage ADC1 4.00V supply generates a destructive reset or a functional reset by LVD14_ADL1 bit. 0 Destructive Reset Generated. LVD assertion on the supply of the cold point causes a destructive system reset 1 Functional Reset Generated. LVD assertion on the supply of the cold point causes a functional system reset
10 RES14_ADL0	RES14_ADL0 reset event select This bit defines whether a LVD400_ADL0 VD, the VD for the high voltage ADC0 4.00V supply generates a destructive reset or a functional reset by LVD14_ADL0 bit. 0 Destructive Reset Generated. LVD assertion on the supply of the cold point causes a destructive system reset 1 Functional Reset Generated. LVD assertion on the supply of the cold point causes a functional system reset
12 RES14_IOL1	RES14_IOL1 reset event select This bit defines whether a LVD400_IOL1 VD, the VD for the high voltage IO1 4.00V supply generates a destructive reset or a functional reset by LVD14_IOL1 bit. 0 Destructive Reset Generated. LVD assertion on the supply of the cold point causes a destructive system reset 1 Functional Reset Generated. LVD assertion on the supply of the cold point causes a functional system reset
13 RES14_IOL0	RES14_IOL0 reset event select This bit defines whether a LVD400_IOL0 VD, the VD for the high voltage IO0 PMU 4.00V supply generates a destructive reset or a functional reset by LVD14_IOL0 bit. 0 Destructive Reset Generated. LVD assertion on the supply of the cold point causes a destructive system reset 1 Functional Reset Generated. LVD assertion on the supply of the cold point causes a functional system reset
23 RES13_C	RES13_C reset event select This bit defines whether a HVD400_C VD, the VD for the high voltage CORE 4.00V supply generates a destructive reset or a functional reset by HVD13_C bit. 0 Destructive Reset Generated. HVD assertion on the supply of the cold point causes a destructive system reset 1 Functional Reset Generated. HVD assertion on the supply of the cold point causes a functional system reset

50.5.19 Interrupt Enable Register (IE_HV1)

This configuration register contains a set of HV1 interrupt Enable bits that correspond to each of the HV1 Event Pending Registers (EPR_HV1). These bits indicate whether an interrupt occurs when the voltage event is seen. A ‘0’ indicates that no interrupt is to be sent, and a ‘1’ indicates that an interrupt is to occur when the voltage detect event occurs. These bits are replicated as a read-only monitor in the register (IE_G) at address 0088h. The IE_HV1 bits can be read or written at any time.

Offset: 0x006C Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	LVDIE14_ADL1	LVDIE14_ADL0	0	LVDIE14_IOL1	LVDIE14_IOL0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	HVDIE13_C	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1042. Interrupt Enable Register (IE_HV1)

Table 1007. IE_HV1 field descriptions

Field	Description
9 LVDIE14_ADL1	LVDIE14_ADL1 Interrupt Enable This bit determines whether an interrupt is seen by the system when LVD400_ADL1 VD, the VD for the high voltage ADC1 4.00V supply event occurs. 0 No interrupt occurs 1 An interrupt occurs
10 LVDIE14_ADL0	LVDIE14_ADL0 Interrupt Enable This bit determines whether an interrupt is seen by the system when LVD400_ADL0 VD, the VD for the high voltage ADC0 4.00V supply event occurs. 0 No interrupt occurs 1 An interrupt occurs
12 LVDIE14_IOL1	LVDIE14_IOL1 Interrupt Enable This bit determines whether an interrupt is seen by the system when LVD400_IOL1 VD, the VD for the high voltage IO1 4.00V supply event occurs. 0 No interrupt occurs 1 An interrupt occurs
13 LVDIE14_IOL0	LVDIE14_IOL0 Interrupt Enable. This bit determines whether an interrupt is seen by the system when LVD400_IOL0 VD, the VD for the high voltage IO0 PMU 4.00V supply event occurs. 0 No interrupt occurs 1 An interrupt occurs
23 HVDIE13_C	HVDIE13_C Interrupt Enable. This bit determines whether an interrupt is seen by the system when HVD400_C VD, the VD for the high voltage CORE 4.00V supply event occurs. 0 No interrupt occurs 1 An interrupt occurs

50.5.20 FCCU Event Enable Register (FEE_HV1)

This FCCU Event Enable Register HV1 controls whether the voltage detect signal events cause an event signal to be sent to the FCCU (Fault Control Unit). If the desired flag bit is enabled, the voltage detect event causes an FCCU event to be generated when the selected voltage passes the trigger event.

Offset: 0x0070 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	FEE14_ADL1	FEE14_ADL0	0	FEE14_IOL1	FEE14_IOL0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	FEE13_C	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1043. FCCU Event Enable Register (FEE_HV1)

Table 1008. FEE_HV1 field descriptions

Field	Description
9 FEE14_ADL1	FEE14_ADL1 FCCU event enable This bit defines whether an VD assertion of LV14 on lvd400_ADL1 for the supply of the ADC1 generates an FCCU event. 0 Disabled. VD assertion of LV14 on the supply of the CORE does not cause an FCCU event 1 Enabled. VD assertion of LV14 on the supply of the CORE causes an FCCU event
10 FEE14_ADL0	FEE14_ADL0 FCCU event enable This bit defines whether an VD assertion of LV14 on lvd400_ADL0 for the supply of the ADC0 generates an FCCU event. 0 Disabled. VD assertion of LV14on the supply of the CORE does not cause an FCCU event 1 Enabled. VD assertion of LV14 on the supply of the CORE causes an FCCU event
12 FEE14_IOL1	FEE14_IOL1 FCCU event enable This bit defines whether an VD assertion of LV14 on lvd400_IOL1 for the supply of the IO1 generates an FCCU event. 0 Disabled. VD assertion of LV14 on the supply of the CORE does not cause an FCCU event 1 Enabled. VD assertion of LV14 on the supply of the CORE causes an FCCU event

Table 1008. FEE_HV1 field descriptions (continued)

Field	Description
13 FEE14_IOL0	FEE14_IOL0 FCCU event enable This bit defines whether an VD assertion of LV14 on lvd400_IOL0 for the supply of the IO0 PMU generates an FCCU event. 0 Disabled. VD assertion of LV14 on the supply of the CORE does not cause an FCCU event 1 Enabled. VD assertion of LV14 on the supply of the CORE causes an FCCU event
23 FEE13_C	FEE13_C FCCU event enable This bit defines whether an VD assertion of HV14 on hvd400_c for the supply of the CORE generates an FCCU event. 0 Disabled. VD assertion of HV14on the supply of the CORE does not cause an FCCU event 1 Enabled. VD assertion of HV14 on the supply of the CORE causes an FCCU event

50.5.21 Supply Gauge Status Register (*GR_S*)

This status register contains the gauge of the indicated supply. This indicates the present state of the voltage detect events for each of the supplies. Each of the voltage detect signals for a given voltage are combined to form a single bit, see *Table 1009*.

Offset: 0x0080 Access: User read-only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	VD14_ADL1	VD14_ADL0	VD14_IOL1	VD14_IOL0	VD13_C	VD11_ADL1	VD11_ADL0	VD11_FL	VD11_IOL1	VD11_OSC	VD11_C	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	VD6_C	0	0	0	0	0	VD3_C	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1044. Supply Gauge Status Register (*GR_S*)

Table 1009. *GR_S* field descriptions

Field	Description
1 VD14_ADL1	VD14_ADL1 low-voltage detect flag This read-only bit is the low-voltage status flag associated with the voltage level detect lvd400_adl1 for the high voltage 4 V ADCL1 supplies. It is asserted when ADCL1 supplies rises above the corresponding LVD threshold, and clears when ADCL1 supplies falls below the corresponding LVD threshold. 0 Currently no occurrence 1 LVD occurrence detected on the high voltage 4.00 V supplies
2 VD14_ADL0	VD14_ADL0 low-voltage detect flag This read-only bit is the low-voltage status flag associated with the voltage level detect lvd400_adl0 for the high voltage 4 V ADCL0 supplies. It is asserted when ADCL0 supplies rises above the corresponding LVD threshold, and clears when ADCL0 supplies falls below the corresponding LVD threshold. 0 Currently no occurrence 1 LVD occurrence detected on the high voltage 4.00 V supplies
3 VD14_IOL1	VD14_IOL1low-voltage detect flag This read-only bit is the low-voltage status flag associated with the voltage level detect lvd400_iol1 for the high voltage 4 V IO1 supplies. It is asserted when IO1 supplies rises above the corresponding LVD threshold, and clears when IO1 supplies falls below the corresponding LVD threshold. 0 Currently no occurrence 1 LVD occurrence detected on the high voltage 4.00 V supplies
4 VD14_IOL0	VD14_IOL0 low-voltage detect flag This read-only bit is the low-voltage status flag associated with the voltage level detect lvd400_iol0 for the high voltage 4 V IO0 supplies. It is asserted when IO0 supplies rises above the corresponding LVD threshold, and clears when IO0 supplies falls below the corresponding LVD threshold. 0 Currently no occurrence 1 LVD occurrence detected on the high voltage 4.00 V supplies
5 VD13_C	VD13_C high-voltage detect flag This read-only bit is the high-voltage status flag associated with the voltage level detect hvd400_c for the high voltage 4 V CORE supply. It is asserted when the CORE supplies falls below its corresponding HVD threshold, and clears when the supplies rises above its corresponding HVD threshold. 0 Currently no occurrence. 1 HVD occurrence detected on the high voltage 4.00 V supply
6 VD11_ADL1	VD11_ADL1 low-voltage detect flag This read-only bit is the low-voltage status flag associated with the voltage level detect lvd290_adl1 for the high voltage 2.9 V ADC1 supplies . It is asserted when ADC1 supplies falls below the corresponding LVD threshold, and clears when ADC1 supplies rises above the corresponding LVD threshold. 0 Currently no occurrence 1 LVD occurrence detected on the high voltage 2.9 V supplies

Table 1009. *GR_S* field descriptions (continued)

Field	Description
7 VD11_ADL0	VD11_ADL0 low-voltage detect flag This read-only bit is the low-voltage status flag associated with the voltage level detect lvd290_adl0 for the high voltage 2.9 V ADC0 supplies . It is asserted when ADC0 supplies falls below the corresponding LVD threshold, and clears when ADC0 supplies rises above the corresponding LVD threshold. 0 Currently no occurrence 1 LVD occurrence detected on the high voltage 2.9 V supplies
8 VD11_FL	VD11_FL low-voltage detect flag This read-only bit is the low-voltage status flag associated with the voltage level detect lvd290_fl for the high voltage 2.9 V FLASH supplies. It is asserted when any of the FLASH supplies falls below the corresponding LVD threshold, and clears when all the FLASH supplies rise above the corresponding LVD threshold. 0 Currently no occurrence 1 LVD occurrence detected on the high voltage 2.9 V supplies
9 VD11_IOL1	VD11_IOL1 low-voltage detect flag This read-only bit is the low-voltage status flag associated with the voltage level detect lvd290_iol1 for the high voltage 2.9 V IO1 supply . It is asserted when IO1 supplies falls below the corresponding LVD threshold, and clears when IO1 supplies rise above the corresponding LVD threshold. 0 Currently no occurrence 1 LVD occurrence detected on the high voltage 2.9 V supplies
10 VD11_OSC	VD11_OSC low-voltage detect flag This read-only bit is the low-voltage status flag associated with the voltage level detect lvd290_OSC for the high voltage 2.9 V OSC supply . It is asserted when OSC supplies falls below the corresponding LVD threshold, and clears when OSC supplies rise above the corresponding LVD threshold. 0 Currently no occurrence 1 LVD occurrence detected on the high voltage 2.9 V supplies
11 VD11_C	VD11_C low-voltage detect flag This read-only bit is the low-voltage status flag associated with the voltage level detect lvd290_c for the high voltage 2.9 V CORE supplies. It is asserted when any of the CORE supplies falls below the corresponding LVD threshold, and clears when all the CORE supplies rise above the corresponding LVD threshold. 0 Currently no occurrence 1 LVD occurrence detected on the high voltage 2.9 V supplies

Table 1009. *GR_S* field descriptions (continued)

Field	Description
19 VD6_C	VD6_C high-voltage detect flag This read-only bit is the high-voltage status flag associated with the voltage level detect hvd140_c for the high voltage 1.40V CORE supplies. It is asserted when any of the CORE supplies rises above the corresponding HVD threshold, and clears when all the CORE supplies fall below the corresponding HVD threshold. 0 Currently no occurrence 1 HVD occurrence detected on the high voltage 1.40 V supply
25 VD3_C	VD3_C low-voltage detect flag This read-only bit is the low-voltage status flag associated with the voltage level detect lvd108_c for the high voltage 1.08 V CORE supplies. It is asserted when any of the CORE supplies falls below the corresponding LVD threshold, and clears when all the CORE supplies rise above the corresponding LVD threshold. 0 Currently no occurrence 1 LVD occurrence detected on the high voltage 1.08 V supplies

50.5.22 Pending Gauge Status Register (GR_P)

This configuration register contains a mirror of the contents of the Event Pending Registers, but in a format that includes one bit for each voltage detect signal. These bits indicate the state of the voltage detect events including whether a voltage level event has occurred at any time in the past but has not been cleared (via the EPR register). These bits are read only, and they are cleared by writing a one to the corresponding EPR register.

The Pending registers are only set on an active edge of an event, so a transition must be seen before they will become set.

The raw analog block voltage detect signals are also synchronized to the bus clock before being used for the pending registers.

Offset: 0x0084 Access: User read-only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	VD14_ADL1	VD14_ADL0	VD14_IOL1	VD14_IOL0	VD13_C	VD11_ADL1	VD11_ADL0	VD11_FL	VD11_IOL1	VD11_OSC	VD11_C	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	VD6_C	0	0	0	0	0	VD3_C	0	0	0	0	IRQ_ST	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1045. Pending Gauge Status Register (GR_P)

Table 1010. GR_P field descriptions

Field	Description
1 VD14_ADL1	VD14_ADL1 Pending Monitor. Read-only register.
2 VD14_ADL0	VD14_ADL0 Pending Monitor. Read-only register.
3 VD14_IOL1	VD14_IOL1 Pending Monitor. Read-only register.
4 VD14_IOL0	VD14_IOL0 Pending Monitor. Read-only register.
5 VD13_C	VD13_C Pending Monitor. Read-only register.
6 VD11_ADL1	VD11_ADI1 Pending Monitor. Read-only register.
7 VD11_ADL0	VD11_ADI0 Pending Monitor. Read-only register.
8 VD11_FL	VD11_FL Pending Monitor. Read-only register.
9 VD11_IOL1	VD11_IOI1 Pending Monitor. Read-only register.
10 VD11_OSC	VD11_OSC Pending Monitor. Read-only register.
11 VD11_C	VD11_C Pending Monitor. Read-only register.
19 VD6_C	VD6_C Pending Monitor. Read-only register.
25 VD3_C	VD3_C Pending Monitor. Read-only register.
30 IRQ_ST	IRQ_ST user BIST Pending Monitor. Read-only register.

50.5.23 Interrupt Enable Pending Register (IE_G)

This configuration register contains a set of Interrupt Enable bits that correspond to each of the Event Pending Registers (EPR_XX), but in a format similar to the Pending Gauge Status register (GR_P). These bits indicate whether an interrupt occurs when the voltage event is seen. A ‘0’ indicates that no interrupt is to be sent, and a ‘1’ indicates that an interrupt is to occur when the voltage detect event occurs. These bits are enabled via a write of ‘1’ to the MSB of this register (IE_EN). After this, these bits can be read or written at any time.

Offset: 0x0088 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	IE_EN	0	VDIE14_ADL1	VDIE14_ADL0	VDIE14_IOL1	VDIE14_IOL0	VDIE13_C	VDIE11_ADL1	VDIE11_ADL0	VDIE11_FL	VDIE11_IOL1	VDIE11_OSC	VDIE11_C	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	VDIE6_C	0	0	0	0	0	VDIE3_C	0	0	0	0	IRQ_EN	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1046. Interrupt Enable Global Register (IE_G)

Table 1011. IE_G field descriptions

Field	Description
0 IE_EN	IE_EN. Interrupt Enable. This bit permits Interrupt Enable bit to be written. 0 No interrupt enables can be written 1 Any interrupt enable can be written
2 VDIE14_ADL1	VDIE14_ADL1 enable interrupt monitor. Read-only register.
3 VDIE14_ADL0	VDIE14_ADL0 enable interrupt monitor. Read-only register.
4 VDIE14_IOL1	VDIE14_IOL1 enable interrupt monitor. Read-only register.
5 VDIE14_IOL0	VDIE14_IOL0 enable interrupt monitor. Read-only register.
6 VDIE13_C	VDIE13_C enable interrupt monitor. Read-only register.
7 VDIE11_ADL1	VDIE11_ADL1 enable interrupt monitor. Read-only register.
8 VDIE11_ADL0	VDIE11_ADL0 enable interrupt monitor. Read-only register.
9 VDIE11_FL	VDIE11_FL enable interrupt monitor. Read-only register.
10 VDIE11_IOL1	VDIE11_IOL1 enable interrupt monitor. Read-only register.
11 VDIE11_OSC	VDIE11_OSC enable interrupt monitor. Read-only register.

Table 1011. IE_G field descriptions (continued)

Field	Description
12 VDIE11_C	VDIE11_C enable interrupt monitor. Read-only register.
19 VDIE6_C	VDIE6_C enable interrupt monitor. Read-only register.
25 VDIE3_C	VDIE3_C enable interrupt monitor. Read-only register.
30 IRQ_EN	IRQ_EN user BIST enable interrupt monitor. Read-only register.

50.5.24 Voltage Selection of IO (VSIO)

This register controls the pad operation in the different I/O supply segments. The configuration bits have to be set accordingly to the supply voltage applied to the I/O segments in order to get the correct behavior of the input and output buffers.

Offset: 0x0104 Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	VSIO_X	VSIO_AD1	VSIO_AD0	VSIO_IM1	VSIO_IM0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1047. Voltage Selection of IO (VSIO)

Table 1012. VSIO field descriptions

Field	Description
27 VSIO_X	VSIO_X Selects between 3.3/5V for IO XOSC ring. 0 Ring will operate at 3.3v 1 Ring will operate at 5v
28 VSIO_AD1	VSIO_AD1 Selects between 3.3/5V for IO ADC ring1. 0 Ring will operate at 3.3v 1 Ring will operate at 5v
29 VSIO_AD0	VSIO_AD0 Selects between 3.3/5V for IO ADC ring0. 0 Ring will operate at 3.3v 1 Ring will operate at 5v

Table 1012. VSIO field descriptions (continued)

Field	Description
30 VSIO_IM1	VSIO_IM1 Selects between 3.3/5V for IO main ring1. 0 Ring will operate at 3.3v 1 Ring will operate at 5v
31 VSIO_IM0	VSIO_IM0 Selects between 3.3/5V for IO main ring0. 0 Ring will operate at 3.3v 1 Ring will operate at 5v

50.5.25 IPS Regulator Control PMOS NMOS register (IREGU_CTL_PN)

The IPS Regulator Control PMOS NMOS register enables both PMOS and NMOS regulator and enables the PMOS NMOS regulator in external regulator mode too.

Offset: 0x0240 Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	REG_MODE	DREG_ENB	FORCEON_PN	PREG_EN	NREG_EN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0/1	0/1	0	0	0

Figure 1048. PMOS NMOS Regulator Control Register (IREGU_CTL_PN)

Table 1013. IREGU_CTL_PN field descriptions

Field	Description
27 REG_MODE	REG_MODE is a status bit for external/internal regulator mode enable. 0 Means internal regulator mode is enabled 1 Means external regulator mode is enabled **Note:** Reset value of this bit depends on voltage applied on PWMODE pad. When PWMODE pad is grounded, reset value of this bit is 1 and it is 0 when PWMODE pad is connected to VDD.
28 DREG_ENB	DREG_ENB is a status bit for didgital regulator enable. 0 Means digital regulator is enabled 1 Means digitall regulator is disabled **Note:** Reset value of this bit depends on voltage applied on PWMODE pad. When PWMODE pad is grounded, reset value of this bit is 1 and it is 0 when PWMODE pad is connected to VDD.

Table 1013. IREGU_CTL_PN field descriptions (continued)

Field	Description
29 FORCEON_PN	FORCEON_PN bit enables pmos and nmos regulators in external regulator mode. 0 Means both Pmos and Nmos regulators are disabled 1 Means Pmos/Nmos regulators enabling depends on PREG_EN and NREG_EN bits This bit has no effect in Internal regulator mode.
30 PREG_EN	PREG_EN is the Pmos Regulator Enabling bit. 0 Means Pmos regulator is enabled 1 Means Pmos regulator is disabled
31 NREG_EN	NREG_EN is the Nmos Regulator Enabling bit. 0 Means Nmos regulator is enabled 1 Means Nmos regulator is disabled

Table 1014. IREGU_CTL_PN - PMOS/NMOS Regulator control configuration

Regulator Mode	FORCEON_PN	PREG_EN	NREG_EN	PREG/NREG Status
External	0	X	X	Both disabled.
	1	0	0	Both enabled
	1	0	1	PREG enabled
	1	1	0	NREG enabled
	1	1	1	Both disabled
Internal	X	0	0	Both enabled
	X	0	1	PREG enabled
	X	1	0	NREG enabled
	X	1	1	Both disabled

50.5.26 Event Pending Register (EPR_TD)

This Event Pending Register indicates the present and past state of the temperature sensor signals. If the temperature event has ever passed the trigger event since the last clearing event, the flag bit is set. In order to clear the flag, a one must be written to the flag bit that needs to be cleared.

Offset: 0x0300 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	TEMP1_1	TEMP1_0
W															w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	TEMP0_1	TEMP0_0
W															w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1049. Event Pending Register (EPR_TD)

Table 1015. EPR_TD field descriptions

Field	Description
14 TEMP1_1	TEMP1_1 flag This bit is the temperature status flag associated with the temperature for the temperature sensor point (150 °C) for temperature sensor 1. It is asserted when the temperature exceeds its corresponding threshold, and cleared by the user by writing ‘1’ in this bit location when the temperature falls below its corresponding threshold. If the IETD_1 bit is also asserted, an interrupt is sent to the CPU. If REE_TD[TEMP_1] is also asserted, a system reset is generated, which clears IETD_1 and negate the interrupt request. If REE_TD[TEMP_1] is asserted and a destructive reset is selected (via RES_TD[TEMP_1] being 0), then a destructive reset is generated. If RES_TD[TEMP_1] is set, then a functional reset is generated. 0 Currently no occurrence 1 Temperature occurrence detected
15 TEMP1_0	TEMP1_0 flag This bit is the temperature status flag associated with the temperature for the *cold* temperature sensor point (-40 °C) for temperature sensor 1. It is asserted when the temperature exceeds its corresponding threshold, and cleared by the user by writing ‘1’ in this bit location when the temperature falls below its corresponding threshold. If the IETD1_0 bit is also asserted, an interrupt is sent to the CPU. If REE_TD[TEMP1_0] is also asserted, a system reset is generated, which clears IETD1_0 and negate the interrupt request. If REE_TD[TEMP1_0] is asserted and a destructive reset is selected (via RES_TD[TEMP1_0] being 0), then a destructive reset is generated. If RES_TD[TEMP1_0] is set, then a functional reset is generated. 0 Currently no occurrence 1 Temperature occurrence detected

Table 1015. EPR_TD field descriptions (continued)

Field	Description
30 TEMP0_1	TEMP0_1 flag This bit is the temperature status flag associated with the temperature for the temperature sensor point (150 °C) for temperature sensor 0. It is asserted when the temperature exceeds its corresponding threshold, and cleared by the user by writing '1' in this bit location when the temperature falls below its corresponding threshold. If the IETD0_1 bit is also asserted, an interrupt is sent to the CPU. If REE_TD[TEMP0_1] is also asserted, a system reset is generated, which clears IETD0_1 and negate the interrupt request. If REE_TD[TEMP0_1] is asserted and a destructive reset is selected (via RES_TD[TEMP0_1] being 0), then a destructive reset is generated. If R ES_TD[TEMP0_1] is set, then a functional reset is generated. 0 Currently no occurrence 1 Temperature occurrence detected
31 TEMP0_0	TEMP0_0 flag This bit is the temperature status flag associated with the temperature for the *cold* temperature sensor point (-40 °C) for temperature sensor 0. It is asserted when the temperature exceeds its corresponding threshold, and cleared by the user by writing '1' in this bit location when the temperature falls below its corresponding threshold. If the IETD0_0 bit is also asserted, an interrupt is sent to the CPU. If REE_TD[TEMP0_0] is also asserted, a system reset is generated, which clears IETD0_0 and negate the interrupt request. If REE_TD[TEMP0_0] is asserted and a destructive reset is selected (via RES_TD[TEMP0_0] being 0), then a destructive reset is generated. If RES_TD[TEMP0_0] is set, then a functional reset is generated. 0 Currently no occurrence 1 Temperature occurrence detected

50.5.27 Reset Event Enable Register (REE_TD)

This Reset Event Enable Register controls whether the temperature detect signal event causes a reset. If the desired flag bit is enabled, the temperature detect event causes a reset to be generated when the selected temperature passes the trigger event.

If the Temperature Sensor REE and RES bits are set to cause a destructive reset, then during that process the TS REE and RES bits will also get cleared by the destructive reset.

Offset: 0x0304 Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	TEMP1_1	TEMP1_0	0	TEMP0_1	TEMP0_0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1050. Reset Event Enable Register (*REE_TD*)

Table 1016. REE_TD field descriptions

Field	Description
27 TEMP1_1	TEMP1_1 reset enable This bit defines whether a temperature detect assertion generates a system reset for temperature sensor 1. The RES_TD[TEMP1_1] bit determines whether the reset is functional or destructive. 0 Disabled. Temperature Sensor detect does not cause system reset 1 Enabled. Temperature Sensor detect causes system reset
28 TEMP1_0	TEMP1_0 reset enable This bit defines whether a temperature detect assertion generates a system reset for temperature sensor 1. The RES_TD[TEMP1_0] bit determines whether the reset is functional or destructive. 0 Disabled. Temperature Sensor detect does not cause system reset 1 Enabled. Temperature Sensor detect causes system reset
30 TEMP0_1	TEMP0_1 reset enable This bit defines whether a temperature detect assertion generates a system reset for temperature sensor 0. The RES_TD[TEMP0_1] bit determines whether the reset is functional or destructive. 0 Disabled. Temperature Sensor detect does not cause system reset 1 Enabled. Temperature Sensor detect causes system reset
31 TEMP0_0	TEMP0_0 reset enable This bit defines whether a temperature detect assertion generates a system reset for temperature sensor 0. The RES_TD[TEMP0_0] bit determines whether the reset is functional or destructive. 0 Disabled. Temperature Sensor detect does not cause system reset 1 Enabled. Temperature Sensor detect causes system reset

50.5.28 Reset Event Select Register (RES_TD)

This Reset Event Select Register controls whether the temperature sensor detect signal event causes a destructive or functional reset. If the desired flag bit is enabled, the temperature sensor event causes either a destructive or functional reset to be generated when the selected temperature passes the trigger event.

If the Temperature Sensor REE and RES bits are set to cause a destructive reset, then during that process the TS REE and RES bits will also get cleared by the destructive reset.

Offset: 0x0308 Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	TEMP1_1	TEMP1_0	0	TEMP0_1	TEMP0_0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1051. Reset Event Select Register (*RES_TD*)

Table 1017. RES_TD field descriptions

Field	Description
27 TEMP1_1	TEMP1_1 reset event select. This bit defines whether a temperature detect assertion generates a destructive reset or a functional reset for temperature sensor 1. The REE_TD[TEMP1_1] bit determines whether the reset event is enabled. 0 Destructive Reset Generated. Temperature Sensor 1 assertion causes a d structive system reset 1 Functional Reset Generated. Temperature Sensor 1 assertion causes a functional system reset
28 TEMP1_0	TEMP1_0 reset event select. This bit defines whether an temperature detect assertion generates a destructive reset or a functional reset for temperature sensor 1. The REE_TD[TEMP1_0] bit determines whether the reset event is enabled. 0 Destructive Reset Generated. Temperature Sensor 1 assertion causes a destructive system reset 1 Functional Reset Generated. Temperature Sensor 1 assertion causes a functional system reset
30 TEMP0_1	TEMP0_1 reset event select This bit defines whether a temperature detect assertion generates a destructive reset or a functional reset for temperature sensor 0. The REE_TD[TEMP0_1] bit determines whether the reset event is enabled. 0 Destructive Reset Generated. Temperature Sensor 0 assertion causes a destructive system reset 1 Functional Reset Generated. Temperature Sensor 0 assertion causes a functional system reset
31 TEMP0_0	TEMP0_0 reset event select This bit defines whether an temperature detect assertion generates a destructive reset or a functional reset for temperature sensor 0. The REE_TD[TEMP0_0] bit determines whether the reset event is enabled. 0 Destructive Reset Generated. Temperature Sensor 0 assertion causes a destructive system reset 1 Functional Reset Generated. Temperature Sensor 0 assertion causes a functional system reset

50.5.29 Temperature Sensor Configuration register (CTL_TD)

The Temperature Sensor Configuration Registers (CTL_TD) contains two enables that must both be enabled in order for the Temperature Sensor to be operable. It also includes two 4-bit signed trim adjustment register fields that are used to modify both TS trim values (one bus for over temperature and one bus for under temperature).

Offset: 0x030C Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	TS1IE1	TS0IE1	0	0	0	0	0	0	0	0	0	0	0	DOUT1_EN	AOUT1_EN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	TS1IE0	TS0IE0	0(1)	0(1)	0(1)	0(1)	0(1)	0(1)	0(1)	0(1)	0(1)	0(1)	0(1)	DOUT0_EN	AOUT0_EN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

1. These bits should not be written

Figure 1052. Temperature Sensor Configuration register (CTL_TD)

Table 1018. CTL_TD field descriptions

Field	Description
1 TS1IE1	TS1IE1 Temperature Sensor input 1 Interrupt Enable for temperature sensor 1. This bit determines whether an interrupt is seen by the system when the voltage detect event occurs. The MSB in IE_G must be set to access this bit. 0 No interrupt occurs 1 An interrupt occurs
2 TS0IE1	TS0IE1 Temperature Sensor input 0 Interrupt Enable. for temperature sensor 1 This bit determines whether an interrupt is seen by the system when the voltage detect event occurs. The MSB in IE_G must be set to access this bit. 0 No interrupt occurs 1 An interrupt occurs
14 DOUT1_EN	DOUT_EN1. Digital Output Enable for temperature sensor 1. See the Temperature Sensor Chapter for details.
15 AOUT1_EN	AOUT_EN1. Analog Output Enable for temperature sensor 1. See the Temperature Sensor Chapter for details.

Table 1018. CTL_TD field descriptions (continued)

Field	Description
17 TS1IE0	TS1IE0 Temperature Sensor input 1 Interrupt Enable for temperature sensor 0. This bit determines whether an interrupt is seen by the system when the voltage detect event occurs. The MSB in IE_G must be set to access this bit. 0 No interrupt occurs 1 An interrupt occurs
18 TS0IE0	TS0IE0 Temperature Sensor input 0 Interrupt Enable for temperature sensor 0. This bit determines whether an interrupt is seen by the system when the voltage detect event occurs. The MSB in IE_G must be set to access this bit. 0 No interrupt occurs 1 An interrupt occurs
30 DOUT0_EN	DOUT_EN0. Digital Output Enable for temperature sensor 0. See the Temperature Sensor Chapter for details.
31 AOUT0_EN	AOUT0_EN. Analog Output Enable for temperature sensor 0. See the Temperature Sensor Chapter for details.

50.5.30 Temperature Sensor FCCU Event Enable Register (FEE_TD)

This Temperature Sensor FCCU Event Enable Register controls whether the temperature sensor detect signal event causes an event signal to be sent to the FCCU (Fault Control Unit). If the desired flag bit (EPR_TD) is enabled, the temperature sensor detect assertion causes an FCCU event to be generated when the selected temperature exceeds the desired value.

Offset: 0x0318 Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	FEE1_TS1	FEE1_TS0	0	FEE0_TS1	FEE0_TS0
W																
Reset	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1

Figure 1053. Temperature Sensor FCCU Event Enable Register (FEE_TD)

Table 1019. FEE_TD Field description

Field	Description
27 FEE1_TS1	FEE1_TS1 Temp Sensor 1 FCCU event enable. This bit defines whether a temperature detect assertion generates an FCCU event. 0 Disabled. Temp Sensor detect does not cause an FCCU event 1 Enabled. Temp Sensor detect causes an FCCU event
28 FEE1_TS0	FEE1_TS0 Temp Sensor 1 FCCU event enable. This bit defines whether a temperature detect assertion generates an FCCU event. 0 Disabled. Temp Sensor detect does not cause an FCCU event 1 Enabled. Temp Sensor detect causes an FCCU event
30 FEE0_TS1	FEE0_TS1 Temp Sensor 0 FCCU event enable. This bit defines whether a temperature detect assertion generates an FCCU event. 0 Disabled. Temp Sensor detect does not cause an FCCU event 1 Enabled. Temp Sensor detect causes an FCCU event
31 FEE0_TS0	FEE0_TS0 Temp Sensor 0 FCCU event enable. This bit defines whether a temperature detect assertion generates an FCCU event. 0 Disabled. Temp Sensor detect does not cause an FCCU event 1 Enabled. Temp Sensor detect causes an FCCU event

50.5.31 IPS Regulator Monitor Register (IREGU_MON)

This register indicates all important parameters of dreg controller.

Offset: 0x0368 Access: User read-only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	QVF_PULSE	VALID	0	DELTA		0	PCURR_COMP					NCURR_COMP				
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	QVF_PEAK								ACTIVE_SOURCES							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1054. Digital Regulator Monitor Register (IREGU_MON)

Table 1020. IREGU_MON field descriptions

Field	Description
0 QVF_PULSE	Error flag indicator. 0 No error 1 Error
1 VALID	Indicates valid thermometric code. 0 thermometric codes are not valid 1 hermometric codes are valid

Table 1020. IREGU_MON field descriptions (continued)

Field	Description
3:4 DELTA	Indicates value in 2'complement.+1 represents increase in current sources and -1 decrease in current sources.
6:10 PCURR_COMP	Represents preg current computational.
11:15 NCURR_COMP	Represents nreg current computational.
16:23 QVF_PEAK	Indicates consecutive number of invalid thermometric codes.
24:31 ACTIVE_SOURCES	Indicates number of digital ballast currently active.

50.5.32 ESR0 ConFiGuration Register (ESR0_CFG)

The ESR0 ConFiGuration Register (ESR0_CFG) is used to map device reset status on an external Pad. The register reset is ipg_hard_async_reset_por_b.

Offset: 0x03A8 Access: User read only

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	ESR0_CFG
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1055. ESR0 ConFiGuration Register (ESR0_CFG)

Table 1021. ESR0_CFG field descriptions

Field	Description
31 ESR0_CFG	The ESR0_CFG register is a read only register by IPS bus at 0x03A8 address, and read-write register through DCF bus at 1F8hex address. 0 means the device reset status is not available on an external pad 1 means the device reset status is available on an external pad

50.5.33 User BIST Flags_IO10 Register (BIST_FLAGS_IO10)

This configuration register contains a monitor of the flags IO10 phase1 and the flags phase2 coming from the User BIST sub-block.

Offset: 0x03CC Access: User read-only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	F_P2_IO10													
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	F_P1_IO10													
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1056. User BIST Flags_IO10 Register (BIST_FLAGS_IO10)

Table 1022. BIST_FLAGS_IO10 field description

Field	Description
2:15 F_P2_IO10	The 14 bits of field F_P2_IO10 flags are the phase two flags of the User BIST. They are read only bits.
18:31 F_P1_IO10	The 14 bits of field F_P1_IO10 flags are the phase one flags of the User BIST. They are read only bits.

50.5.34 User BIST Flags Register (BIST_FLAGS)

This configuration register contains a monitor of the flags phase1 and the flags phase2 coming from the User BIST sub-block.

Offset: 0x03D0 Access: User read-only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	F_P2_AD1		FLAGS_P2													
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	F_P1_AD1		FLAGS_P1													
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1057. User BIST Flags Register (BIST_FLAGS)

Table 1023. BIST_FLAGS field description

Field	Description
0:1 F_P2_AD1	The 2 bits of field F_P2_AD1 are:bist_fsm_flags_p2[14], bist_fsm_flags_p2[10] of User BIST. They are read only bits.
2:15 FLAGS_P2	The 14 bits of field FLAGS_P2 are the phase two flags of the User BIST. They are read only bits.

Table 1023. BIST_FLAGS field description (continued)

Field	Description
16:17 F_P1_AD1	The 2 bits of field F_P1_AD1 are:bist_fsm_flags_p1[14], bist_fsm_flags_p1[10] of User BIST. They are read only bits.
18:31 FLAGS_P1	The 14 bits of field FLAGS_P1 are the phase one flags of the User BIST. They are read only bits.

Note: See BIST flag list in Table 148: Monitors to check of Chapter 10: Power management.

50.5.35 User BIST Control Register (BIST_CTRL)

The User BIST Control register (BIST_CTRL) contains both control bits and status bits to manage the User BIST.

Offset: 0x03D4 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	IRQST	0	0	0	IRQEN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	NCFST				NCFEN	0	STATUS			0	0	0	START
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1058. User BIST Control Register (BIST_CTRL)

Table 1024. BIST_CTRL field descriptions

Field	Description
11 IRQST	IRQST is the InteRrupt reQuest pending STatus bit to generate a User BIST Interrupt. It is a R/W0 register. That means Set by HW and Cleared by SW. The destructive ipg_hard_async_reset_dest_b resets this bit. 0 No interrupt occurs 1 An interrupt occurs
15 IRQEN	IRQEN is the InteRrupt reQuest ENable bit to permit a User BIST Interrupt. It is a R/W register. The destructive ipg_hard_async_reset_dest_b resets this bit. 0 No interrupt occurs 1 An interrupt occurs
19 NCFST	NCFST is the Not Critical Fault STatus bit for User BIST. It is set by HW when NCFEN is set and User BIST fault occurs and Cleared by SW by writing '1' to a FCCU Register bit. The destructive ipg_hard_async_reset_dest_b resets this bit. 0 No User BIST NCF occured 1 A User BIST NCF occured

Table 1024. BIST_CTRL field descriptions (continued)

Field	Description
23 NCFEN	NCFEN is the Not Critical Fault ENable bit to permit a User BIST Not Critical Fault. It is a R/W register. The destructive ipg_hard_async_reset_dest_b resets these bits. 0 No User BIST NCF enabled 1 A User BIST NCF enabled
25:27 STATUS	STATUS Register are three read-only bits, that according to the value has a own meaning. 000 BIST_IDLE 001 BIST_RUN 010 BIST_PASSED 011 BIST_FAILED 100 BIST_ABORT 101 Reserved 110 Reserved 111 Reserved
31 START	START User BIST bit. If set by SW a 1 is present in the START bit for a ipg_clk clock cycle only, then the START bit register is clear by HW. The destructive ipg_hard_async_reset_dest_b resets this bit. 0 No start pulse 1 A start pulse occurs of a 1 ipg_clk clock cycle duration

50.5.36 User BIST Time1 and Time0 Register (BIST_TIME10)

The User BIST Time1 and Time0 Register (BIST_TIME10) contains two time numbers Time1 and the Time0 giving the ipg_clk cycles number required by the User BIST timer1 and timer0. The reset value is calculated based on 70MHz. This register cannot be written bytewise. See *Table 147: Calculating the BIST_TIME values* of *Chapter 10: Power management*.

Offset: 0x03D8 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	TIME_1												
W																
Reset	0	0	0	0	0	1	0	1	0	1	1	1	0	1	1	1

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	TIME_0												
W																
Reset	0	0	0	0	0	0	1	0	1	0	1	1	1	0	1	1

Figure 1059. User BIST Time1 and Time0 Register (BIST_TIME10)

Table 1025. BIST_TIME10 field descriptions

Field	Description
3:15 TIME_1	TIME_1 is the the ipg_clk cycles number required by the TIMER 1 in the User BIST management. The destructive ipg_hard_async_reset_dest_b resets these bits. They are both read/write register bits. The reset value corresponds to 20 µs.
19:31 TIME_0	TIME_0 is the the ipg_clk cycles number required by the TIMER 0 in the User BIST management. The destructive ipg_hard_async_reset_dest_b resets these bits. They are both read/write register bits. The reset value corresponds to 10 µs.

50.5.37 User BIST Time3 and Time2 Register (BIST_TIME32)

The User BIST Time3 and Time2 Register (BIST_TIME32) contains two time numbers Time3 and the Time2 giving the ipg_clk cycles number required by the User BIST timer3 and timer2. The reset value is calculated based on 70MHz. This register cannot be written bytewise. See *Table 147: Calculating the BIST_TIME values* of *Chapter 10: Power management*.

Offset: 0x03DC Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	TIME_3												
W																
Reset	0	0	0	0	1	0	0	0	0	0	1	1	0	0	1	1

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	TIME_2												
W																
Reset	0	0	0	0	0	0	0	1	0	1	0	1	1	1	0	1

Figure 1060. User BIST Time3 and Time2 Register (BIST_TIME32)

Table 1026. BIST_TIME32 field descriptions

Field	Description
3:15 TIME_3	TIME_3 is the the ipg_clk cycles number required by the TIMER 3 in the User BIST management. The destructive ipg_hard_async_reset_dest_b resets these bits. They are both read/write register bits. The reset value corresponds to 30 µs.
19:31 TIME_2	TIME_2 is the the ipg_clk cycles number required by the TIMER 2 in the User BIST management. The destructive ipg_hard_async_reset_dest_b resets these bits. They are both read/write register bits. The reset value corresponds to 5 µs.

50.5.38 User BIST Time6 and Time5 Register (BIST_TIME65)

The User BIST Time6 and Time5 Register (BIST_TIME65) contains two time numbers Time6 and the Time5 giving the ipg_clk cycles number required by the User BIST timer6

and timer5. The reset value is calculated based on 70MHz. This register cannot be written bytewise. See *Table 147: Calculating the BIST_TIME values* of *Chapter 10: Power management*.

Offset: 0x03E0 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	TIME_6												
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	TIME_5												
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0

Figure 1061. User BIST Time6 and Time5 Register (BIST_TIME65)

Table 1027. BIST_TIME65 field descriptions

Field	Description
3:15 TIME_6	TIME_6 is the the ipg_clk cycles number required by the TIMER 6 in the User BIST management. The destructive ipg_hard_async_reset_dest_b resets these bits. They are both read/write register bits. The reset value corresponds to 200 ns.
19:31 TIME_5	TIME_5 is the the ipg_clk cycles number required by the TIMER 5 in the User BIST management. The destructive ipg_hard_async_reset_dest_b resets these bits. They are both read/write register bits. The reset value corresponds to 200 ns.

50.5.39 User BIST VD Under Test Monitor Register (BIST_DEBUG)

This VD Under Test Monitor register contains a monitor of the Voltage Detector Under Test of the USER BIST sub-block.

Offset: 0x03E4 Access: User read-only

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	VD_MON					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1062. User BIST VD Under Test Monitor Register (BIST_DEBUG)

Table 1028. BIST_DEBUG field description

Field	Description
26:31 VD_MON	The six read only bits of field VD_MON give the current VD Under Test of the User BIST unit, among 14 VDs available, according to the following values: 001011 MVD098_C 001100 MVD098_FL 000100 MVD270_C 000101 MVD270_FL 001010 LVD108_C 000001 LVD290_C 001111 LVD290_XOSC 001110 LVD290_IOL1 000011 LVD290_FL 000010 LVD290_ADL0 010101 LVD290_ADL1 010001 LVD400_IOL0 010000 LVD400_IOL1 010011 LVD400_ADL0 010010 LVD400_ADL1 001001 HVD140_C 011101 HVD400_C 000111 UVD145_FL 001000 UVD145_C 011100 UVD600_C 000000 No VD under Test

50.6 Temperature Sensor Interface Logic

The Temperature Sensor interface consists of two enables: an analog and digital enable, a set of mode registers, and an overtemp threshold select control.

For more information regarding the Temperature Sensor function, please refer to the *Chapter 23: Temperature sensor (TSENS)*.

50.7 FCCU Interface

The PMU digital generates five different failure events to the FCCU.

- **HVD_ERROR** : This failure event is generated to the FCCU when any of the HVD trips and FCCU event is enabled in FEE_HVx registers.
- **LVD_ERROR** : This failure event is generated to the FCCU when any of the LVD trips and FCCU event is enabled in FEE_LVx registers.
- **TEMP_ERROR** : This failure event is generated to the FCCU when undertemp or overtemp event is occured and this event is enabled in FEE_TD register.
- **PMC VD BIST**: This failure event is generated to the FCCU when PMC BIST failure happens.
- **DPMC_DCF_SAFETY_ERR** : This failure event is generated to the FCCU if any of the safety features of the DCF client occurs, that is parity error, second write mismatch or triple voted register flip. This event can be a pulse if the failure is transient or toggle if the failure is permanent such as parity error.

50.8 Power On Reset (MC_RGM Phase Gates)

The PMU digital logic interfaces to the Reset Generation Module in order to indicate the type and conditions of the resets from the PMU and Temperature Sensor Analog blocks.

There are two signals for each the PMU and Temp Sensor blocks; one is used to indicate a destructive reset, and one is used to indicate a functional reset.

The PMU digital logic also indicates when is ok to for the RGM to allow transition to the next POR phase.

The digital PMU also uses signals from the RGM to indicate when some of the reset signals should be masked during the POR sequence.

During the POR phase 3 exit, the PMU digital logic uses all the enabled Voltage Detects to determine if it is correct to exit from Phase 3. Once all the enabled LVDs and HVDs have become deasserted and the time from the LVD circuits to adjust to new trim values has elapsed, then the signal is sent to the RGM that the correct voltage levels have been achieved to exit from Phase 3.

50.9 Flash interface (DCF client usage)

The PMU digital is updated during POR from the Flash block via a DCF client. This load is used for several pieces of data, but most of the data is trim values being used for analog modules (PMU and Temperature Sensor).

These trim bits that are loaded from the flash are not readable via the IP bus. These trim values loaded via the DCF client are not compacted (they are written individually, and not several at a time).

A DCF client is used to interface to the flash for these early data accesses. This client also performs triple voting to keep the data safe from stray bit flips.

Another set of data that is loaded during the DCF single read process are updated values for the REE and RES bits. Please see *Section 50.5.27* for more information regarding the REE/RES bits load from flash.

If the REE bit is loaded via the DCF as set (enabled), then it is not possible to clear this REE bit. The reset value of the REE DCF bits are all low (thus to allow the bits to be rewriteable), but the reset value of the REE register bits are all high (to enable the resets normally out of POR).

The REE DCF bits loaded are compacted into a single DCF write (they do not match the IPS Bus register bit locations). The REE DCF loaded bits do not have the same limitation that they can only be written once and any further writes must be to the same value. The REE DCF bits can be rewritten to a new value during a new reset event.

When the REE DCF bits are updated, the REE and RES register bits are updated at the same time to the new value.

All of the DCF clients are reset via the destructive reset signal.

For more information regarding the DCF records, please refer to the *Chapter 9: Device Configuration Format (DCF) Records*.

The trim data is read out of the flash's differential read space (slower access, but less issues with voltage variances). Once the differential reads are completed, the flash indicates this to the PMU digital via the SSCM module.

Once the PMU trim values have been read, the analog circuit has been given time to update to the new trim values, and the LVDs have all deasserted, a signal is asserted to the flash to indicate that the voltage is high enough for faster single access reads.

All of the LVD signals must indicate that a valid voltage has been reached before the voltage indication will be sent to the flash.

For more information regarding the Flash differential reads and voltage levels required, please refer to the Flash chapter.

51 Power Control Unit (MC_PCU)

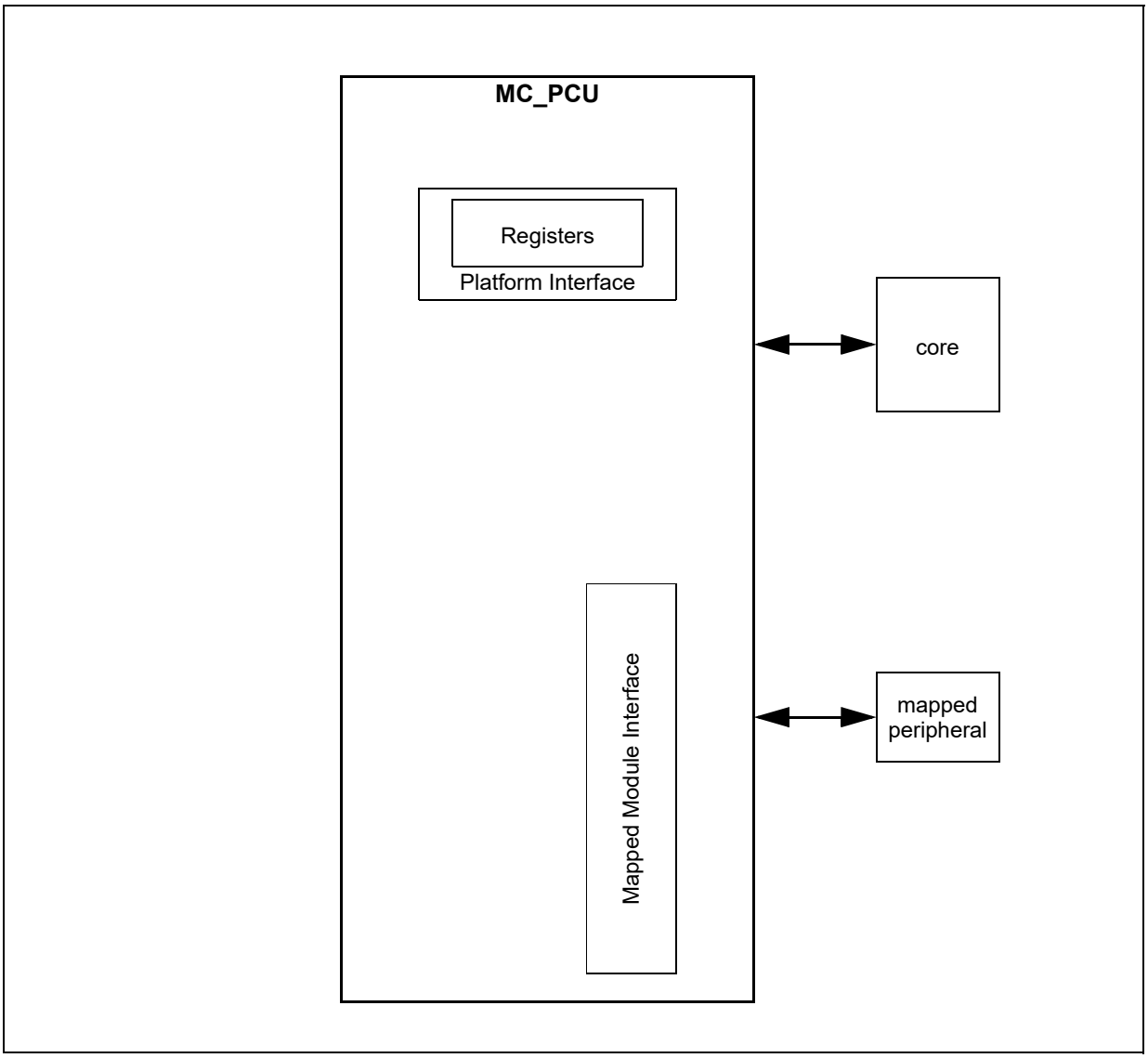
51.1 Introduction

51.1.1 Overview

The power control unit (MC_PCU) acts as a bridge for mapping the PMC peripheral to the MC_PCU address space.

Figure 1063 shows the MC_PCU block diagram.

Figure 1063. MC_PCU block diagram



51.1.2 Features

The MC_PCU includes the following feature:

- Maps the PMC registers to the MC_PCU address space

51.2 External signal description

The MC_PCU has no connections to any external pins.

51.3 Memory map and register definition

51.3.1 Memory map

Table 1029. MC_PCU register description

Offset Address	Name	Description	Size	Access			Location
				User	Supervisor	Test	
0x0040	PCU_PSTAT	Power Domain Status Register	word	read	read	read	*51.3.2.1 on page 1777*

Note: Any access to unused registers as well as write accesses to read-only registers:
– does not change register content
– causes a transfer error

Table 1030. MC_PCU memory map

Address	Name
0x0004 – 0x003C	Reserved
0x0040	PCU_PSTAT
0x0044 – 0x03FC	Reserved
0x0400 – 0x07FC	PMC registers
0x0800 – 0x3FFC	Reserved

51.3.2 Register descriptions

All registers may be accessed as 32-bit words, 16-bit half-words, or 8-bit bytes. The bytes are ordered according to big endian. For example, the PD0 field of the PCU_PSTAT register may be accessed as a word at address 0x0040, as a half-word at address 0x0042, or as a byte at address 0x0043.

51.3.2.1 Power Domain Status Register (PCU_PSTAT)

Address: 0x0040 Access: User read, Supervisor read, Test read

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	PD0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Figure 1064. Power Domain Status Register (PCU_PSTAT)

This register reflects the power status of all available power domains.

Table 1031. Power Domain Status Register (PCU_PSTAT) field descriptions

Field	Description
PD0	Power status for power domain #0 0 Power domain is inoperable 1 Power domain is operable

52 Mode Entry Module (MC_ME)

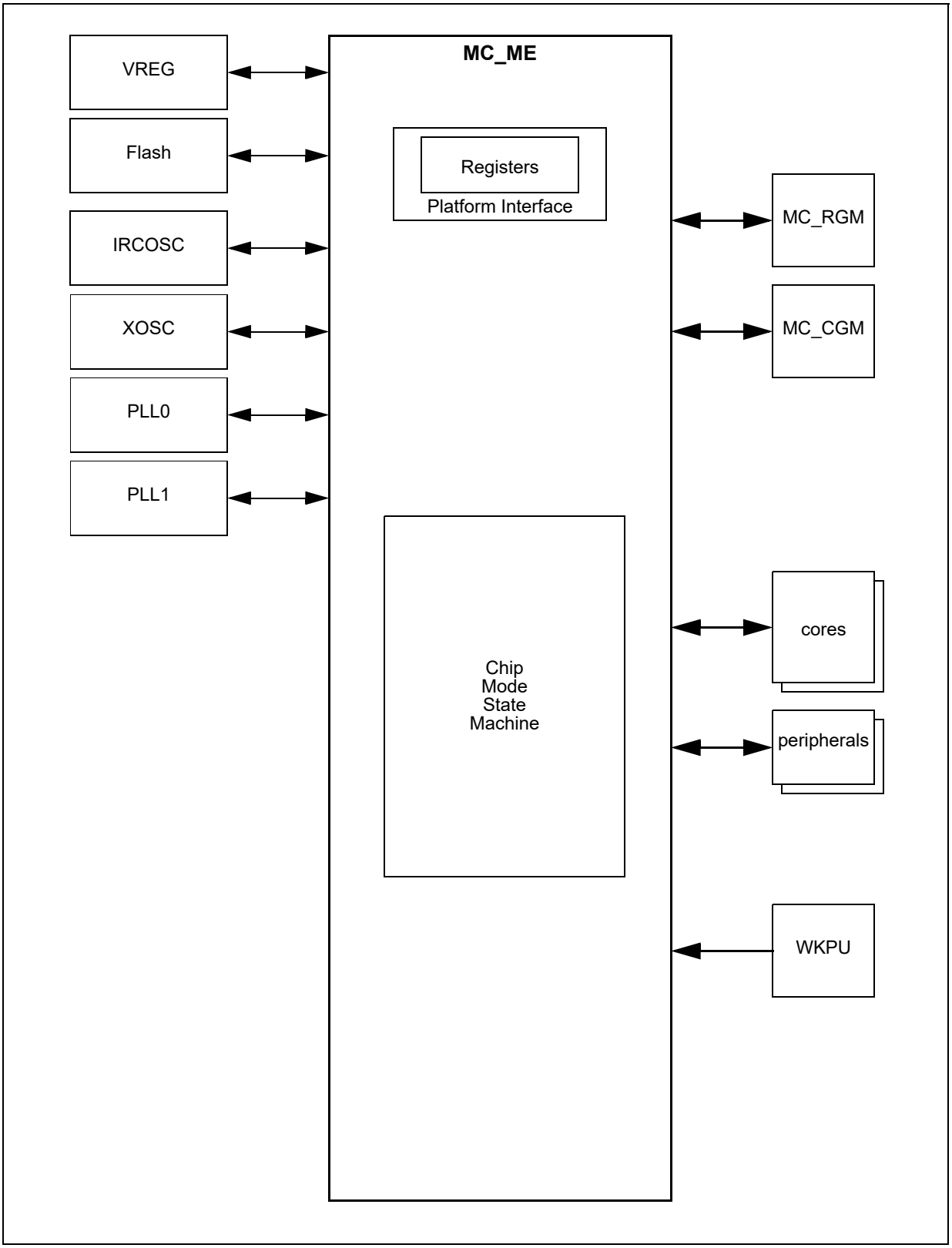
52.1 Introduction

52.1.1 Overview

The MC_ME controls the chip mode and mode transition sequences in all functional states. It also contains configuration, control and status registers accessible for the application.

Figure 1065 shows the MC_ME block diagram.

Figure 1065. MC_ME block diagram



52.1.2 Features

The MC_ME includes the following features:

- Control of the available modes by the ME_ME register
- Definition of various chip mode configurations by the ME_<*mode*>_MC registers
- Control of the actual chip mode by the ME_MCTL register
- Capture of the current mode and various resource status within the contents of the ME_GS register
- Optional generation of various mode transition interrupts
- Status bits for each cause of invalid mode transitions
- Peripheral clock gating control based on the ME_RUN_PC0–7, ME_LP_PC0–7, and ME_PCTL*n* registers
- Capture of current peripheral and additional core clock gated/enabled status
- Progressive system clock switching when transitioning from a lower power consumption mode to a higher power consumption mode, and vice versa

52.1.3 Modes of operation

The MC_ME is based on several chip modes corresponding to different usage models of the chip. Each mode is configurable and can define a policy for energy and processing power management to fit particular system requirements. An application can easily switch from one mode to another depending on the current needs of the system. The operating modes controlled by the MC_ME are divided into system and user modes. The system modes are modes such as RESET, DRUN, SAFE, and TEST. These modes aim to ease the configuration and monitoring of the system. The user modes are modes such as RUN0–3, HALT0, and STOP0 which can be configured to meet the application requirements in terms of energy management and available processing power. The modes DRUN, SAFE, TEST, and RUN0–3 are the chip software running modes.

Table 1032 describes the MC_ME modes.

Table 1032. MC_ME mode descriptions

Name	Description	Entry	Exit
RESET	This is a chip-wide virtual mode during which the application is not active. The system remains in this mode until all resources are available for the embedded software to take control of the chip. It manages hardware initialization of chip configuration, voltage regulators, clock sources, and flash modules.	system reset assertion from MC_RGM	system reset deassertion from MC_RGM
DRUN	This is the entry mode for the embedded software. It provides full accessibility to the system and enables the configuration of the system at startup. It provides the unique gate to enter user modes.	system reset deassertion from MC_RGM, software request from SAFE, TEST and RUN0–3	system reset assertion, RUN0–3, TEST via software, SAFE via software or hardware failure.

Table 1032. MC_ME mode descriptions (continued)

Name	Description	Entry	Exit
SAFE	This is a chip-wide service mode which may be entered on the detection of a recoverable error. It forces the system into a predefined safe configuration from which the system may try to recover.	hardware failure, software request from DRUN, TEST, and RUN0–3	system reset assertion, DRUN via software
TEST	This is a chip-wide service mode which is intended to provide a control environment for chip software testing.	software request from DRUN	system reset assertion, DRUN via software
RUN0–3	These are software running modes where most processing activity is done. These various run modes allow to enable different clock and power configurations of the system with respect to each other.	software request from DRUN or other RUN0–3, interrupt event from HALT0, interrupt or wakeup event from STOP0	system reset assertion, SAFE via software or hardware failure, other RUN0–3 modes, HALT0, STOP0 via software
HALT0	This is a reduced-activity low-power mode during which the clock to the core is disabled. It can be configured to switch off analog peripherals such as: clock sources, flash, main regulator, for efficient power management at the cost of higher wakeup latency.	software request from RUN0–3	system reset assertion, SAFE on hardware failure, RUN0–3 on interrupt event
STOP0	This is an advanced low-power mode during which the clock to the core is disabled. It may be configured to switch off most of the peripherals including clock sources for efficient power management at the cost of higher wakeup latency.	software request from RUN0–3	system reset assertion, SAFE on hardware failure, RUN0–3 on interrupt event or wakeup event

52.2 External signal description

The MC_ME has no connections to any external pins.

52.3 Memory map and register definition

The MC_ME contains registers for:

- Mode selection and status reporting
- Mode configuration
- Mode transition interrupts status and mask control
- Scalable number of peripheral sub-mode selection and status reporting
- Scalable number of additional core enabling and disabling control and status reporting

52.3.1 Memory map

Table 1033. MC_ME register description

Address offset	Name	Description	Location
0x000	ME_GS	Global Status	*Section 52.3.2.1*
0x004	ME_MCTL	Mode Control	*Section 52.3.2.2*
0x008	ME_ME	Mode Enable	*Section 52.3.2.3*
0x00C	ME_IS	Interrupt Status	*Section 52.3.2.4*
0x010	ME_IM	Interrupt Mask	*Section 52.3.2.5*
0x014	ME_IMTS	Invalid Mode Transition Status	*Section 52.3.2.6*
0x018	ME_DMTS	Debug Mode Transition Status	*Section 52.3.2.7*
0x020	ME_RESET_MC	RESET Mode Configuration	*Section 52.3.2.8*
0x024	ME_TEST_MC	TEST Mode Configuration	*Section 52.3.2.9*
0x028	ME_SAFE_MC	SAFE Mode Configuration	*Section 52.3.2.10*
0x02C	ME_DRUN_MC	DRUN Mode Configuration	*Section 52.3.2.11*
0x030	ME_RUN0_MC	RUN0 Mode Configuration	*Section 52.3.2.12*
0x034	ME_RUN1_MC	RUN1 Mode Configuration	*Section 52.3.2.12*
0x038	ME_RUN2_MC	RUN2 Mode Configuration	*Section 52.3.2.12*
0x03C	ME_RUN3_MC	RUN3 Mode Configuration	*Section 52.3.2.12*
0x040	ME_HALT0_MC	HALT0 Mode Configuration	*Section 52.3.2.13*
0x048	ME_STOP0_MC	STOP0 Mode Configuration	*Section 52.3.2.14*
0x060	ME_PS0	Peripheral Status 0	*Section 52.3.2.15*
0x064	ME_PS1	Peripheral Status 1	*Section 52.3.2.16*
0x068	ME_PS2	Peripheral Status 2	*Section 52.3.2.17*
0x06C	ME_PS3	Peripheral Status 3	*Section 52.3.2.18*
0x070	ME_PS4	Peripheral Status 4	*Section 52.3.2.19*
0x074	ME_PS5	Peripheral Status 5	*Section 52.3.2.20*
0x078	ME_PS6	Peripheral Status 6	*Section 52.3.2.21*
0x07C	ME_PS7	Peripheral Status 7	*Section 52.3.2.22*
0x080	ME_RUN_PC0	Run Peripheral Configuration 0	*Section 52.3.2.23*
0x084	ME_RUN_PC1	Run Peripheral Configuration 1	*Section 52.3.2.23*
0x088	ME_RUN_PC2	Run Peripheral Configuration 2	*Section 52.3.2.23*
0x08C	ME_RUN_PC3	Run Peripheral Configuration 3	*Section 52.3.2.23*
0x090	ME_RUN_PC4	Run Peripheral Configuration 4	*Section 52.3.2.23*
0x094	ME_RUN_PC5	Run Peripheral Configuration 5	*Section 52.3.2.23*
0x098	ME_RUN_PC6	Run Peripheral Configuration 6	*Section 52.3.2.23*
0x09C	ME_RUN_PC7	Run Peripheral Configuration 7	*Section 52.3.2.23*

Table 1033. MC_ME register description (continued)

Address offset	Name	Description	Location
0x0A0	ME_LP_PC0	Low-Power Peripheral Configuration 0	Section 52.3.2.24
0x0A4	ME_LP_PC1	Low-Power Peripheral Configuration 1	Section 52.3.2.24
0x0A8	ME_LP_PC2	Low-Power Peripheral Configuration 2	Section 52.3.2.24
0x0AC	ME_LP_PC3	Low-Power Peripheral Configuration 3	Section 52.3.2.24
0x0B0	ME_LP_PC4	Low-Power Peripheral Configuration 4	Section 52.3.2.24
0x0B4	ME_LP_PC5	Low-Power Peripheral Configuration 5	Section 52.3.2.24
0x0B8	ME_LP_PC6	Low-Power Peripheral Configuration 6	Section 52.3.2.24
0x0BC	ME_LP_PC7	Low-Power Peripheral Configuration 7	Section 52.3.2.24
0x0C0–0x0CE	Reserved		
0x0CF	ME_PCTL15	SIUL Control	Section 52.3.2.25
0x0D0–0x0DD	Reserved		
0x0DE	ME_PCTL30	PIT_0 Control	Section 52.3.2.25
0x0DF–0x0E3	Reserved		
0x0E4	ME_PCTL36	DMAMUX_0 Control	Section 52.3.2.25
0x0E5	Reserved		
0x0E6	ME_PCTL38	CRC_0 Control	Section 52.3.2.25
0x0E7–0x0FD	Reserved		
0x0FE	ME_PCTL62	CCCU_0 Control	Section 52.3.2.25
0x0FF–0x104	Reserved		
0x105	ME_PCTL69	MCAN_2 Control	Section 52.3.2.25
0x106	ME_PCTL70	MCAN_0 Control	Section 52.3.2.25
0x107–0x109	Reserved		
0x10A	ME_PCTL74	MCAN_RAM_0 Control	Section 52.3.2.25
0x10B–0x11A	Reserved		
0x11B	ME_PCTL91	LINFlexD_2 Control	Section 52.3.2.25
0x11C	ME_PCTL92	LINFlexD_0 Control	Section 52.3.2.25
0x11D–0x121	Reserved		
0x122	ME_PCTL98	DSPI_1 Control	Section 52.3.2.25
0x123	ME_PCTL99	DSPI_0 Control	Section 52.3.2.25
0x124–0x127	Reserved		
0x128	ME_PCTL104	SENT_0 Control	Section 52.3.2.25
0x129–0x12A	Reserved		
0x12B	ME_PCTL107	FlexRay Control	Section 52.3.2.25
0x12C–0x12E	Reserved		
0x12F	ME_PCTL111	ADCS_SEQ_AIPS_0 Control	Section 52.3.2.25

Table 1033. MC_ME register description (continued)

Address offset	Name	Description	Location
0x130	ME_PCTL112	ADCS_AIPS_0 Control	Section 52.3.2.25
0x131–0x138	Reserved		
0x139	ME_PCTL121	ADC_SEQ_2 Control	Section 52.3.2.25
0x13A	ME_PCTL122	ADC_SEQ_1 Control	Section 52.3.2.25
0x13B	ME_PCTL123	ADC_SEQ_0 Control	Section 52.3.2.25
0x13C	Reserved		
0x13D	ME_PCTL125	ADCSAR_2 Control	Section 52.3.2.25
0x13E	ME_PCTL126	ADCSAR_1 Control	Section 52.3.2.25
0x13F	ME_PCTL127	ADCSAR_0 Control	Section 52.3.2.25
0x140	ME_PCTL128	eTimer_1 Control	Section 52.3.2.25
0x141	ME_PCTL129	eTimer_0 Control	Section 52.3.2.25
0x142	ME_PCTL130	eTimer_3 Control	Section 52.3.2.25
0x143	ME_PCTL131	eTimer_2 Control	Section 52.3.2.25
0x144–0x147	Reserved		
0x148	ME_PCTL136	CTU_1 Control	Section 52.3.2.25
0x149	Reserved		
0x14A	ME_PCTL138	FlexPWM_1 Control	Section 52.3.2.25
0x14B	ME_PCTL139	FlexPWM_0 Control	Section 52.3.2.25
0x14C	ME_PCTL140	FlexPWM_3 Control	Section 52.3.2.25
0x14D	ME_PCTL141	FlexPWM_2 Control	Section 52.3.2.25
0x14E	Reserved		
0x14F	ME_PCTL143	PWM_SYNCHR Control	Section 52.3.2.25
0x150–0x17D	Reserved		
0x17E	ME_PCTL190	CCCU_1 Control	Section 52.3.2.25
0x17F–0x185	Reserved		
0x186	ME_PCTL198	M_CAN_1 Control	Section 52.3.2.25
0x187–0x189	Reserved		
0x18A	ME_PCTL202	MCAN_RAM_1 Control	Section 52.3.2.25
0x18B–0x19A	Reserved		
0x19B	ME_PCTL219	LINFlexD_3 Control	Section 52.3.2.25
0x19C	ME_PCTL220	LINFlexD_1 Control	Section 52.3.2.25
0x19D–0x1A1	Reserved		
0x1A2	ME_PCTL226	DSPI_3 Control	Section 52.3.2.25
0x1A3	ME_PCTL227	DSPI_2 Control	Section 52.3.2.25
0x1A4–0x1A7	Reserved		

Table 1033. MC_ME register description (continued)

Address offset	Name	Description	Location
0x1A8	ME_PCTL232	SENT_1 Control	Section 52.3.2.25
0x1A9–0x1AE	Reserved		
0x1AF	ME_PCTL239	ADCS_SEQ_AIPS_1 Control	Section 52.3.2.25
0x1B0	ME_PCTL240	ADCS_AIPS_1 Control	Section 52.3.2.25
0x1B1–0x1B3	Reserved		
0x1B4	ME_PCTL244	CTU_0 Control	Section 52.3.2.25
0x1B5–0x1B8	Reserved		
0x1B9	ME_PCTL249	ADC_SEQ_5 Control	Section 52.3.2.25
0x1BA	ME_PCTL250	ADC_SEQ_4 Control	Section 52.3.2.25
0x1BB	ME_PCTL251	ADC_SEQ_3 Control	Section 52.3.2.25
0x1BC	Reserved		
0x1BD	ME_PCTL253	ADC_5 Control	Section 52.3.2.25
0x1BE	ME_PCTL254	ADC_4 Control	Section 52.3.2.25
0x1BF	ME_PCTL255	ADC_3 Control	Section 52.3.2.25
0x1C0	ME_CS	Core Status	Section 52.3.2.26
0x1C4	ME_CCTL0	Core Control	Section 52.3.2.27
0x1C6–0x1DF	Reserved		
0x1E0	ME_CADDR0	Core Address	Section 52.3.2.28
0x1E4–0x1EB	Reserved		

Note: Any access to unused registers as well as write accesses to read-only registers will:

- *Not change register content*
- *Cause a transfer error*

52.3.2 Register description

Unless otherwise noted, all registers may be accessed as 32-bit words, 16-bit half-words, or 8-bit bytes. The bytes are ordered according to big endian. For example, the ME_RUN_PC0 register may be accessed as a word at address 0x080, as a half-word at address 0x082, or as a byte at address 0x083.

Some fields may be read-only, and their reset value of ‘1’ or ‘0’ and the corresponding behavior cannot be changed.

52.3.2.1 Global Status Register (ME_GS)

Address: 0x000 Access: User read, Supervisor read, Test read

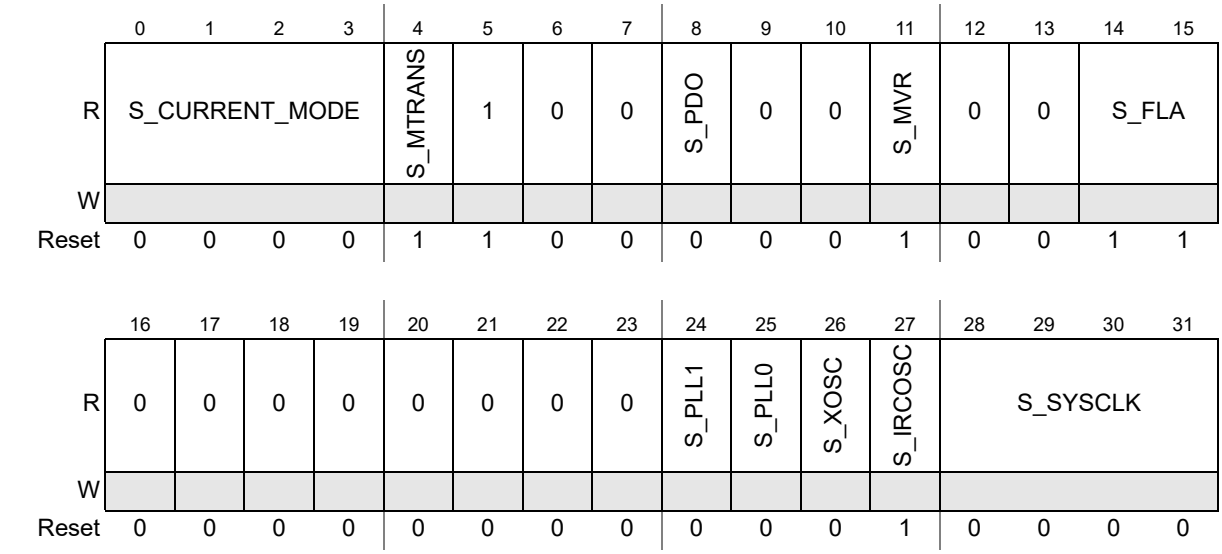


Figure 1066. Global Status Register (ME_GS)

This register contains global mode status.

Table 1034. ME_GS field descriptions

Field	Description
0:3 S_CURRENT_MODE	Current chip mode status 0000 RESET 0001 TEST 0010 SAFE 0011 DRUN 0100 RUN0 0101 RUN1 0110 RUN2 0111 RUN3 1000 HALT0 1001 Reserved 1010 STOP0 1011 Reserved 1100 Reserved 1101 Reserved 1110 Reserved 1111 Reserved
4 S_MTRANS	Mode transition status 0 Mode transition process is not active 1 Mode transition is ongoing

Table 1034. ME_GS field descriptions (continued)

Field	Description
8 S_PDO	Output power-down status This bit specifies output power-down status of I/Os. This bit is asserted whenever outputs of pads are forced to high impedance state or the pads power sequence driver is switched off. 0 No automatic safe gating of I/Os used and pads power sequence driver is enabled 1 In STOP0 mode, the power sequence driver of the output pads is disable to reduce power consumption, but the state of the output remains functional.
11 S_MVR	Main voltage regulator status 0 Main voltage regulator is not ready 1 Main voltage regulator is ready for use
14:15 S_FLA	Flash availability status 00 Flash is not available 01 Flash is in power-down mode 10 Flash is in low-power mode 11 Flash is in normal mode and available for use
24 S_PLL1	Secondary PLL status 0 Secondary PLL is not stable 1 Secondary PLL is providing a stable clock
25 S_PLL0	Primary PLL status 0 Primary PLL is not stable 1 Primary PLL is providing a stable clock
26 S_XOSC	External crystal oscillator status 0 External crystal oscillator is not stable 1 External crystal oscillator is providing a stable clock
27 S_IRCOSC	16 MHz Internal RC oscillator status 0 16 MHz Internal RC oscillator is not stable 1 16 MHz Internal RC oscillator is providing a stable clock
28:31 S_SYSCLK	System clock switch status These bits specify the system clock currently used by the system. 0000 16 MHz Internal RC oscillator 0001 external crystal oscillator 0010 primary PLL (PHI) 0011 Reserved 0100 Secondary PLL 0101 Reserved 0110 Reserved 0111 Reserved 1000 Reserved 1001 Reserved 1010 Reserved 1011 Reserved 1100 Reserved 1101 Reserved 1110 Reserved 1111 System clock is disabled

52.3.2.2 Mode Control Register (ME_MCTL)

Address: 0x004 Access: User read/write, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	TARGET_MODE				0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	1	0	1	0	0	1	0	1	0	0	0	0	1	1	1	1
W	KEY															
Reset	1	0	1	0	0	1	0	1	0	0	0	0	1	1	1	1

Figure 1067. Mode Control Register (ME_MCTL)

This register is used to trigger software-controlled mode changes. Depending on the modes as enabled by ME_ME register bits, configurations corresponding to unavailable modes are reserved and access to ME_<*mode*>_MC registers must respect this for successful mode requests.

Note: Byte and half-word write accesses are not allowed for this register as a predefined key is required to change its value.

Table 1035. ME_MCTL field descriptions

Field	Description
0:3 TARGET_MODE	Target chip mode These bits provide the target chip mode to be entered by software programming. The mechanism to enter into any mode by software requires the write operation twice: first time with key, and second time with inverted key. These bits are automatically updated by hardware while entering SAFE on hardware request. Also, while exiting from the HALT0 and STOP0 modes on hardware exit events, these are updated with the appropriate RUN0–3 mode value. 0000 RESET (triggers a 'functional' reset event) 0001 TEST 0010 SAFE 0011 DRUN 0100 RUN0 0101 RUN1 0110 RUN2 0111 RUN3 1000 HALT0 1001 reserved 1010 STOP0 1011 Reserved 1100 Reserved 1101 Reserved 1110 Reserved 1111 RESET (triggers a 'destructive' reset event)
16:31 KEY	Control key These bits enable write access to this register. Any write access to the register with a value different from the keys is ignored. Read access will always return inverted key. 0101 1010 1111 0000 (0x5AF0): KEY 1010 0101 0000 1111 (0xA50F): INVERTED KEY

52.3.2.3 Mode Enable Register (ME_ME)

Address: 0x008 Access: User read/write, Supervisor read/write, Test read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	RESET_DEST	0	0	0	0	STOP0	0	HALT0	RUN3	RUN2	RUN1	RUN0	DRUN	SAFE	TEST	RESET_FUNC
W																
Reset	1	0	0	0	0	0	0	0	0	0	0	1	1	1	0	1

Figure 1068. Mode Enable Register (ME_ME)

This register can disable the chip modes which are not required for a given chip. RESET_FUNC, SAFE, DRUN, RUN0, and RESET_DEST modes are always enabled.

Table 1036. ME_ME field descriptions

Field	Description
16 RESET_DEST	'Destructive' RESET mode enable 1 'Destructive' RESET mode is enabled
21 STOP0	STOP0 mode enable 0 STOP0 mode is disabled 1 STOP0 mode is enabled
23 HALT0	HALT0 mode enable 0 HALT0 mode is disabled 1 HALT0 mode is enabled
24 RUN3	RUN3 mode enable 0 RUN3 mode is disabled 1 RUN3 mode is enabled
25 RUN2	RUN2 mode enable 0 RUN2 mode is disabled 1 RUN2 mode is enabled
26 RUN1	RUN1 mode enable 0 RUN1 mode is disabled 1 RUN1 mode is enabled
27 RUN0	RUN0 mode enable 1 RUN0 mode is enabled
28 DRUN	DRUN mode enable 1 DRUN mode is enabled
29 SAFE	SAFE mode enable 1 SAFE mode is enabled
30 TEST	TEST mode enable 0 TEST mode is disabled 1 TEST mode is enabled
31 RESET_FUNC	'Functional' RESET mode enable 1 'Functional' RESET mode is enabled

52.3.2.4 Interrupt Status Register (ME_IS)

Address: 0x00C Access: User read/write, Supervisor read/write, Test read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	I_ICONF_CC	I_ICONF_CU	I_ICONF	I_IMODE	I_SAFE	I_MTC
W											w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1069. Interrupt Status Register (ME_IS)

This register provides the current interrupt status.

Table 1037. ME_IS field descriptions

Field	Description
26 I_ICONF_CC	Invalid mode configuration interrupt (core configuration) This bit is set if a write access to one of the ME_CCTL registers is attempted while a mode transition is in progress. 0 No write to an ME_CCTL register was attempted during an ongoing mode transition 1 A write to an ME_CCTL register was attempted during an ongoing mode transition
27 I_ICONF_CU	Invalid mode configuration interrupt (clock usage) This bit is set during a mode transition if a clock which is required to be on by an enabled peripheral is configured to be turned off. It is cleared by writing a ‘1’ to this bit. 0 No invalid mode configuration (clock usage) interrupt occurred 1 Invalid mode configuration (clock usage) interrupt is pending
28 I_ICONF	Invalid mode configuration interrupt This bit is set whenever a write operation to ME_<mode>_MC registers with invalid mode configuration is attempted. It is cleared by writing a ‘1’ to this bit. 0 No invalid mode configuration interrupt occurred 1 Invalid mode configuration interrupt is pending
29 I_IMODE	Invalid mode interrupt This bit is set whenever an invalid mode transition is requested. It is cleared by writing a ‘1’ to this bit. 0 No invalid mode interrupt occurred 1 Invalid mode interrupt is pending

Table 1037. ME_IS field descriptions (continued)

Field	Description
30 I_SAFE	SAFE mode interrupt This bit is set whenever the chip enters SAFE mode on hardware requests generated in the system. It is cleared by writing a '1' to this bit. 0 No SAFE mode interrupt occurred 1 SAFE mode interrupt is pending
31 I_MTC	Mode transition complete interrupt This bit is set whenever the mode transition process completes (S_MTRANS transits from 1 to 0). It is cleared by writing a '1' to this bit. This mode transition interrupt bit will not be set while entering low-power modes HALT0, or STOP0. 0 No mode transition complete interrupt occurred 1 Mode transition complete interrupt is pending

52.3.2.5 Interrupt Mask Register (ME_IM)

Address: 0x010 Access: User read/write, Supervisor read/write, Test read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	M_ICONF_CC	M_ICONF_CU	M_ICONF	M_IMODE	M_SAFE	M_MTC
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1070. Interrupt Mask Register (ME_IM)

This register controls whether an event generates an interrupt or not.

Table 1038. ME_IM field descriptions

Field	Description
26 M_ICONF_CC	Invalid mode configuration (core configuration) interrupt mask 0 Invalid mode interrupt is masked 1 Invalid mode interrupt is enabled
27 M_ICONF_CU	Invalid mode configuration (clock usage) interrupt mask 0 Invalid mode interrupt is masked 1 Invalid mode interrupt is enabled
28 M_ICONF	Invalid mode configuration interrupt mask 0 Invalid mode interrupt is masked 1 Invalid mode interrupt is enabled
29 M_IMODE	Invalid mode interrupt mask 0 Invalid mode interrupt is masked 1 Invalid mode interrupt is enabled

Table 1038. ME_IM field descriptions (continued)

Field	Description
30 M_SAFE	SAFE mode interrupt mask 0 SAFE mode interrupt is masked 1 SAFE mode interrupt is enabled
31 M_MTC	Mode transition complete interrupt mask 0 Mode transition complete interrupt is masked 1 Mode transition complete interrupt is enabled

52.3.2.6 Invalid Mode Transition Status Register (ME_IMTS)

Address: 0x014 Access: User read/write, Supervisor read/write, Test read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	S_MTI	S_MRI	S_DMA	S_NMA	S_SEA
W												w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1071. Invalid Mode Transition Status Register (ME_IMTS)

This register provides the status bits for the possible causes of an invalid mode interrupt.

Table 1039. ME_IMTS field descriptions

Field	Description
27 S_MTI	Mode Transition Illegal status This bit is set whenever a new mode is requested while some other mode transition process is active (S_MTRANS is ‘1’). Please refer to *Section 52.4.5* for the exceptions to this behavior. It is cleared by writing a ‘1’ to this bit. 0 Mode transition requested is not illegal 1 Mode transition requested is illegal
28 S_MRI	Mode Request Illegal status This bit is set whenever the target mode requested is not a valid mode with respect to current mode. It is cleared by writing a ‘1’ to this bit. 0 Target mode requested is not illegal with respect to current mode 1 Target mode requested is illegal with respect to current mode
29 S_DMA	Disabled Mode Access status This bit is set whenever the target mode requested is one of those disabled modes determined by ME_ME register. It is cleared by writing a ‘1’ to this bit. 0 Target mode requested is not a disabled mode 1 Target mode requested is a disabled mode

Table 1039. ME_IMTS field descriptions (continued)

Field	Description
30 S_NMA	Non-existing Mode Access status This bit is set whenever the target mode requested is one of those non existing modes determined by ME_ME register. It is cleared by writing a '1' to this bit. 0 Target mode requested is an existing mode 1 Target mode requested is a non-existing mode
31 S_SEA	SAFE Event Active status This bit is set whenever the chip is in SAFE mode, SAFE event bit is pending and a new mode requested other than RESET/SAFE modes. It is cleared by writing a '1' to this bit. 0 No new mode requested other than RESET/SAFE while SAFE event is pending 1 New mode requested other than RESET/SAFE while SAFE event is pending

52.3.2.7 Debug Mode Transition Status Register (ME_DMTS)

Address: 0x018 Access: User read/write, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	PREVIOUS_MODE				0	0	0	0	MPH_BUSY	0	0	PMC_PROG	DBG_MODE	CCKL_PROG	PCS_PROG	SMR
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	VREG_CSRC_SC	CSRC_CSRC_SC	IRCOSC_SC	SCSRC_SC	SYSCLK_SW	Reserved	FLASH_SC	CDP_PRPH_224_255	CDP_PRPH_192_223	CDP_PRPH_160_191	CDP_PRPH_128_159	CDP_PRPH_96_127	CDP_PRPH_64_95	CDP_PRPH_32_63	CDP_PRPH_0_31
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1072. Debug Mode Transition Status Register (ME_DMTS)

This register provides the status of different factors which influence mode transitions. It is used to give an indication of why a mode transition indicated by ME_GS[S_MTRANS] may be taking longer than expected.

Note: The ME_DMTS register does not indicate whether a mode transition is ongoing. Therefore, some ME_DMTS bits may still be asserted after the mode transition has completed.

Table 1040. ME_DMTS field descriptions

Field	Description
0:3 PREVIOUS_MODE	Previous chip mode These bits show the mode in which the chip was prior to the latest change to the current mode. 0000 RESET 0001 TEST 0010 SAFE 0011 DRUN 0100 RUN0 0101 RUN1 0110 RUN2 0111 RUN3 1000 HALT0 1001 Reserved 1010 STOP0 1011 Reserved 1100 Reserved 1101 Reserved 1110 Reserved 1111 Reserved
8 MPH_BUSY	MC_ME/MC_PCU Handshake Busy indicator This bit is set if the MC_ME has requested a mode change from the MC_PCU and the MC_PCU has not yet completed its power-up/down sequencing. It is cleared when the MC_PCU has power-up/down sequencing. 0 Handshake is not busy 1 Handshake is busy
11 PMC_PROG	MC_PCU Mode Change in Progress indicator This bit is set if the MC_PCU is in the process of powering up or down power domains. It is cleared when all power-up/down processes have completed. 0 Power-up/down transition is not in progress 1 Power-up/down transition is in progress
12 DBG_MODE	Debug mode indicator This bit is set while the chip is in debug mode. 0 The chip is not in debug mode 1 The chip is in debug mode
13 CCKL_PROG	Core Clock Enable/Disable in Progress This bit is set while any core's clock is in the process of being enabled or disabled. 0 No core clock is being enabled or disabled 1 A core clock is being enabled or disabled
14 PCS_PROG	Progressive System Clock Switching in Progress This bit is set while the progressive system clock switching process is in progress. 0 PCS is not in progress 1 PCS is in progress

Table 1040. ME_DMTS field descriptions (continued)

Field	Description
15 SMR	SAFE mode request from MC_RGM is active indicator This bit is set if a hardware SAFE mode request has been triggered. It is cleared when the hardware SAFE mode request has been cleared. 0 A SAFE mode request is not active 1 A SAFE mode request is active
17 VREG_CSRC_SC	Main VREG dependent Clock Source State Change during mode transition indicator This bit is set when a clock source which depends on the main voltage regulator to be powered-up is requested to change its power up/down state. It is cleared when the clock source has completed its state change. 0 No state change is taking place 1 A state change is taking place
18 CSRC_CSRC_SC	(Other) Clock Source dependent Clock Source State Change during mode transition indicator This bit is set when a clock source which depends on another clock source to be powered-up is requested to change its power up/down state. It is cleared when the clock source has completed its state change. 0 No state change is taking place 1 A state change is taking place
19 IRCOSC_SC	IRCOSC State Change during mode transition indicator This bit is set when the 16 MHz Internal RC oscillator is requested to change its power up/down state. It is cleared when the 16 MHz Internal RC oscillator has completed its state change. 0 No state change is taking place 1 A state change is taking place
20 SCSRC_SC	Secondary Clock Sources State Change during mode transition indicator This bit is set when a secondary clock source is requested to change its power up/down state. It is cleared when all secondary system clock sources have completed their state changes. (A ‘secondary clock source’ is a clock source other than IRCOSC.) 0 No state change is taking place 1 A state change is taking place
21 SYSCLK_SW	System Clock Switching pending status 0 No system clock source switching is pending 1 A system clock source switching is pending
23 FLASH_SC	FLASH State Change during mode transition indicator This bit is set when the FLASH is requested to change its power up/down state. It is cleared when the DFLASH has completed its state change. 0 No state change is taking place 1 A state change is taking place
24 CDP_PRPH_224_255	Clock Disable Process Pending status for Peripherals 224–255(1) This bit is set when any peripheral has been requested to have its clock disabled. It is cleared when all the peripherals which have been requested to have their clocks disabled have entered the state in which their clocks may be disabled. 0 No peripheral clock disabling is pending 1 Clock disabling is pending for at least one peripheral

Table 1040. ME_DMTS field descriptions (continued)

Field	Description
25 CDP_PRPH_192_223	Clock Disable Process Pending status for Peripherals 192–223(1) This bit is set when any peripheral appearing in ME_PS4 has been requested to have its clock disabled. It is cleared when all these peripherals which have been requested to have their clocks disabled have entered the state in which their clocks may be disabled. 0 No peripheral clock disabling is pending 1 Clock disabling is pending for at least one peripheral
26 CDP_PRPH_160_191	Clock Disable Process Pending status for Peripherals 160–191(1) This bit is set when any peripheral appearing in ME_PS4 has been requested to have its clock disabled. It is cleared when all these peripherals which have been requested to have their clocks disabled have entered the state in which their clocks may be disabled. 0 No peripheral clock disabling is pending 1 Clock disabling is pending for at least one peripheral
27 CDP_PRPH_128_159	Clock Disable Process Pending status for Peripherals 128–159(1) This bit is set when any peripheral appearing in ME_PS4 has been requested to have its clock disabled. It is cleared when all these peripherals which have been requested to have their clocks disabled have entered the state in which their clocks may be disabled. 0 No peripheral clock disabling is pending 1 Clock disabling is pending for at least one peripheral
28 CDP_PRPH_96_127	Clock Disable Process Pending status for Peripherals 96–127(1) This bit is set when any peripheral appearing in ME_PS3 has been requested to have its clock disabled. It is cleared when all these peripherals which have been requested to have their clocks disabled have entered the state in which their clocks may be disabled. 0 No peripheral clock disabling is pending 1 Clock disabling is pending for at least one peripheral
29 CDP_PRPH_64_95	Clock Disable Process Pending status for Peripherals 64–95(1) This bit is set when any peripheral appearing in ME_PS2 has been requested to have its clock disabled. It is cleared when all these peripherals which have been requested to have their clocks disabled have entered the state in which their clocks may be disabled. 0 No peripheral clock disabling is pending 1 Clock disabling is pending for at least one peripheral

Table 1040. ME_DMTS field descriptions (continued)

Field	Description
30 CDP_PRPH_32_63	Clock Disable Process Pending status for Peripherals 32–63(1) This bit is set when any peripheral appearing in ME_PS1 has been requested to have its clock disabled. It is cleared when all these peripherals which have been requested to have their clocks disabled have entered the state in which their clocks may be disabled. 0 No peripheral clock disabling is pending 1 Clock disabling is pending for at least one peripheral
31 CDP_PRPH_0_31	Clock Disable Process Pending status for Peripherals 0–31(1) This bit is set when any peripheral appearing in ME_PS0 has been requested to have its clock disabled. It is cleared when all these peripherals which have been requested to have their clocks disabled have entered the state in which their clocks may be disabled. 0 No peripheral clock disabling is pending 1 Clock disabling is pending for at least one peripheral

1. Peripheral n corresponds to the ME_PCTL*n* register. Please refer to *Table 1033* for the ME_PCTL*n* locations actually occupied, which in turn indicates which peripherals are reported in the ME_DMTS register.

52.3.2.8 RESET Mode Configuration Register (ME_RESET_MC)

Address: 0x020 Access: User read/write, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	PWRLVL			0	0	0	0	PDO	0	0	MVRON	0	0	FLAON	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	1

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	PLL1ON	PLL0ON	XOSCON	IRCOSCON	SYSCLK			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0

Figure 1073. RESET Mode Configuration Register (ME_RESET_MC)

This register configures system behavior during RESET mode. Please refer to *Table 1041* for details.

Note: Half word access is only allowed to the lower half word of the register.

Table 1041. ME_RESET_MC field descriptions

Field	Description
1:3 PWRLVL	Power level These bits indicate the relative power consumption level of this mode with respect to that of other modes. When switching between two modes with differing power levels, system clock progressive switching is enabled.
8 PDO	I/O output power-down control This bit controls the output power-down of I/Os. 0 No automatic safe gating of I/Os used and pads power sequence driver is enabled 1 In SAFE/TEST modes, outputs of pads are forced to high impedance state and pads power sequence driver is disabled. The inputs are level unchanged. In STOP0 mode, only the pad power sequence driver is disabled, but the state of the output remains functional.
11 MVRON	Main voltage regulator control This bit specifies whether main voltage regulator is switched off or not while entering this mode. 1 Main voltage regulator is switched on
14:15 FLAON	Flash power-down control This bit specifies the operating mode of the code flash after entering this mode. 00 Reserved 01 Flash is in power-down mode 10 Flash is in low-power mode 11 Flash is in normal mode
24 PLL1ON	Secondary PLL control 0 Secondary PLL is switched off 1 Secondary PLL is switched on
25 PLL0ON	Primary PLL control 0 Primary PLL is switched off 1 Primary PLL is switched on
26 XOSCON	External crystal oscillator control 0 External crystal oscillator is switched off 1 External crystal oscillator is switched on

Table 1041. ME_RESET_MC field descriptions (continued)

Field	Description
27 IRCOSCON	16 MHz Internal RC oscillator control 0 16 MHz Internal RC oscillator is switched off 1 16 MHz Internal RC oscillator is switched on
28:31 SYSCLK	System clock switch control These bits specify the system clock to be used by the system. 0000 16 MHz Internal RC oscillator 0001 external crystal oscillator 0010 primary PLL (PHI) 0011 Reserved 0100 Secondary PLL 0101 Reserved 0110 Reserved 0111 Reserved 1000 Reserved 1001 Reserved 1010 Reserved 1011 Reserved 1100 Reserved 1101 Reserved 1110 Reserved 1111 System clock is disabled in TEST mode, reserved in all other modes

Note: The following configuration values are set according to the chip configuration: XOSCON.
Half word access is only allowed to the lower half word of the register.

52.3.2.9 TEST Mode Configuration Register (ME_TEST_MC)

Address: 0x024 Access: User read/write, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	PWRLVL			0	0	0	0	PDO	0	0	MVRON	0	0	FLAON	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	1

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	PLL1ON	PLL0ON	XOSCON	IRCOSCON	SYSCLK			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0

Figure 1074. TEST Mode Configuration Register (ME_TEST_MC)

This register configures system behavior during TEST mode. Please refer to *Table 1042* for details.

Note: Half word access is only allowed to the lower half word of the register.

Table 1042. ME_TEST_MC field descriptions

Field	Description
1:3 PWRLVL	Power level These bits indicate the relative power consumption level of this mode with respect to that of other modes. When switching between two modes with differing power levels, system clock progressive switching is enabled.
8 PDO	I/O output power-down control This bit controls the output power-down of I/Os. 0 No automatic safe gating of I/Os used and pads power sequence driver is enabled 1 In SAFE/TEST modes, outputs of pads are forced to high impedance state and pads power sequence driver is disabled. The inputs are level unchanged. In STOP0 mode, only the pad power sequence driver is disabled, but the state of the output remains functional.
11 MVRON	Main voltage regulator control This bit specifies whether main voltage regulator is switched off or not while entering this mode. 1 Main voltage regulator is switched on
14:15 FLAON	Flash power-down control This bit specifies the operating mode of the code flash after entering this mode. 00 Reserved 01 Flash is in power-down mode 10 Flash is in low-power mode 11 Flash is in normal mode
24 PLL1ON	Secondary PLL control 0 Secondary PLL is switched off 1 Secondary PLL is switched on
25 PLL0ON	Primary PLL control 0 Primary PLL is switched off 1 Primary PLL is switched on
26 XOSCON	External crystal oscillator control 0 External crystal oscillator is switched off 1 External crystal oscillator is switched on

Table 1042. ME_TEST_MC field descriptions (continued)

Field	Description
27 IRCOSCON	16 MHz Internal RC oscillator control 0 16 MHz Internal RC oscillator is switched off 1 16 MHz Internal RC oscillator is switched on
28:31 SYSCLK	System clock switch control These bits specify the system clock to be used by the system. 0000 16 MHz Internal RC oscillator 0001 external crystal oscillator 0010 primary PLL (PHI) 0011 Reserved 0100 Secondary PLL 0101 Reserved 0110 Reserved 0111 Reserved 1000 Reserved 1001 Reserved 1010 Reserved 1011 Reserved 1100 Reserved 1101 Reserved 1110 Reserved 1111 System clock is disabled in TEST mode, reserved in all other modes

52.3.2.10 SAFE Mode Configuration Register (ME_SAFE_MC)

Address: 0x028 Access: User read/write, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	PWRLVL			0	0	0	0	PDO	0	0	MVRON	0	0	FLAON	
W																
Reset	0	0	0	0	0	0	0	0	1	0	0	1	0	0	1	1

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	PLL1ON	PLL0ON	XOSCON	IRCOSCON	SYSCLK			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0

Figure 1075. SAFE Mode Configuration Register (ME_SAFE_MC)

This register configures system behavior during SAFE mode. Please refer to *Table 1043* for details.

Note: Half word access is only allowed to the lower half word of the register.

Table 1043. ME_SAFE_MC field descriptions

Field	Description
1:3 PWRLVL	Power level These bits indicate the relative power consumption level of this mode with respect to that of other modes. When switching between two modes with differing power levels, system clock progressive switching is enabled.
8 PDO	I/O output power-down control This bit controls the output power-down of I/Os. 0 No automatic safe gating of I/Os used and pads power sequence driver is enabled 1 In SAFE/TEST modes, outputs of pads are forced to high impedance state and pads power sequence driver is disabled. The inputs are level unchanged. In STOP0 mode, only the pad power sequence driver is disabled, but the state of the output remains functional.
11 MVRON	Main voltage regulator control This bit specifies whether main voltage regulator is switched off or not while entering this mode. 1 Main voltage regulator is switched on
14:15 FLAON	Flash power-down control This bit specifies the operating mode of the code flash after entering this mode. 00 reserved 01 Flash is in power-down mode 10 Flash is in low-power mode 11 Flash is in normal mode
24 PLL1ON	Secondary PLL control 0 Secondary PLL is switched off 1 Secondary PLL is switched on
25 PLL0ON	Primary PLL control 0 Primary PLL is switched off 1 Primary PLL is switched on
26 XOSCON	External crystal oscillator control 0 External crystal oscillator is switched off 1 External crystal oscillator is switched on

Table 1043. ME_SAFE_MC field descriptions (continued)

Field	Description
27 IRCOSCON	16 MHz Internal RC oscillator control 0 16 MHz Internal RC oscillator is switched off 1 16 MHz Internal RC oscillator is switched on
28:31 SYSCLK	System clock switch control These bits specify the system clock to be used by the system. 0000 16 MHz Internal RC oscillator 0001 external crystal oscillator 0010 primary PLL (PHI) 0011 Reserved 0100 Secondary PLL 0101 Reserved 0110 Reserved 0111 Reserved 1000 Reserved 1001 Reserved 1010 Reserved 1011 Reserved 1100 Reserved 1101 Reserved 1110 Reserved 1111 System clock is disabled in TEST mode, reserved in all other modes

52.3.2.11 DRUN Mode Configuration Register (ME_DRUN_MC)

Address: 0x02C Access: User read/write, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	PWRLVL			0	0	0	0	PDO	0	0	MVRON	0	0	FLAON	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	1

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	PLL1ON	PLL0ON	XOSCON	IRCOSCON	SYSCLK			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0

Figure 1076. DRUN Mode Configuration Register (ME_DRUN_MC)

This register configures system behavior during DRUN mode. Please refer to *Table 1044* for details.

Note: Half word access is only allowed to the lower half word of the register.

Table 1044. ME_DRUN_MC field descriptions

Field	Description
1:3 PWRLVL	Power level These bits indicate the relative power consumption level of this mode with respect to that of other modes. When switching between two modes with differing power levels, system clock progressive switching is enabled.
8 PDO	I/O output power-down control This bit controls the output power-down of I/Os. 0 No automatic safe gating of I/Os used and pads power sequence driver is enabled 1 In SAFE/TEST modes, outputs of pads are forced to high impedance state and pads power sequence driver is disabled. The inputs are level unchanged. In STOP0 mode, only the pad power sequence driver is disabled, but the state of the output remains functional.
11 MVRON	Main voltage regulator control This bit specifies whether main voltage regulator is switched off or not while entering this mode. 1 Main voltage regulator is switched on
14:15 FLAON	Flash power-down control This bit specifies the operating mode of the code flash after entering this mode. 00 reserved 01 Flash is in power-down mode 10 Flash is in low-power mode 11 Flash is in normal mode
24 PLL1ON	Secondary PLL control 0 Secondary PLL is switched off 1 Secondary PLL is switched on
25 PLL0ON	Primary PLL control 0 Primary PLL is switched off 1 Primary PLL is switched on
26 XOSCON	External crystal oscillator control 0 External crystal oscillator is switched off 1 External crystal oscillator is switched on

Table 1044. ME_DRUN_MC field descriptions (continued)

Field	Description
27 IRCOSCON	16 MHz Internal RC oscillator control 0 16 MHz Internal RC oscillator is switched off 1 16 MHz Internal RC oscillator is switched on
28:31 SYSCLK	System clock switch control These bits specify the system clock to be used by the system. 0000 16 MHz Internal RC oscillator 0001 external crystal oscillator 0010 primary PLL (PHI) 0011 Reserved 0100 Secondary PLL 0101 Reserved 0110 Reserved 0111 Reserved 1000 Reserved 1001 Reserved 1010 Reserved 1011 Reserved 1100 Reserved 1101 Reserved 1110 Reserved 1111 System clock is disabled in TEST mode, reserved in all other modes

The following configuration values are set according to the chip configuration: XOSCON.

52.3.2.12 RUN*n* Mode Configuration Register (ME_RUN*n*_MC)

Address: 0x030 + *n**0x4 (*n* = 0 to 3) Access: User read/write, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	PWRLVL			0	0	0	0	PDO	0	0	MVRON	0	0	FLAON	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	1

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	PLL1ON	PLL0ON	XOSCON	IRCOSCON	SYSCLK			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0

Figure 1077. RUN*n* Mode Configuration Register (ME_RUN*n*_MC)

This register configures system behavior during RUN0–3 modes. Please refer to *Table 1045* for details.

Note: Half word access is only allowed to the lower half word of the register.

Table 1045. ME_RUN*n*_MC field descriptions

Field	Description
1:3 PWRLVL	Power level These bits indicate the relative power consumption level of this mode with respect to that of other modes. When switching between two modes with differing power levels, system clock progressive switching is enabled.
8 PDO	I/O output power-down control This bit controls the output power-down of I/Os. 0 No automatic safe gating of I/Os used and pads power sequence driver is enabled 1 In SAFE/TEST modes, outputs of pads are forced to high impedance state and pads power sequence driver is disabled. The inputs are level unchanged. In STOP0 mode, only the pad power sequence driver is disabled, but the state of the output remains functional.
11 MVRON	Main voltage regulator control This bit specifies whether main voltage regulator is switched off or not while entering this mode. 1 Main voltage regulator is switched on
14:15 FLAON	Flash power-down control This bit specifies the operating mode of the code flash after entering this mode. 00 Reserved 01 Flash is in power-down mode 10 Flash is in low-power mode 11 Flash is in normal mode
24 PLL1ON	Secondary PLL control 0 Secondary PLL is switched off 1 Secondary PLL is switched on
25 PLL0ON	Primary PLL control 0 Primary PLL is switched off 1 Primary PLL is switched on
26 XOSCON	External crystal oscillator control 0 External crystal oscillator is switched off 1 External crystal oscillator is switched on

Table 1045. ME_RUN*n*_MC field descriptions (continued)

Field	Description
27 IRCOSCON	16 MHz Internal RC oscillator control 0 16 MHz Internal RC oscillator is switched off 1 16 MHz Internal RC oscillator is switched on
28:31 SYSCLK	System clock switch control These bits specify the system clock to be used by the system. 0000 16 MHz Internal RC oscillator 0001 external crystal oscillator 0010 primary PLL (PHI) 0011 Reserved 0100 Secondary PLL 0101 Reserved 0110 Reserved 0111 Reserved 1000 Reserved 1001 Reserved 1010 Reserved 1011 Reserved 1100 Reserved 1101 Reserved 1110 Reserved 1111 System clock is disabled in TEST mode, reserved in all other modes

52.3.2.13 HALT0 Mode Configuration Register (ME_HALT0_MC)

Address: 0x040 Access: User read/write, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	PWRLVL			0	0	0	0	PDO	0	0	MVRON	0	0	FLAON	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	PLL1ON	PLL0ON	XOSCON	IRCOSCON	SYSCLK			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0

Figure 1078. HALT0 Mode Configuration Register (ME_HALT0_MC)

This register configures system behavior during HALT0 mode. Please refer to *Table 1046* for details.

Note: Half word access is only allowed to the lower half word of the register.

Table 1046. ME_HALT0_MC field descriptions

Field	Description
1:3 PWRLVL	Power level These bits indicate the relative power consumption level of this mode with respect to that of other modes. When switching between two modes with differing power levels, system clock progressive switching is enabled.
8 PDO	I/O output power-down control This bit controls the output power-down of I/Os. 0 No automatic safe gating of I/Os used and pads power sequence driver is enabled 1 In SAFE/TEST modes, outputs of pads are forced to high impedance state and pads power sequence driver is disabled. The inputs are level unchanged. In STOP0 mode, only the pad power sequence driver is disabled, but the state of the output remains functional.
11 MVRON	Main voltage regulator control This bit specifies whether main voltage regulator is switched off or not while entering this mode. 1 Main voltage regulator is switched on
14:15 FLAON	Flash power-down control This bit specifies the operating mode of the code flash after entering this mode. 00 reserved 01 Flash is in power-down mode 10 Flash is in low-power mode 11 Flash is in normal mode
24 PLL1ON	Secondary PLL control 0 Secondary PLL is switched off 1 Secondary PLL is switched on
25 PLL0ON	Primary PLL control 0 Primary PLL is switched off 1 Primary PLL is switched on
26 XOSCON	External crystal oscillator control 0 External crystal oscillator is switched off 1 External crystal oscillator is switched on

Table 1046. ME_HALT0_MC field descriptions (continued)

Field	Description
27 IRCOSCON	16 MHz Internal RC oscillator control 0 16 MHz Internal RC oscillator is switched off 1 16 MHz Internal RC oscillator is switched on
28:31 SYSCLK	System clock switch control These bits specify the system clock to be used by the system. 0000 16 MHz Internal RC oscillator 0001 external crystal oscillator 0010 primary PLL (PHI) 0011 Reserved 0100 Secondary PLL 0101 Reserved 0110 Reserved 0111 Reserved 1000 Reserved 1001 Reserved 1010 Reserved 1011 Reserved 1100 Reserved 1101 Reserved 1110 Reserved 1111 system clock is disabled in TEST mode, reserved in all other modes

52.3.2.14 STOP0 Mode Configuration Register (ME_STOP0_MC)

Address: 0x048 Access: User read/write, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	PWRLVL			0	0	0	0	PDO	0	0	MVRON	0	0	FLAON	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	PLL1ON	PLL0ON	XOSCON	IRCOSCON	SYSCLK			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0

Figure 1079. STOP0 Mode Configuration Register (ME_STOP0_MC)

This register configures system behavior during STOP0 mode. Please refer to *Table 1047* for details.

Note: Half word access is only allowed to the lower half word of the register.

Table 1047. ME_STOP0_MC field descriptions

Field	Description
1:3 PWRLVL	Power level These bits indicate the relative power consumption level of this mode with respect to that of other modes. When switching between two modes with differing power levels, system clock progressive switching is enabled.
8 PDO	I/O output power-down control This bit controls the output power-down of I/Os. 0 No automatic safe gating of I/Os used and pads power sequence driver is enabled 1 In SAFE/TEST modes, outputs of pads are forced to high impedance state and pads power sequence driver is disabled. The inputs are level unchanged. In STOP0 mode, only the pad power sequence driver is disabled, but the state of the output remains functional.
11 MVRON	Main voltage regulator control This bit specifies whether main voltage regulator is switched off or not while entering this mode. 1 Main voltage regulator is switched on
14:15 FLAON	Flash power-down control This bit specifies the operating mode of the code flash after entering this mode. 00 reserved 01 Flash is in power-down mode 10 Flash is in low-power mode 11 Flash is in normal mode
24 PLL1ON	Secondary PLL control 0 Secondary PLL is switched off 1 Secondary PLL is switched on
25 PLL0ON	Primary PLL control 0 Primary PLL is switched off 1 Primary PLL is switched on
26 XOSCON	External crystal oscillator control 0 External crystal oscillator is switched off 1 External crystal oscillator is switched on

Table 1047. ME_STOP0_MC field descriptions (continued)

Field	Description
27 IRCOSCON	16 MHz Internal RC oscillator control 0 16 MHz Internal RC oscillator is switched off 1 16 MHz Internal RC oscillator is switched on
28:31 SYSCLK	System clock switch control These bits specify the system clock to be used by the system. 0000 16 MHz Internal RC oscillator 0001 external crystal oscillator 0010 primary PLL (PHI) 0011 Reserved 0100 Secondary PLL 0101 Reserved 0110 Reserved 0111 Reserved 1000 Reserved 1001 Reserved 1010 Reserved 1011 Reserved 1100 Reserved 1101 Reserved 1110 Reserved 1111 system clock is disabled in TEST mode, reserved in all other modes

52.3.2.15 Peripheral Status Register 0 (ME_PS0)

Address: 0x060 Access: User read, Supervisor read, Test read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	S_PIT_RTI_0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

Figure 1080. Peripheral Status Register 0 (ME_PS0)

This register provides the status of the peripherals. Please refer to *Table 1048* for details.

52.3.2.16 Peripheral Status Register 1 (ME_PS1)

Address: 0x064 Access: User read, Supervisor read, Test read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	S_CCCU_0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	S_CRC_0	0	S_DMA_CH_MUX	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1081. Peripheral Status Register 1 (ME_PS1)

This register provides the status of the peripherals. Please refer to *Table 1048* for details.

52.3.2.17 Peripheral Status Register 2 (ME_PS2)

Address 0x068 Access: User read, Supervisor read, Test read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	S_LINFlexD_0	S_LINFlexD_2	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	MCAN_RAM_0	0	0	0	MCAN_0	MCAN_2	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1082. Peripheral Status Register 2 (ME_PS2)

This register provides the status of the peripherals. Please refer to *Table 1048* for details.

52.3.2.18 Peripheral Status Register 3 (ME_PS3)

Address: 0x06C Access: User read, Supervisor read, Test read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	S_ADCSAR_0	S_ADCSAR_1	S_ADCSAR_2	0	S_ADC_SEQ_0	S_ADC_SEQ_1	S_ADC_SEQ_2	0	0	0	0	0	0	0	0	S_ADCS_AIPS_0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	S_ADCS_SEQ_AIPS_0	0	0	0	S_FlexRay	0	0	S_SENT_0	0	0	0	0	S_DSPI_0	S_DSPI_1	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1083. Peripheral Status Register 3 (ME_PS3)

This register provides the status of the peripherals. Please refer to *Table 1048* for details.

52.3.2.19 Peripheral Status Register 4 (ME_PS4)

Address: 0x070 Access: User read, Supervisor read, Test read

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	S_PWM_SYNCHR	0	S_FlexPWM_2	S_FlexPWM_3	S_FlexPWM_0	S_FlexPWM_1	0	S_CTU_1	0	0	0	0	S_eTimer_2	S_eTimer_3	S_eTimer_0	S_eTimer_1
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1084. Peripheral Status Register 4 (ME_PS4)

This register provides the status of the peripherals. Please refer to *Table 1048* for details.

52.3.2.20 Peripheral Status Register 5 (ME_PS5)

Address: 0x074 Access: User read, Supervisor read, Test read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	S_CCCU_1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

Figure 1085. Peripheral Status Register 5 (ME_PS5)

This register provides the status of the peripherals. Please refer to *Table 1048* for details.

52.3.2.21 Peripheral Status Register 6 (ME_PS6)

Address: 0x078 Access: User read, Supervisor read, Test read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	S_LINFlexD_1	S_LINFlexD_3	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	S_MCAN_RAM_1	0	0	0	S_M_CAN_1	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1086. Peripheral Status Register 6 (ME_PS6)

This register provides the status of the peripherals. Please refer to *Table 1048* for details.

52.3.2.22 Peripheral Status Register 7 (ME_PS7)

Address: 0x07C Access: User read, Supervisor read, Test read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	S_ADC_3	S_ADC_4	S_ADC_5	0	S_ADC_SEQ_3	S_ADC_SEQ_4	S_ADC_SEQ_5	0	0	0	0	S_CTU_0	0	0	0	S_ADCS_AIPS_1
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	S_ADCS_SEQ_AIPS_1	0	0	0	0	0	0	S_SENT_1	0	0	0	0	S_DSPI_2	S_DSPI_3	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1087. Peripheral Status Register 7 (ME_PS7)

This register provides the status of the peripherals. Please refer to *Table 1048* for details.

Table 1048. ME_PS*n* field descriptions

Field	Description
S_<*periph*>	Peripheral status These bits specify the current status of each peripheral which is controlled by the MC_ME. 0 Peripheral is frozen 1 Peripheral is active

52.3.2.23 Run Peripheral Configuration Register *n* (ME_RUN_PC*n*)

Address: 0x080 + *n**0x4 (*n* = 0 to 7) Access: User read/write, Supervisor read/write, Test read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	RUN3	RUN2	RUN1	RUN0	DRUN	SAFE	TEST	RESET
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1088. Run Peripheral Configuration Registers n (ME_RUN_PC*n*)

These registers configure eight different types of peripheral behavior during run modes.

Table 1049. ME_RUN_PC*n* field descriptions

Field	Description
24 RUN3	Peripheral control during RUN3 0 Peripheral is frozen with clock gated 1 Peripheral is active
25 RUN2	Peripheral control during RUN2 0 Peripheral is frozen with clock gated 1 Peripheral is active
26 RUN1	Peripheral control during RUN1 0 Peripheral is frozen with clock gated 1 Peripheral is active
27 RUN0	Peripheral control during RUN0 0 Peripheral is frozen with clock gated 1 Peripheral is active
28 DRUN	Peripheral control during DRUN 0 Peripheral is frozen with clock gated 1 Peripheral is active
29 SAFE	Peripheral control during SAFE 0 Peripheral is frozen with clock gated 1 Peripheral is active
30 TEST	Peripheral control during TEST 0 Peripheral is frozen with clock gated 1 Peripheral is active
31 RESET	Peripheral control during RESET 0 Peripheral is frozen with clock gated 1 Peripheral is active

52.3.2.24 Low-Power Peripheral Configuration Register *n* (ME_LP_PC*n*)

Address: 0x0A0 + *n**0x4 (*n* = 0 to 7) Access: User read/write, Supervisor read/write, Test read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	STOP0	0	HALT0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1089. Low-Power Peripheral Configuration Register *n* (ME_LP_PC*n*)

These registers configure eight different types of peripheral behavior during non-run modes.

Table 1050. ME_LP_PC*n* field descriptions

Field	Description
21 STOP0	Peripheral control during STOP0 0 Peripheral is frozen with clock gated 1 Peripheral is active
23 HALT0	Peripheral control during HALT0 0 Peripheral is frozen with clock gated 1 Peripheral is active

52.3.2.25 Peripheral Control Register *n* (ME_PCTL*n*)

Address: 0x0C0 + *n**0x1 (*n* = 0 to 255) Access: User read/write, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7
R	0	0(1)	LP_CFG			RUN_CFG		
W								
Reset	0	0	0	0	0	0	0	0

1. This bit is writable but has no impact.

Figure 1090. Peripheral Control Register *n* (ME_PCTL*n*)

These registers select the configurations during run and non-run modes for each peripheral. The unoccupied locations contain a read-only byte value of 0x00.

Table 1051. ME_PCTL*n* field descriptions

Field	Description
2:4 LP_CFG	Peripheral configuration select for non-run modes These bits associate a configuration as defined in the ME_LP_PC0–7 registers to the peripheral. 000 Selects ME_LP_PC0 configuration 001 Selects ME_LP_PC1 configuration 010 Selects ME_LP_PC2 configuration 011 Selects ME_LP_PC3 configuration 100 Selects ME_LP_PC4 configuration 101 Selects ME_LP_PC5 configuration 110 Selects ME_LP_PC6 configuration 111 Selects ME_LP_PC7 configuration
5:7 RUN_CFG	Peripheral configuration select for run modes These bits associate a configuration as defined in the ME_RUN_PC0–7 registers to the peripheral. 000 Selects ME_RUN_PC0 configuration 001 Selects ME_RUN_PC1 configuration 010 Selects ME_RUN_PC2 configuration 011 Selects ME_RUN_PC3 configuration 100 Selects ME_RUN_PC4 configuration 101 Selects ME_RUN_PC5 configuration 110 Selects ME_RUN_PC6 configuration 111 Selects ME_RUN_PC7 configuration

52.3.2.26 Core Status Register (ME_CS)

Address: 0x1C0 Access: User read, Supervisor read, Test read

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	S_CORE0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1091. Core Status Register (ME_CS)

This register provides the status of each core.

Table 1052. ME_CS field descriptions

Field	Description
31 S_CORE0	Core status 0 This bits specifies the current status of main core_0 which is controlled by the ME_CCTL0 register. 0 Main core_0 is disabled 1 Main core_0 is running

52.3.2.27 Core Control Register (ME_CCTL0)

Address: 0x1C4 Access: User read-only, Supervisor read-only, Test read-only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	STOP0	0	HALT0	RUN3	RUN2	RUN1	RUN0	DRUN	SAFE	TEST	RESET
W																
Reset	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	0

Figure 1092. Core Control Register (ME_CCTL0)

This register controls whether main core_0 is disabled or running during run modes.

This register cannot be written after a mode change request has been made until the mode transition has completed (that is, while the bit ME_GS[S_MTRANS] = ‘1’). A write access to this register during this time will result in the ICONF_CC flag in the ME_IS register being asserted.

Table 1053. ME_CCTL0 field descriptions

Field	Description
5 STOP0	Core control during STOP0 Main core_0 is always disabled during STOP0.
7 HALT0	Core control during HALT0 Main core_0 is always disabled during HALT0.
8 RUN3	Core control during RUN3 0 Main core_0 is disabled with clock gated 1 Main core_0 is running
9 RUN2	Core control during RUN2 0 Main core_0 is disabled with clock gated 1 Main core_0 is running
10 RUN1	Core control during RUN1 0 Main core_0 is disabled with clock gated 1 Main core_0 is running
11 RUN0	Core control during RUN0 0 Main core_0 is frozen with clock gated 1 Main core_0 is running

Table 1053. ME_CCTL0 field descriptions (continued)

Field	Description
12 DRUN	Core control during DRUN 0 Main core_0 is frozen with clock gated 1 Main core_0 is running
13 SAFE	Core control during SAFE 0 Main core_0 is frozen with clock gated 1 Main core_0 is running
14 TEST	Core control during TEST 0 Main core_0 is frozen with clock gated 1 Main core_0 is running
15 RESET	Core control during RESET Main core_0 is always disabled during RESET.

52.3.2.28 Core Address Register (ME_CADDR0)

Address: 0x1E0 Access: User read/write, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	ADDR															
Reset	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	ADDR														0	RMC
Reset	X	X	X	X	X	X	X	X	X	X	X	X	X	X	0	0

Figure 1093. Core Address Register (ME_CADDR0)

This register gives the boot address for main core_0 and a bit for controlling whether main core_0 is to be reset on the next mode change that has main core_0 configured to be running in the target mode.

This register can be written only as a word and cannot be written after a mode change request has been made until the mode transition has completed (that is, while the bit ME_GS[S_MTRANS] register = ‘1’). A write access to this register during this time will result in the ICONF_CC flag in the ME_IS register being asserted.

Note: The reset value of the field ME_CADDR0[ADDR] is determined by the chip configuration and boot mode.

Table 1054. ME_CADDR0 field descriptions

Field	Description
0:29 ADDR	Core Address This field is used by main core_0 as the boot address (32-bit word aligned) when main core_0 next exits reset.
31 RMC	Reset on Mode Change Main core_0 will be reset on the next mode change that has main core_0 configured to be running in the target mode. 0 Main core_0 will not be reset on the next mode change 1 Main core_0 will be reset on the next mode change that has main core_0 configured to be running in the target mode

52.4 Functional description

52.4.1 Mode Transition Request

The transition from one mode to another mode is normally handled by software by accessing the mode control register ME_MCTL. But in the case of special events, the mode transition can be automatically managed by hardware. In order to switch from one mode to another, the application should access the ME_MCTL register twice by writing

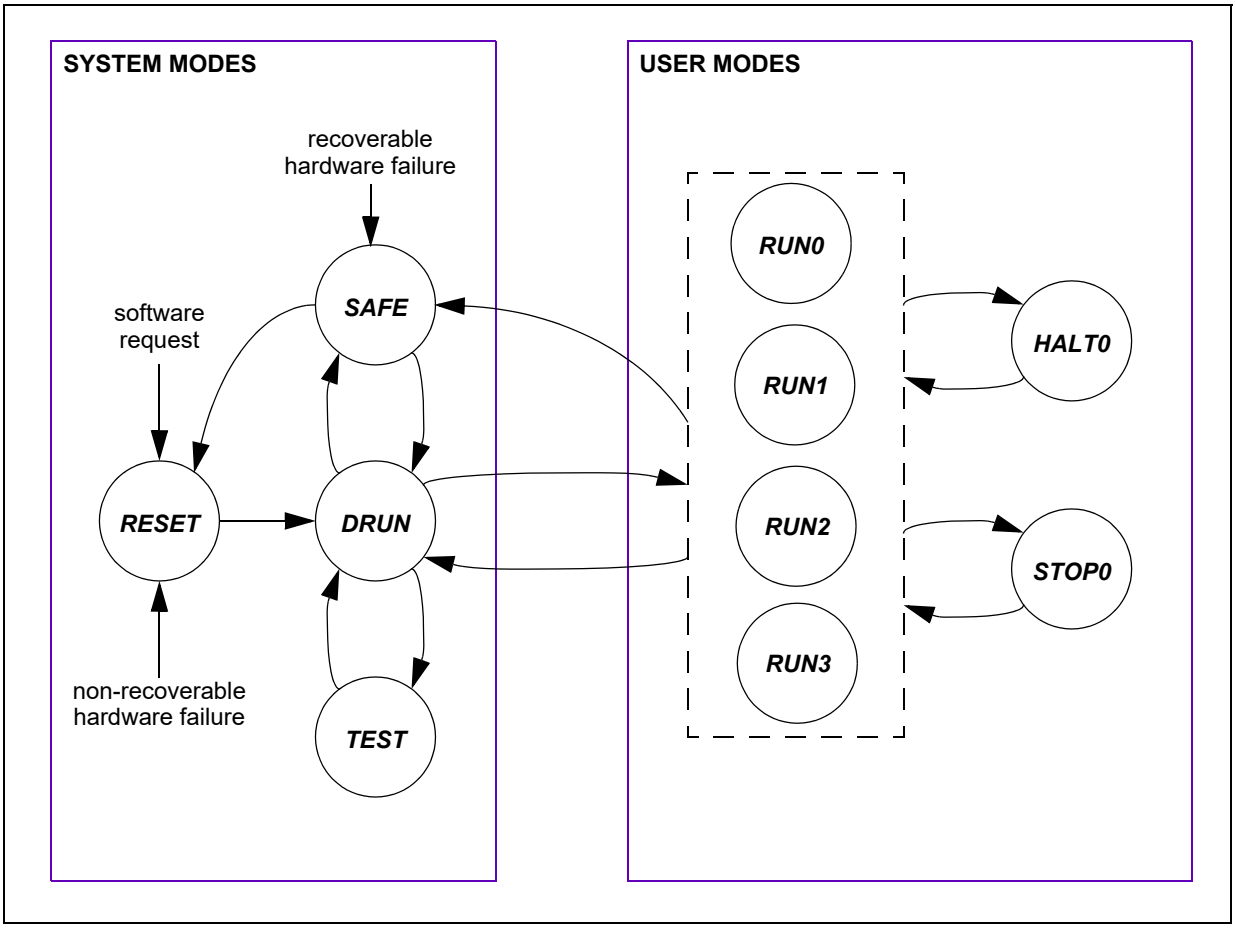
- the first time with the value of the key (0x5AF0) into the KEY bit field and the required target mode into the TARGET_MODE bit field,
- and the second time with the inverted value of the key (0xA50F) into the KEY bit field and the required target mode into the TARGET_MODE bit field.

Once a valid mode transition request is detected, the target mode configuration information is loaded from the corresponding ME_<*mode*>_MC register. The mode transition request may require a number of cycles depending on the programmed configuration, and software should check the S_CURRENT_MODE bit field and the S_MTRANS bit of the global status register ME_GS to verify when the mode has been correctly entered and the transition process has completed. For a description of valid mode requests, please refer to *Section 52.4.5*.

Any modification of the mode configuration register of the currently selected mode will not be taken into account immediately but on the next request to enter this mode. This means that transition requests such as RUN0–3 to RUN0–3, DRUN to DRUN, SAFE to SAFE, and TEST to TEST are considered valid mode transition requests. As soon as the mode request is accepted as valid, the S_MTRANS bit is set till the status in the ME_GS register matches the configuration programmed in the respective ME_<mode>_MC register.

Note: It is recommended that software polls bit ME_GS[S_MTRANS] after requesting a transition to HALT0 or STOP0 modes.

Figure 1094. MC_ME mode diagram



52.4.2 Modes details

52.4.2.1 RESET mode

The chip enters this mode on the following events:

- from SAFE, DRUN, RUN0–3, or TEST mode when the field ME_MCTL[TARGET_MODE] is written with either ‘0000’ for a ‘functional’ reset or ‘1111’ for a ‘destructive’ reset
- from any mode due to a system reset by the MC_RGM because of some non-recoverable hardware failure in the system (see the MC_RGM chapter for details)

Transition to this mode is instantaneous, and the system remains in this mode until the reset sequence is finished. The mode configuration information for this mode is provided by the ME_RESET_MC register. This mode has a predefined configuration, and the 16 MHz Internal RC oscillator is selected as the system clock.

52.4.2.2 DRUN mode

The chip enters this mode on the following events:

- automatically from RESET mode after completion of the reset sequence
- from RUN0–3, SAFE, or TEST mode when the field ME_MCTL[TARGET_MODE] is written with ‘0011’

As soon as any of the above events has occurred, a DRUN mode transition request is generated. The mode configuration information for this mode is provided by the ME_DRUN_MC register. In this mode, the flash, all clock sources, and the system clock configuration can be controlled by software as required. After system reset, the software execution starts with the default configuration selecting the 16 MHz Internal RC oscillator as the system clock. The clock source configuration except system clock source 0 can be chosen by preloaded information in flash memory. This information is loaded into the ME_DRUN_MC register during PHASE3 of the reset sequence.

This mode is intended to be used by software

- to initialize all registers as per the system needs

Note: Software must ensure that the code executes from RAM before changing to this mode if the flash is configured to be in the low-power or power-down state in this mode.

52.4.2.3 SAFE mode

The chip enters this mode on the following events:

- from any mode except RESET when the field ME_MCTL[TARGET_MODE] is written with ‘0010’
- from any mode except RESET due to a SAFE mode request generated by the MC_RGM because of some potentially recoverable hardware failure in the system (see the MC_RGM chapter for details)

Note: If a hardware SAFE mode request occurs during RESET, depending on the timing of the SAFE mode request, SAFE mode may be entered immediately after the normal completion of the reset sequence or several system clock cycles after DRUN entry. The SAFE mode request does not have any influence on the execution of the reset sequence itself.

As soon as any of the above events has occurred, a SAFE mode transition request is generated. The mode configuration information for this mode is provided by the ME_SAFE_MC register. This mode has a pre-defined configuration, and the 16 MHz Internal RC oscillator is selected as the system clock.

If the SAFE mode is requested by software while some other mode transition process is ongoing, the new target mode becomes the SAFE mode regardless of other pending requests or new requests during the mode transition. No new mode request made during a transition to the SAFE mode will cause an invalid mode interrupt.

Note: If software requests to change to the SAFE mode and then requests to change back to the parent mode before the mode transition is completed, the chip’s final mode after mode transition will be the SAFE mode, and an invalid mode transition interrupt will be generated.

As long as a SAFE event is active, the system remains in the SAFE mode, and any software mode request during this time is ignored and lost.

This mode is intended to be used by software

- to assess the severity of the cause of failure and then to either
  - re-initialize the chip via the DRUN mode, or
  - completely reset the chip via the RESET mode.

If the outputs of the system I/Os need to be forced to a high impedance state upon entering this mode, the bit ME_SAFE_MC[PDO] should be set. The input levels remain unchanged.

52.4.2.4 TEST mode

The chip enters this mode on the following events:

- from the DRUN mode when the field ME_MCTL[TARGET_MODE] is written with '0001'

As soon as any of the above events has occurred, a TEST mode transition request is generated. The mode configuration information for this mode is provided by the ME_TEST_MC register. Except for the main voltage regulator, all resources of the system are configurable in this mode. The system clock to the whole system can be stopped by programming the SYSCLK bit field to '1111', and in this case, the only way to exit this mode is via a chip reset.

This mode is used by software to execute software test routines

Note: Software must ensure that the code executes from RAM before changing to this mode if the flash is configured to be in the low-power or power-down state in this mode.

52.4.2.5 RUN0–3 modes

The chip enters one of these modes on the following events:

- from the DRUN, SAFE, or another RUN0–3 mode when the field ME_MCTL[TARGET_MODE] is written with '0100–0111'
- from the HALT0 mode due to an interrupt event
- from the STOP0 mode due to an interrupt or wakeup event

As soon as any of the above events has occurred, a RUN0–3 mode transition request is generated. The mode configuration information for these modes is provided by the ME_RUN0–3_MC registers. In these modes, the flash, all clock sources, and the system clock configuration can be controlled by software as required.

These modes are intended to be used by software

- to execute application routines

Note: Software must ensure that the code executes from RAM before changing to this mode if the flash is configured to be in the low-power or power-down state in this mode.

52.4.2.6 HALT0 mode

The chip enters this mode on the following events:

- from one of the RUN0–3 modes when the field ME_MCTL[TARGET_MODE] is written with '1000'.

As soon as any of the above events has occurred, a HALT0 mode transition request is generated. The mode configuration information for this mode is provided by ME_HALT0_MC register. This mode is quite configurable, and the ME_HALT0_MC register should be programmed according to the system needs. The flash can be put in low-power or power-down mode as needed. If there is a HALT0 mode request while an interrupt request

is active, the transition to HALT0 is aborted with the resultant mode being the current mode, SAFE (on SAFE mode request), or DRUN (on reset), and an invalid mode interrupt is not generated.

This mode is intended as a first-level low-power mode with

- the core clocks frozen
- only a few peripherals running

and to be used by software

- to wait until it is required to do something and then to react quickly (that is, within a few system clock cycles of an interrupt event)

Note: It is good practice for software to ensure that bit ME_GS[S_MTRANS] has been cleared on HALT0 mode exit to ensure that the previous RUN0–3 mode configuration has been fully restored before executing critical code.

52.4.2.7 STOP0 mode

The chip enters this mode on the following events:

- from one of the RUN0–3 modes when the field ME_MCTL[TARGET_MODE] is written with ‘1010’.

As soon as any of the above events has occurred, a STOP0 mode transition request is generated. The mode configuration information for this mode is provided by the ME_STOP0_MC register. This mode is fully configurable, and the ME_STOP0_MC register should be programmed according to the system needs. The following clock sources are switched off in this mode:

- The Primary PLL
- The Secondary PLL

The flash can be put in power-down mode as needed. If there is a STOP0 mode request while any interrupt or wakeup event is active, the transition to STOP0 is aborted with the resultant mode being the current mode, SAFE (on SAFE mode request), or DRUN (on reset), and an invalid mode interrupt is not generated.

This can be used as an advanced low-power mode with the core clock frozen and almost all peripherals stopped.

This mode is intended as an advanced low-power mode with

- The core clock frozen
- Almost all peripherals stopped

and to be used by software

- To wait until it is required to do something with no need to react quickly (for example, allow for system clock source to be restarted)

This mode can be used to stop all clock sources and thus preserve the chip status. When exiting the STOP0 mode, the 16 MHz Internal RC oscillator clock is selected as the system clock until the target clock is available.

Note: It is good practice for software to ensure that bit ME_GS[S_MTRANS] has been cleared on STOP0 mode exit to ensure that the previous RUN0–3 mode configuration has been fully restored before executing critical code.

52.4.3 Mode Transition Process

The following subsections describe the mode transition process steps. The steps are followed in a predefined manner depending on the current chip mode and the requested target mode. In many cases of mode transition, not all steps need to be executed based on the mode control information, and some steps may not be applicable according to the mode definition itself.

52.4.3.1 Target Mode Request

The target mode is requested by accessing the ME_MCTL register with the required keys. This mode transition request by software must be a valid request satisfying a set of predefined rules to initiate the process. If the request fails to satisfy these rules, it is ignored, and the field ME_MCTL[TARGET_MODE] is not updated. An optional interrupt can be generated for invalid mode requests. Refer to *Section 52.4.5* for details.

In the case of mode transitions occurring because of hardware events such as a reset, a SAFE mode request, or interrupt requests and wakeup events to exit from low-power modes, the field ME_MCTL[TARGET_MODE] is automatically updated with the appropriate target mode. The mode change process start is indicated by the setting of the mode transition status bit ME_GS[S_MTRANS].

A RESET mode requested via the ME_MCTL register is passed to the MC_RGM, which generates a global system reset and initiates the reset sequence. The RESET mode request has the highest priority, and the MC_ME is kept in the RESET mode during the entire reset sequence.

The SAFE mode request has the next highest priority after reset. It can be generated either by software via the ME_MCTL register from all software running modes or by the MC_RGM after the detection of system hardware failures, which may occur in any mode.

52.4.3.2 Target Mode Configuration Loading

On completion of the *Section 52.4.3.1* step, the target mode configuration from the ME_<*target mode*>_MC register is loaded to start the resources (such as: voltage sources, clock sources, flash, pads) control process.

An overview of resource control possibilities for each mode is shown in *Table 1055*. A '√' indicates that a given resource is configurable for a given mode.

Table 1055. MC_ME resource control overview

Resource	Mode						
	RESET	TEST	SAFE	DRUN	RUN0–3	HALT0	STOP0
IRCOSC	On	√ On	On	On	On	On	On
XOSC	Off	√ Off	Off	√ Off	√ Off	√ Off	√ Off
PLL0	Off	√ Off	Off	√ Off	√ Off	√ Off	Off

Table 1055. MC_ME resource control overview (continued)

Resource	Mode						
	RESET	TEST	SAFE	DRUN	RUN0–3	HALT0	STOP0
PLL1	Off	√ Off	Off	√ Off	√ Off	√ Off	√ Off
FLASH	Normal	√ normal	Normal	√ Normal	√ Normal	√ Low-power	√ Power-down
MVREG	On	On	On	On	On	On	On
PDO	Off	√ Off	√ On	Off	Off	Off	√ Off

52.4.3.3 Peripheral Clocks Disable

On completion of the *Section 52.4.3.1* step and the system clock frequency ramp-down of the *Section 52.4.3.13* step, the MC_ME requests each peripheral to enter its stop mode when:

- The peripheral is configured to be disabled via the target mode, the peripheral configuration registers ME_RUN_PC0–7 and ME_LP_PC0–7, and the peripheral control registers ME_PCTL*n*

Note: The MC_ME automatically requests peripherals to enter their stop modes if the power domains in which they are residing are to be turned off due to a mode change. However, it is good practice for software to ensure that those peripherals that are to be powered down are configured in the MC_ME to be frozen.

Each peripheral acknowledges its stop mode request after closing its internal activity. The MC_ME then disables the corresponding clock(s) to this peripheral.

In the case of a *SAFE* mode transition request, the MC_ME does not wait for the peripherals to acknowledge the stop requests. The *SAFE* mode clock gating configuration is applied immediately regardless of the status of the peripherals' stop acknowledges.

Please refer to *Section 52.4.6* for more details.

Each peripheral that may block or disrupt a communication bus to which it is connected ensures that these outputs are forced to a safe or recessive state when the chip enters the SAFE mode.

52.4.3.4 Processor Low-Power Mode Entry

If, on completion of the *Section 52.4.3.3* step, the mode transition is to the HALT0, STOP0 mode, the MC_ME requests processor to enter its stopped state. processor acknowledges its stop state request bit after completing all outstanding bus transactions.

If, on completion of the *Section 52.4.3.3* step, the mode transition is from a non-low-power mode to another non-low-power mode, the MC_ME requests processor which is configured in the ME_CCTL registers to be disabled to enter its stopped state. The processor acknowledges its stop state request bit after completing all outstanding bus transactions.

52.4.3.5 Processor Clocks Disable

If, on completion of the *Section 52.4.3.4* step, and processor which are configured in the ME_CCTL register to be disabled are in their appropriate stopped state, the MC_ME disables the applicable processor clocks bits to achieve further power saving.

The clocks to processor which are configured to be running in both the current and target modes or disabled in both the current and target modes are unaffected while transitioning between non-low-power modes.

52.4.3.6 System Clock Disable

If, on completion of the *Section 52.4.3.5* step, all processor clocks have been disabled, the system clock used for non-processor system functions and memory is disabled.

52.4.3.7 Clock Sources Switch-On

On completion of the *Section 52.4.3.4* step, the MC_ME switches on all clock sources based on the <*clock source*>ON bits of the ME_<*current mode*>_MC and ME_<*target mode*>_MC registers. The following clock sources are switched on at this step:

- The 16 MHz Internal RC oscillator
- The External crystal oscillator
- The Primary PLL
- The Secondary PLL

The clock sources that are required by the target mode are switched on. The duration required for the output clocks to be stable depends on the type of source, and all further steps of mode transition depending on one or more of these clocks waits for the stable status of the respective clocks. The availability status of these clocks is updated in the S_<*clock source*> bits of ME_GS register.

The clock sources which need to be switched off are unaffected during this process in order to not disturb the system clock which might require one of these clocks before switching to a different target clock.

52.4.3.8 Flash Module Switch-On

On completion of the step, if the flash needs to be switched to normal mode from its low-power or power-down mode based on the FLAON bit field of the ME_<*current mode*>_MC and ME_<*target mode*>_MC registers, the MC_ME requests the flash to exit from its low-power/power-down mode. When the flash is available for access, the field ME_GS[S_FLA] is updated to ‘11’ by hardware.

Warning: It is illegal to switch the FLASH from low-power mode to power-down mode and from power-down mode to low-power mode. The MC_ME, however, does not prevent this nor does it flag it.

52.4.3.9 Pad Outputs-On

On completion of the step, if the PDO bit of the ME_<*target mode*>_MC register is cleared, then

- All pad outputs are enabled to return to their previous state
- The power sequence driver is switched on

52.4.3.10 Peripheral Clocks Enable

Based on the current and target chip modes, the peripheral configuration registers ME_RUN_PC0–7, ME_LP_PC0–7, and the peripheral control registers ME_PCTL*n*, the MC_ME enables the clocks for selected modules as required. This step is executed only after the process is completed, and if the value of the PWRLVL field of the ME_<*target mode*>_MC register is process is completed.

52.4.3.11 System and Processor Clocks Enable

On completion of the *Section 52.4.3.8* step, the MC_ME enables the non-processor system clock and the clocks of processor which are configured in the ME_CCTL register to be disabled in the current mode and to be running in the target mode.

The MC_ME initiates the resetting of processor which are configured in the ME_CCTL register to be running in the target mode and which have their RMC bits in the ME_CADDR register set to '1'. The core_rst_b outputs remain asserted until the end of the mode transition as indicated by the bit ME_GS[S_MTRANS] at which time they are deasserted.

If the value of the PWRLVL field of the ME_<*target mode*>_MC register is different from that of the ME_<*current mode*>_MC register's, the mode change handshake will not be initiated until after the system clock frequency ramp-down step of the *Section 52.4.3.13* process is completed.

52.4.3.12 Processor Low-Power Mode Exit

On completion of the *Section 52.4.3.11* step, the MC_ME requests processor which are configured in the ME_CCTL registers to be disabled in the current mode and to be running in the target mode.

52.4.3.13 System Clock Switching

Based on the SYSCLK bit field of the ME_<*current mode*>_MC and ME_<*target mode*>_MC registers, if the target and current system clock configurations differ, the following method is implemented for clock switching:

- The target clock configuration for the 16 MHz Internal RC oscillator takes effect only after the bit ME_GS[S_IRCOSC] is set by hardware (that is, the 16 MHz Internal RC oscillator has stabilized).
- The target clock configuration for the external crystal oscillator takes effect only after the bit ME_GS[S_XOSC] is set by hardware (that is, the External crystal oscillator has stabilized).
- The target clock configuration for the primary PLL (PHI) takes effect only after the bit ME_GS[S_PLL0] is set by hardware (that is, the Primary PLL has stabilized).
- The target clock configuration for the Secondary PLL takes effect only after the bit ME_GS[S_PLL1] is set by hardware (that is, the Secondary PLL has stabilized).
- If the clock is to be disabled, the SYSCLK bit field should be programmed with '1111'. This is possible only in the TEST mode.

If the value of the PWRLVL field of the ME_<*target mode*>_MC register not equal to that of the ME_<*current mode*>_MC register's, a progressive clock switching is performed.

The current system clock configuration can be observed by reading the field ME_GS[S_SYSCLK], which is updated after every system clock switching. Until the target clock is available, the system uses the previous clock configuration.

System clock switching starts only after

- the *Section 52.4.3.7* process has completed if the target system clock source is one of the following:
  - The 16 MHz Internal RC oscillator
  - The External crystal oscillator
  - The Primary PLL
- the *Section 52.4.3.3* process has completed in order to ensure that a system clock frequency change does not lead to unpredictable peripheral behavior before peripherals have been cleanly turned off

An overview of system clock source selection possibilities for each mode is shown in the following table. A shaded cell indicates the given clock source is not selectable for a given mode. A '√' indicates that a given clock source is selectable for a given mode.

Table 1056. MC_ME system clock selection overview

System clock source	Mode						
	RESET	TEST	SAFE	DRUN	RUN 0–3	HALT0	STOP0
16 MHz Internal RC oscillator	√ (Default)	√ (Default)	√ (Default)	√ (Default)	√ (Default)	√ (Default)	√ (Default)
external crystal oscillator		√		√	√	√	√

Table 1056. MC_ME system clock selection overview (continued)

System clock source	Mode						
	RESET	TEST	SAFE	DRUN	RUN 0–3	HALT0	STOP0
primary PLL (PHI)		√		√	√	√	√
Secondary PLL		√		√	√		
system clock is disabled		√(1)					

1. Disabling the system clock during ***TEST*** mode will require a reset in order to exit ***TEST*** mode.

52.4.3.14 Pad Switch-Off

If the PDO bit of the ME_<target mode>_MC register is ‘1’ then

- The outputs of the pads are forced to the high impedance state if the target mode is SAFE or TEST

This step is executed only after the *Section 52.4.3.3* process has completed.

52.4.3.15 Clock Sources Switch-Off

Based on the chip mode and the <*clock source*>ON bits of the ME_<*mode*>_MC registers, if a given clock source is to be switched off, the MC_ME requests the clock source to power down and updates its availability status bit S_<*clock source*> of the ME_GS register to ‘0’. The following clock sources are switched off at this step:

- The 16 MHz Internal RC oscillator
- The External crystal oscillator
- The Primary PLL
- The Secondary PLL

For the clock sources related to the system clock, this step is executed only after the *Section 52.4.3.13* process has completed.

52.4.3.16 Flash Switch-Off

Based on the FLAON bit field of the ME_<*current mode*>_MC and ME_<*target mode*>_MC registers, if the flash is to be put in its low-power or power-down mode, the MC_ME requests the flash to enter the corresponding power mode and waits for the flash to acknowledge. The exact power mode status of the flash is updated in the field ME_GS[S_FLA]. This step is executed only when the *Section 52.4.3.5* and *Section 52.4.3.6* processes have completed.

52.4.3.17 Current Mode Update

The current mode status field ME_GS[S_CURRENT_MODE] is updated with the target mode field ME_MCTL[TARGET_MODE] when:

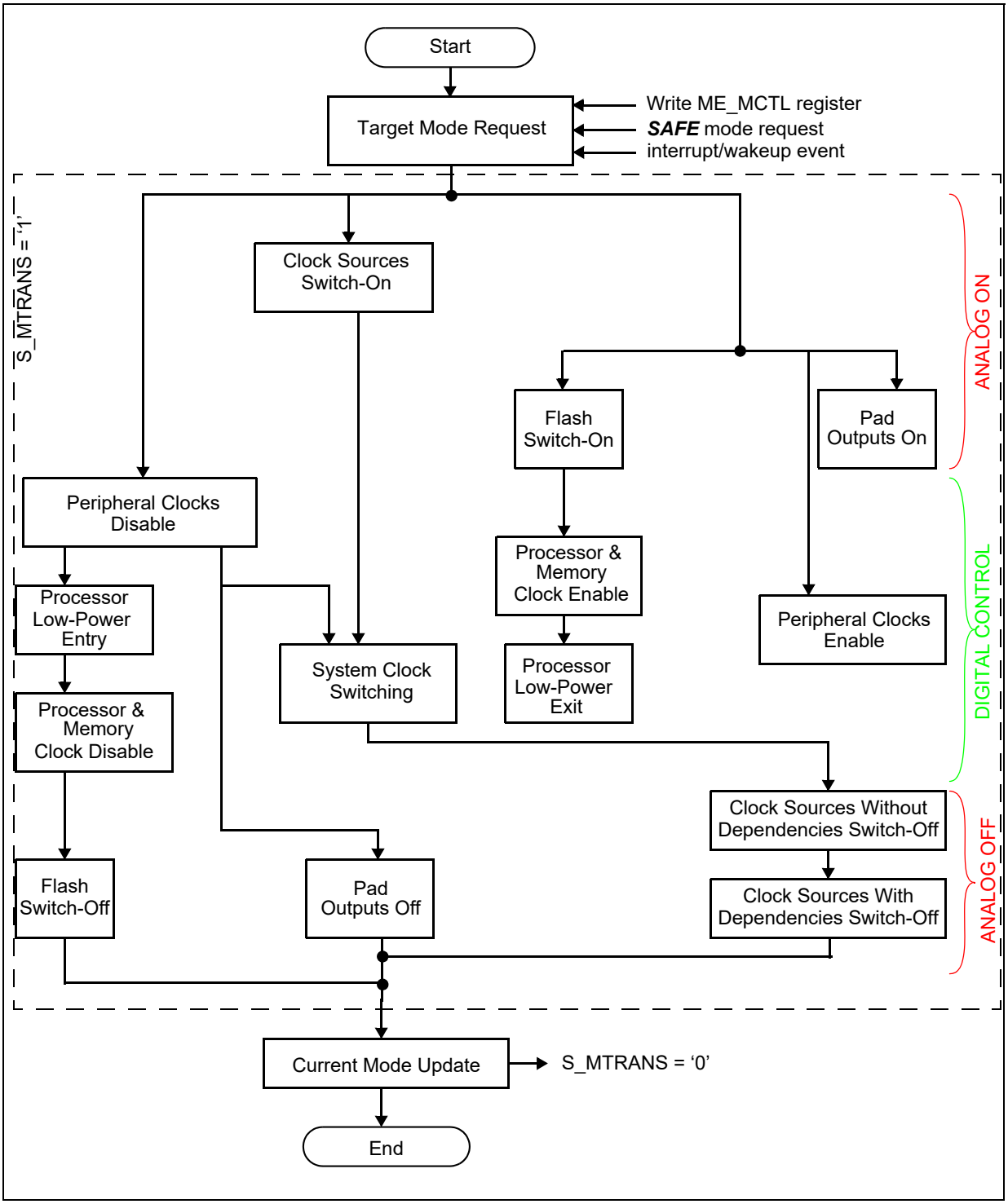
- All the updated status bits in the ME_GS register match the configuration specified in the ME_<*target mode*>_MC register
- Power sequences are done
- Clock disable/enable process is finished
- Processor low-power mode (stop) entry and exit processes are finished

Note: SAFE mode entry does not wait for the clock disable/enable process to finish. It only waits for the ME_GS[S_RC] bit to be set. This is to ensure that the SAFE mode is entered as quickly as possible.

Software can monitor the mode transition status by reading the bit ME_GS[S_MTRANS]. The mode transition latency can differ from one mode to another depending on the resources' availability before the new mode request and the target mode's requirements.

If a mode transition is taking longer to complete than is expected, the ME_DMTS register can indicate which process is still in progress.

Figure 1095. MC_ME transition diagram



52.4.4 Protection of mode configuration registers

While programming the mode configuration registers ME_<*mode*>_MC, the following rules must be respected. Otherwise, the write operation is ignored and an invalid mode configuration interrupt may be generated.

- If the 16 MHz Internal RC oscillator is selected as the system clock, IRCOSC must be on.
- If the external crystal oscillator clock is selected as the system clock, XOSC must be on.
- If the primary PLL (PHI) clock is selected as the system clock, PLL0 must be on.
- If the Secondary PLL clock is selected as the system clock, PLL1 must be on.

Note: Software must ensure that clock sources with dependencies other than those mentioned above are switched on as needed. There is no automatic protection mechanism to check this in the MC_ME.

- Configuration '00' for the FLAON bit field is reserved.
- System clock configurations marked as 'reserved' may not be selected.
- Configuration '1111' for the SYSCLK bit field is allowed only for the TEST mode, and only in this case may all system clock sources be turned off.

Warning: If the system clock is stopped during TEST mode, the chip can exit only via a system reset.

52.4.5 Mode Transition Interrupts

The MC_ME provides interrupts for incorrectly configuring a mode, requesting an invalid mode transition, indicating a SAFE mode transition not due to a software request, and indicating when a mode transition has completed.

52.4.5.1 Invalid Mode Configuration Interrupt

Whenever a write operation is attempted to the ME_<*mode*>_MC registers violating the protection rules mentioned in the *Section 52.4.4*, the interrupt pending bit ME_IS[I_ICONF] is set and an interrupt request is generated if the mask bit ME_IM[M_ICONF] is '1'.

In addition, during a mode transition, if a clock source has been configured in the ME_<*target mode*>_MC register to be off and a peripheral requiring this clock source to be on has been enabled via the ME_RUN_PC0–7/ME_LP_PC0–7 and ME_PCTL*n* registers, the interrupt pending bit ME_IS[I_ICONF_CU] is set and an interrupt request is generated if the mask bit ME_IM[M_ICONF_CU] is '1'.

If an attempt to write to one of the ME_CCTL0 or ME_CADDR0 registers is made after a mode change request has been made via the second write to the ME_CTL register and before the completion of that mode transition as indicated by the bit ME_GS[S_MTRANS] deasserting, the interrupt pending bit ME_IS[I_ICONF_CC] is set and an interrupt request is generated if the mask bit ME_IM[M_ICONF_CC] is '1'.

52.4.5.2 Invalid Mode Transition Interrupt

The mode transition request is considered invalid under the following conditions:

- If the system is in the SAFE mode and the SAFE mode request from MC_RGM is active, and if the target mode requested is other than RESET or SAFE, then this new mode request is considered to be invalid, and the bit ME_IMTS[S_SEA] is set.
- If the field ME_MCTL[TARGET_MODE] is written with a value different from the specified mode values (that is, a non-existing mode), an invalid mode transition event is generated. When such a non-existing mode is requested, the bit ME_IMTS[S_NMA] is set. This condition is detected regardless of whether the proper key mechanism is followed while writing the ME_MCTL register.
- If some of the chip modes are disabled as programmed in the ME_ME register, their respective configurations are considered reserved, and any access to the ME_MCTL register with those values results in an invalid mode transition request. When such a disabled mode is requested, the bit ME_IMTS[S_DMA] is set. This condition is detected regardless of whether the proper key mechanism is followed while writing the ME_MCTL register.
- If the target mode is not a valid mode with respect to the current mode, the mode request illegal status bit ME_IMTS[S_MRI] is set. This condition is detected only when the proper key mechanism is followed while writing the ME_MCTL register. Otherwise, the write operation is ignored.
- If further new mode requests occur while a mode transition is in progress (the bit ME_GS[S_MTRANS] is ‘1’), the mode transition illegal status bit ME_IMTS[S_MTI] is set. This condition is detected only when the proper key mechanism is followed while writing the ME_MCTL register. Otherwise, the write operation is ignored.

Note: As the causes of invalid mode transitions may overlap at the same time, the priority implemented for invalid mode transition status bits of the ME_IMTS register in the order from highest to lowest is S_SEA, S_NMA, S_DMA, S_MRI, and S_MTI.

As an exception, the mode transition request is not considered as invalid under the following conditions:

- A new request is allowed to enter the RESET or SAFE mode irrespective of the mode transition status.
- As the exit of HALT0 and STOP0 modes depends on the interrupts of the system which can occur at any instant, these requests to return to RUN0–3 modes are always valid.
- In order to avoid any unwanted lockup of the chip modes, software can abort a mode transition by requesting the parent mode if, for example, the mode transition has not completed after a software determined ‘reasonable’ amount of time for whatever reason. The parent mode is the chip mode before a valid mode request was made.
- Self-transition requests (for example, RUN0 to RUN0) are not considered as invalid even when the mode transition process is active (that is, S_MTRANS is ‘1’). During the low-power mode exit process, if the system is not able to enter the respective RUN0–3 mode properly (that is, all status bits of the ME_GS register match with configuration bits in the ME_<mode>_MC register), then software can only request the SAFE or RESET mode. It is not possible to request any other mode or to go back to the low-power mode again.

Whenever an invalid mode request is detected, the interrupt pending bit ME_IS[I_IMODE] is set, and an interrupt request is generated if the mask bit ME_IM[M_IMODE] is ‘1’.

52.4.5.3 SAFE Mode Transition Interrupt

Whenever the system enters the SAFE mode as a result of a SAFE mode request from the MC_RGM due to a hardware failure, the interrupt pending bit ME_IS[I_SAFE] is set, and an interrupt is generated if the mask bit ME_IM[M_SAFE] is ‘1’.

The SAFE mode interrupt pending bit can be cleared only when the SAFE mode request is deasserted by the MC_RGM (see the MC_RGM chapter for details on how to clear a SAFE mode request). If the system is already in SAFE mode, any new SAFE mode request by the MC_RGM also sets the interrupt pending bit I_SAFE. However, the SAFE mode interrupt pending bit is not set when the SAFE mode is entered by a software request (that is, programming of ME_MCTL register).

52.4.5.4 Mode Transition Complete interrupt

Whenever the system fully completes a mode transition (that is, the bit ME_GS[S_MTRANS] transits from ‘1’ to ‘0’), the interrupt pending bit ME_IS[I_MTC] is set, and an interrupt request is generated if the mask bit ME_IM[M_MTC] is ‘1’. The interrupt bit I_MTC is not set when entering low-power modes HALT0 and STOP0 in order to avoid the same event requesting the immediate exit of these low-power modes.

52.4.6 Peripheral Clock Gating

During all chip modes, each peripheral can be associated with a particular clock gating policy determined by two groups of peripheral configuration registers.

The run peripheral configuration registers ME_RUN_PC0–7 are chosen only during the software running modes DRUN, TEST, SAFE, and RUN0–3. All configurations are programmable by software according to the needs of the application. Each configuration register contains a mode bit which determines whether or not a peripheral clock is to be gated. Run configuration selection for each peripheral is done by the RUN_CFG bit field of the ME_PCTL*n* registers.

The low-power peripheral configuration registers ME_LP_PC0–7 are chosen only during the low-power modes HALT0 and STOP0. All configurations are programmable by software according to the needs of the application. Each configuration register contains a mode bit which determines whether or not a peripheral clock is to be gated. Low-power configuration selection for each peripheral is done by the LP_CFG bit field of the ME_PCTL*n* registers.

Any modifications to the ME_RUN_PC0–7, ME_LP_PC0–7, and ME_PCTL*n* registers do not affect the clock gating behavior until a new mode transition request is generated.

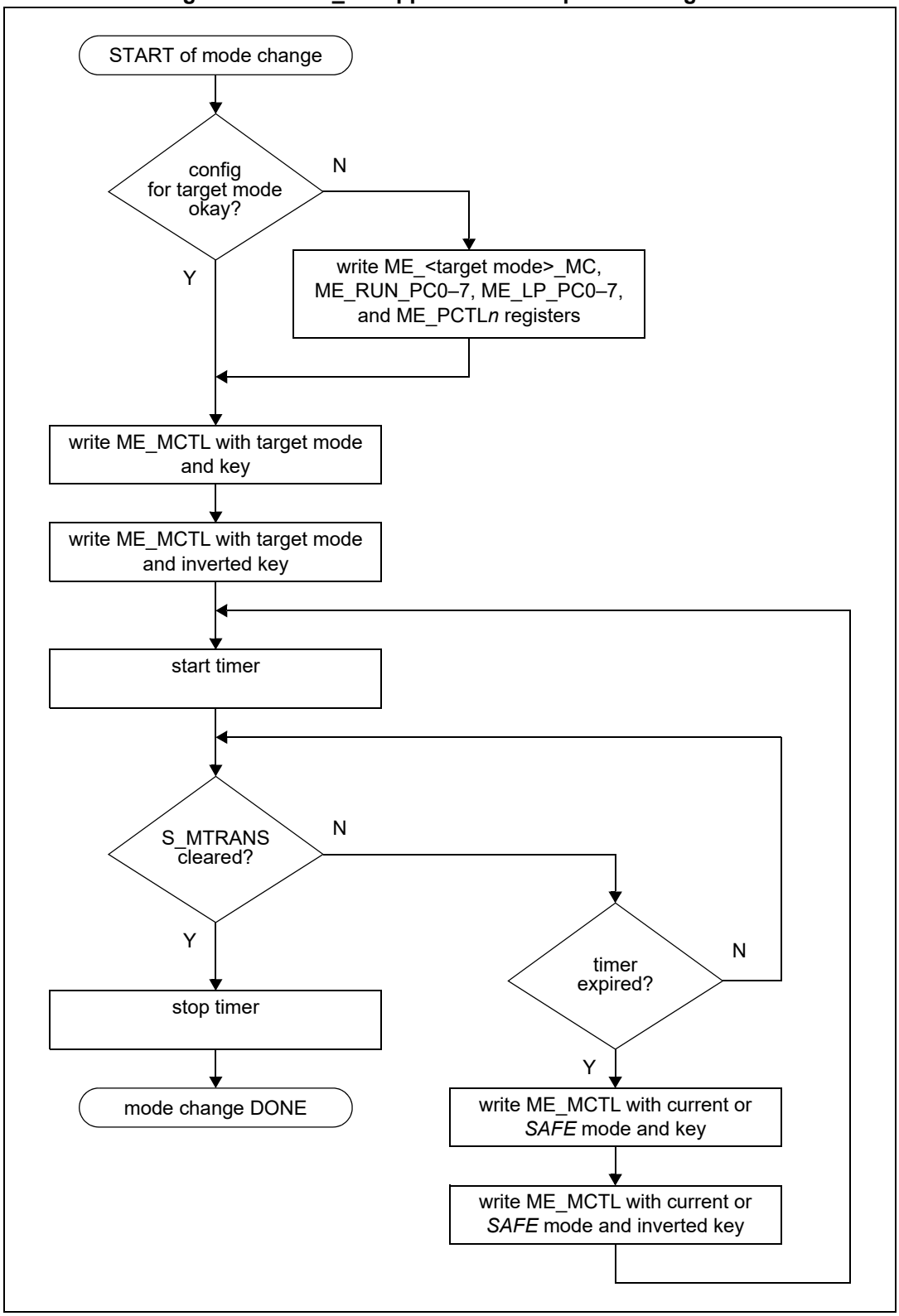
Whenever the chip enters a debug session during any mode, the following occurs for each peripheral:

- The clock is gated if the DBG_F bit of the associated ME_PCTL*n* register is set. Otherwise, the peripheral clock gating status depends on the RUN_CFG and LP_CFG bits.

52.4.7 Application example

Figure 1096 shows an example application flow for requesting a mode change and then waiting until the mode transition has completed.

Figure 1096. MC_ME application example flow diagram



53 e200z420n3 Core Debug Support

This chapter describes the debug features of the e200z420n3 core.

53.1 Overview

Internal debug support in the e200z420n3 core allows for software and hardware debug by providing debug functions, such as instruction and data breakpoints and program trace modes. For software based debugging, debug facilities consisting of a set of software accessible debug registers and interrupt mechanisms are provided. These facilities are also available to a hardware based debugger which communicates using a modified IEEE 1149.1 Test Access Port (TAP) controller and pin interface. When hardware debug is enabled, the debug facilities controlled by hardware are protected from software modification.

Software debug facilities are defined as part of <asisitalic>PowerISA 2.06. e200z420n3 supports a subset of these defined facilities. In addition to the facilities defined in <asisitalic>PowerISA 2.06, e200z420n3 provides additional flexibility and functionality in the form of additional instruction breakpoints, linked instruction and data breakpoints, and extended range and value masking. These features are also available to a hardware-based debugger.

When not being used for debugging purposes, a portion of the debug facilities may be configured to provide a stack limit checking mechanism.

53.1.1 Software Debug Facilities

e200z420n3 provides debug facilities to enable hardware and software debug functions, such as instruction and data breakpoints and program single stepping. The debug facilities consist of a set of debug control registers (DBCR0–2,4–8, EDBRAC0), a set of address compare registers (IAC1–8, DAC1–4), a set of 64-bit data value compare registers (DVC1, DVC2), a Debug Status Register (DBSR) for enabling and recording various kinds of debug events, a Debug Data Effective Address Register (DDEAR), and a special Debug interrupt type built into the interrupt mechanism.

The debug facilities also provide a mechanism for software-controlled processor reset, and debug mode entry. In addition, the Performance Monitor may be configured to utilize the debug interrupt type. Also, the Memory Protection Unit (MPU) may be configured to generate debug events using region descriptors which are not utilized for performing a protection function.

Software debug facilities are enabled by setting the internal debug mode bit in Debug Control register 0 ($DBCR0_{IDM}$). When internal debug mode is enabled, debug events can occur, and can be enabled to record exceptions in the Debug Status register (DBSR). If enabled by MSR_{DE}, these recorded exceptions cause Debug interrupts to occur. When $DBCR0_{IDM}$ is cleared (and $EDBCR0_{EDM}$ is cleared as well), no debug events occur, and no status flags are set in DBSR unless already set. In addition, when $DBCR0_{IDM}$ is cleared (or is overridden by $EDBCR0_{EDM}$ being set and EDBRAC0 indicating no resource is “owned” by software) no Debug interrupts will occur, regardless of the contents of DBSR. A software Debug interrupt handler may access all system resources and perform necessary functions appropriate for system debug.

53.1.1.1 <asisitalic>PowerISA 2.06 Compatibility

The e200z420n3 core implements a subset of the <asisitalic>PowerISA 2.06 internal debug features. The following restrictions on functionality are present:

- Instruction address compares do not support compare on physical (real) addresses.
- Data address compares do not support compare on physical (real) addresses.

53.1.2 Additional Debug Facilities

In addition to the debug functionality defined in <asisitalic>PowerISA 2.06, e200z420n3 provides capability to link instruction and data breakpoints, provides additional instruction and data breakpoints, extended range and value masking, multiple watchpoint outputs, provides capability for the Performance Monitor to generate debug events, and allows for sharing of debug resources between software and a hardware debugger. (See *Section 53.1.4*.) A data trace port is also provided.

53.1.2.1 Data Trace Port

The data trace port interface is provided to assist in implementing extended debug watchpoint/breakpoint/trace capture capability with logic external to the e200z420n3 core. This port provides information corresponding to each read or write access completed without error by the CPU. In order to report data accesses, the Nexus 3 DTC register DI control bit must be set to trace data accesses, not instruction accesses (that is the normal setting), see the “Data Trace Control Register (DTC)” section in the Core (e200z420n3) Nexus 3 Module chapter). The data trace port will report aligned accesses and misaligned accesses as one access, unless the two portions of a misaligned access are non-contiguous, such as when the second portion of the misaligned access is to address 0.

53.1.3 Hardware Debug Facilities

The e200z420n3 core contains facilities that allow for external test and debugging. A modified IEEE 1149.1 control interface is used to communicate with the core resources. This interface is implemented through a standard 1149.1 TAP (test access port) controller.

By using public instructions, the external debugger can freeze or halt the e200z420n3 core, read and write internal state and debug facilities, single-step instructions, and resume normal execution.

Hardware Debug is enabled by setting the External Debug Mode enable bit in External Debug Control register 0 ($EDBCR0_{EDM}$), which is also aliased to $DBCR0_{EDM}$. Setting $EDBCR0_{EDM}$ overrides the Internal Debug Mode enable bit $DBCR0_{IDM}$ unless resources are provided back to software via the settings in EDBRAC0. When the Hardware Debug facility is enabled, software is blocked from modifying the “hardware-owned” debug facilities. In addition, since resources are “owned” by the hardware debugger, inconsistent values may be present if software attempts to read “hardware-owned” debug-related resources.

When hardware debug is enabled by setting $EDBCR0_{EDM}$ = 1, the control registers and resources described in *Section 53.3* are reserved for use by the external debugger. The same events described in *Section 53.2* are also used for external debugging, but exceptions are not generated to running software. Hardware-owned debug events enabled in the respective DBCR0–8 registers are recorded in the EDBSR0 register (not the DBSR) regardless of MSR_{DE}, and no debug interrupts are generated unless a) the resource is granted back to software via EDBRAC0 settings, and b) debug mode entry is not masked by

the corresponding event bit in EDBSRMSK0. Instead, the CPU will enter debug mode when an enabled event causes a EDBSR0 bit to become set. $EDBCR0_{EDM}$, EDBSR0, EDBSRMSK0, EDDEAR, and EDBRAC0 may only be written through the OnCE port.

Access to most debug resources (registers) requires that the CPU clock (**m_clk**) be running in order to perform write accesses from the external hardware debugger.

53.1.4 Sharing Debug Resources by Software/Hardware

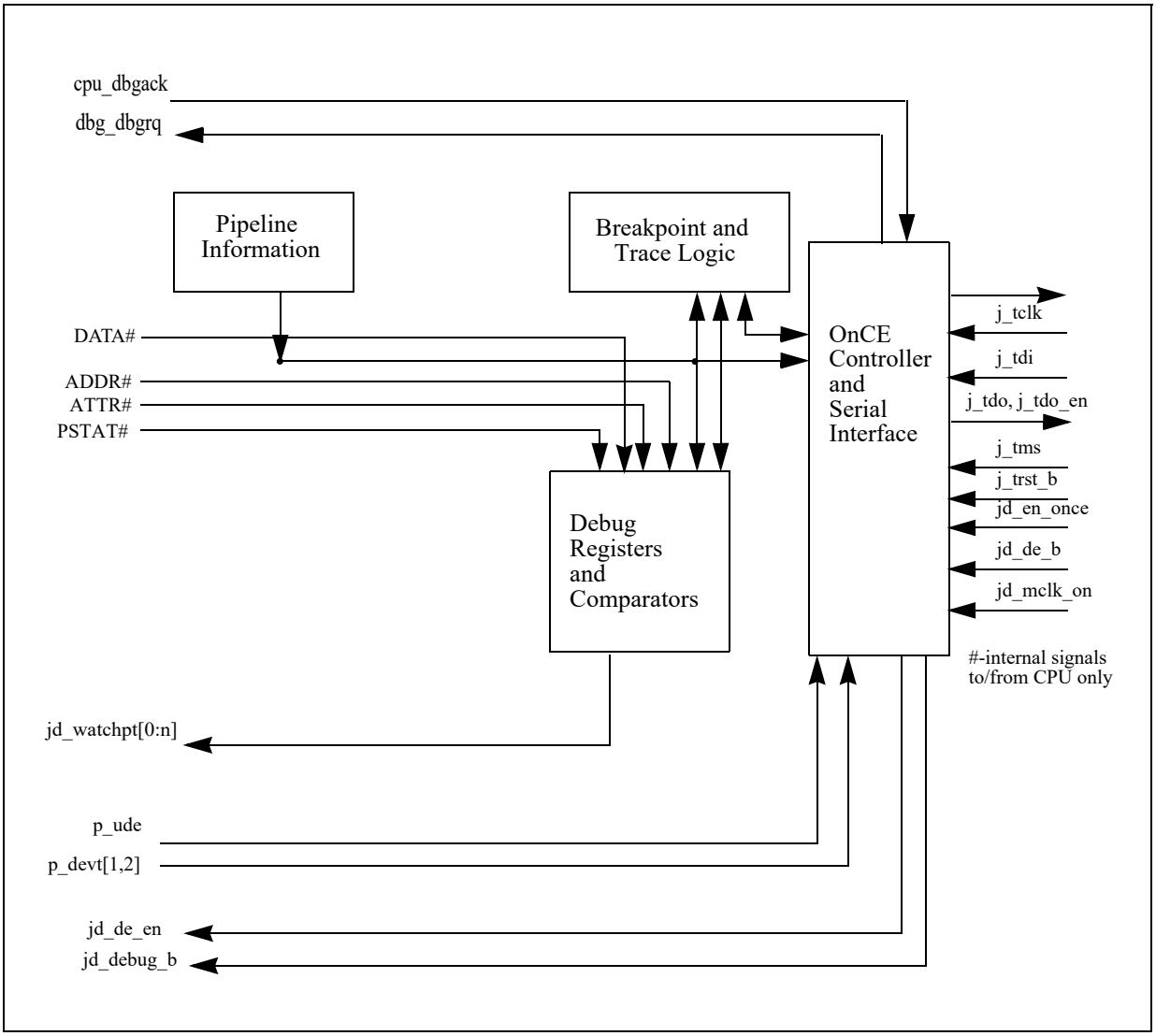
Debug resources may be shared by a hardware debugger and software debug based on the settings of debug control register EDBRAC0. When $EDBCR0_{EDM}$ is set, EDBRAC0 settings determine which debug resources are allocated to software and which resources remain under exclusive hardware control. Software-owned resources which set DBSR bits when $DBCR0_{IDM}$ = 1 will cause a debug interrupt to occur when enabled with MSR_{DE}. Hardware-owned resources which set EDBSR0 bits when $EDBCR0_{EDM}$ = 1 will cause an entry into debug mode if the event is not masked in EDBSRMSK0. EDBRAC0 is read-only by software. When resource sharing is enabled, ($EDBCR0_{EDM}$ = 1 and $EDBRAC0_{IDM}$ = 1), only software-owned resources may be modified by software. Hardware always has full access to all registers and all register fields through the OnCE register access mechanism, and it is up to the debug firmware to properly implement modifications to these registers with read-modify-write operations to implement any control sharing with software. Hardware-owned resources will set status bits in the EDBSR0 register instead of in DBSR, and will report the effective address of data breakpoints in EDDEAR instead of DDEAR. Settings in EDBRAC0 should be considered by the debug firmware in order to preserve software settings of control and status registers as appropriate when hardware modifications to the debug registers are performed.

53.1.4.1 Simultaneous Hardware and Software Debug Event Handing

Since it is possible that a “hardware-owned” resource can produce a debug event in conjunction with a software-owned resource producing a different debug event simultaneously, a priority ordering mechanism is implemented which guarantees that the hardware event is handled as soon as possible, while preserving the recognition of the software event. The CPU will give highest priority to the software event initially in order to reach a recoverable boundary, and then will give highest priority to the hardware event in order to enter debug mode as near the point of event occurrence as possible. This is implemented by allowing software exception handing to begin internal to the CPU and to reach the point where the current program counter and MSR values have been saved into DSRR0/1, and the new PC pointing to the debug interrupt handler, along with the new MSR updates. At this point, hardware priority takes over, and the CPU enters debug mode.

Figure 1097 shows the e200z420n3 debug resources.

Figure 1097. e200z420n3 Debug Resources



53.2 Software Debug Events and Exceptions

Software debug events and exceptions are available when internal debug mode is enabled ($DBCR0_{IDM}$ = 1) and not overridden by external debug mode ($EDBCR0_{EDM}$ must either be cleared or corresponding resources must be allocated to software debug by the settings in EDBRAC0). When enabled, debug events cause debug exceptions to be recorded in the Debug Status Register. Specific event types are enabled by the Debug Control Registers (DBCR0–8). The Unconditional Debug Event (UDE) is an exception to this rule; it is always enabled. Once a Debug Status Register (DBSR) bit is set by a debug resource which is “owned” by software (other than MRR, DAC_OFST, or VLES), if Debug interrupts are enabled by MSR_{DE}, a Debug interrupt will be generated. The debug interrupt handler is responsible for ensuring that multiple repeated debug interrupts do not occur by clearing the DBSR as appropriate.

Certain debug events are not allowed to occur when MSR_{DE} = 0 and $DBCR0_{IDM}$ = 1. In such situations, no debug exception occurs and thus no DBSR bit is set. Other debug events may cause debug exceptions and set DBSR bits regardless of the state of MSR_{DE}. A Debug interrupt will be delayed until MSR_{DE} is later set to '1'.

When a Debug Status Register bit is set while MSR_{DE} = 0, an Imprecise Debug Event flag ($DBSR_{IDE}$) will also be set to indicate that an exception bit in the Debug Status Register was set while Debug interrupts were disabled. Debug interrupt handler software can use this bit to determine whether the address recorded in Debug Save/Restore Register 0 is an address associated with the instruction causing the debug exception, or the address of the instruction which enabled a delayed Debug interrupt by setting the MSR_{DE} bit. A **mtmsr** or **mtdbcr0** which causes both MSR_{DE} and $DBCR0_{IDM}$ to become set, enabling precise debug mode, may cause an Imprecise (Delayed) Debug exception to be generated due to an earlier recorded event in the Debug Status register.

There are eight types of debug events defined by <asisitalic>PowerISA 2.06:

1. Instruction Address Compare debug events
2. Data Address Compare debug events
3. Trap debug events
4. Branch Taken debug events
5. Instruction Complete debug events
6. Interrupt/Critical Interrupt Taken debug events
7. Return/Critical Return debug events
8. Unconditional debug events

In addition, e200z420n3 defines additional debug events:

- The External debug events DEVT1 and DEVT2 which are described in *Section 53.2.11*.
- The Performance Monitor Interrupt event PMI which is described in *Section 53.2.13*.

The e200z420n3 debug configuration supports most of these event types. Unsupported <asisitalic>PowerISA 2.06 defined functionality is as follows:

- Instruction Address Compare and Data Address Compare *Real address* mode is not supported

A brief description of each of the event types follows.

53.2.1 Instruction Address Compare Event

Instruction Address Compare debug events occur when enabled and execution is attempted of an instruction at an address that meets the criteria specified in the DBCR0, DBCR1, DBCR5, DBCR6, and IAC1–8 Registers. Instruction Address compares may specify user/supervisor mode, along with an effective address, masked effective address, or range of effective addresses for comparison. This event can occur and be recorded in DBSR regardless of the setting of MSR_{DE}. IAC events will not occur when an instruction would not have normally begun execution due to a higher priority exception at an instruction boundary.

IAC compares perform a 31-bit compare for VLE instruction storage accesses. Each halfword fetched by the instruction fetch unit will be marked with a set of bits indicating whether an Instruction Address Compare occurred on that halfword. Debug exceptions will occur if enabled and a 16-bit instruction, or the first halfword of a 32-bit instruction, is tagged with an IAC hit.

53.2.2 Data Address Compare Event

Data Address Compare debug events occur when enabled and execution of a load or store class instruction or a cache maintenance instruction results in a data access that meets the criteria specified in the DBCR0, DBCR2, DBCR4, DBCR7–8, DAC1–4, DVC1, and DVC2 Registers. Data address compares may specify user/supervisor mode, along with an effective address, masked effective address, or range of effective addresses for comparison. This event can occur and be recorded in DBSR regardless of the setting of MSR_{DE}. Four address compare values (DAC1–4) are provided.

The effective address of the load, store, or cache control operation is recorded in the DDEAR register when a data address compare event is recorded in DBSR if the previous values of the $DBSR_{DAC\{R,W\}}$ bits are zero. Once a DDEAR update is performed, these bits must be cleared by software prior to another DDEAR hardware update occurring. A subsequent DAC event will not update the DDEAR register if either of the $DBSR_{DAC\{R,W\}}$ bits are set.

Note: In contrast to the <asisitalic>PowerISA 2.06 definition, Data Address Compare events on e200z420n3 do not prevent the load or store class instruction from completing. If a load or store class instruction completes successfully without a Data Storage interrupt, Data Address Compare exceptions are reported at the completion of the instruction. If the exception results in a precise Debug interrupt, the address value saved in DSRR0 is the address of the instruction following the load or store class instruction. For DVC DAC events, the exception can be imprecisely reported even further past the load or store class instruction generating the event (without necessarily affecting $DBSR_{IDE}$) and the saved address value can point to a subsequent instruction past the next instruction. This occurrence is indicated in the $DBSR_{DAC_OFST}$ field.

If a load or store class instruction does not complete successfully due to a Data Storage exception or a machine check condition for the load or store, and a Data Address Compare debug exception also occurs, the result is an imprecise Debug interrupt, the address value saved in DSRR0 is the address of the load or store class instruction, and the $DBSR_{IDE}$ bit will be set. In addition to occurring when $DBCR0_{IDM}$ = 1, this circumstance can also occur when $EDBCR0_{EDM}$ = 1 and the event is hardware-owned, in which case $EDBSR0_{IDE}$ will be set.

DAC events will not be recorded if a load multiple word or store multiple word instruction is interrupted prior to completion by a critical input or external input interrupt.

DAC events will occur for cache control instructions without performing data compares, regardless of the settings of the DBCRx registers and DVC registers for data value comparison qualifications, that is no data comparisons are performed since these instructions do not have associated data values.

DAC events are not signaled on the second portion of a misaligned load or store that is broken up into two separate accesses.

*DAC events are not signaled on the **mpure** or **mpuwe** MPU instructions.*

*DAC[1,2] events are not signaled if DVC[1,2]M is non-zero and a DSI exception occurs on the load or store, since the load or store access is not performed. For a **lmw** or **stmw** transfer however, if a DVC successfully occurs on a transfer and a later transfer encounters a DSI exception, the DAC event will be reported, since a successful data value compare took place.*

*Due to internal pipeline status, for a reported DAC event on a **lmw** or **stmw** transfer, the value stored in the DDEAR/EDDEAR will be the effective address of the first transfer of the sequence, and not necessarily the precise transfer which caused the DAC event.*

53.2.2.1 Data Address Compare Event Status Updates

Data Address Compare debug events with Data Value compares can be reported ambiguously in several circumstances involving issuing a sequence of load or store class instructions. Due to the CPU pipeline and the delay in performing the data value compare following completion of the access, if the first load or store class instruction generates a DVC DAC, a second and possibly third load or store class instruction may also generate a DAC or DVC DAC event, or may generate a DSI exception with or without a simultaneous DAC event.

Also, since non-load/store instructions may be dual-issued in combination with a load/store instruction, the actual number of additional instructions which are completed following a recognized DVC DAC on a load/store instruction may vary from 0 to 5. This value will be reported in the $DBSR_{DAC_OFST}$ field when the DVC DAC status is recorded.

Table 1057 outlines the settings of the DBSR, DSRR0 saved value, and potential updating of the ESR register for various exception cases on sequences of load/store class instructions. Not all exception combinations are covered in the table, such as IAC, ISI, or Alignment exceptions on subsequent instructions. In general these exceptions will cause further instruction issue to be halted, execution of the excepting instruction to be aborted, and reporting of these exceptions will be masked. The saved DSRR0 value will point to this excepting instruction, and the exception(s) may be regenerated after returning from the debug interrupt handler and attempting to re-execute the instruction pointed to by DSRR0. In addition, in the examples in *Table 1057*, the DAC_OFST and DSRR0 values assume no dual issue occurs. If dual-issue occurs with the first, second, or third column, then the DAC_OFST and DSRR0 values will point beyond the values shown.

Table 1057. DAC events and Resultant Updates

1st load/store class instruction	2nd instruction (load/store class unless otherwise specified)	3rd instruction (load/store class unless otherwise specified)	Result
DSI, no DAC	—	—	Take DSI exception, no DBSR update. Update ESR.
DSI, with DACx	—	—	Take Debug exception, DBSR update setting DACx and IDE, DAC_OFST not set. DSRR0 points to 1st load/store class instruction. No ESR update.
DACx	—	—	Take Debug exception, DBSR update setting DACx, DAC_OFST not set. DSRR0 points to 2nd load/store class instruction. No ESR update.
DVC DACx	No exceptions, any instruction	No exceptions, Non-ldst instruction	Take Debug exception, DBSR update setting DACx, DAC_OFST set to 3’b001. DSRR0 points to 3rd instruction. No ESR update.
DVC DACx	No exceptions	No exceptions, Ldst instruction	Take Debug exception, DBSR update setting DACx, DAC_OFST set to 3’b010. DSRR0 points to instruction after 3rd instruction. No ESR update.

Table 1057. DAC events and Resultant Updates (continued)

1st load/store class instruction	2nd instruction (load/store class unless otherwise specified)	3rd instruction (load/store class unless otherwise specified)	Result
DVC DACx	DSI, no DAC	—	Take Debug exception, DBSR update setting DACx, DAC_OFST not set. DSRR0 points to 2nd load/store class instruction. No ESR update. **Note:** In this case the 2nd ld/st exception is masked. This behavior is implementation dependent and may differ on other CPUs.
DVC DACx	DSI, with DACy	—	Take Debug exception, DBSR update setting DACx. DAC_OFST not set. DSRR0 points to 2nd load/store class instruction. No ESR update. **Note:** In this case the 2nd ld/st exception is masked. This behavior is implementation dependent and may differ on other CPUs.
DVC DACx	DACy	—	Take Debug exception, DBSR update setting DACx, DACy. DAC_OFST set to 3'b001. DSRR0 points to 3rd instruction. **Note:** In this case if x==y, then the resultant state of DBSR and DSRR0 may be indistinguishable from the "no DACy" case.
DVC DACx	DVC DACy, Normal Ldst	Non-Ldst instruction	Take Debug exception, DBSR update setting DACx, DACy. DAC_OFST set to 3'b001. DSRR0 points to the 3rd instruction. **Note:** In this case if x==y, then the resultant state of DBSR and DSRR0 may be indistinguishable from the "no DACy" case.
DVC DACx	DVC DACy, Normal Ldst	Ldst instruction, no exception	Take Debug exception, DBSR update setting DACx, DACy. DAC_OFST set to 3'b010. DSRR0 points to instruction after the 3rd load/store class instruction. **Note:** In this case if x==y, then the resultant state of DBSR and DSRR0 may be indistinguishable from the "no DACy" case.
DVC DACx	DVC DACy, Normal Ldst	DSI Error, with or without DAC	Take Debug exception, DBSR update setting DACx, DACy. DAC_OFST set to 3'b001. No ESR update. DSRR0 points to 3rd instruction. **Note:** In this case if x==y, then the resultant state of DBSR and DSRR0 may be indistinguishable from the "no DACy" case. In this case the 3rd ld/st exception is masked. This behavior is implementation dependent and may differ on other CPUs.

Table 1057. DAC events and Resultant Updates (continued)

1st load/store class instruction	2nd instruction (load/store class unless otherwise specified)	3rd instruction (load/store class unless otherwise specified)	Result
DVC DACx	DVC DACy, Normal Ldst	DACy, or DVC DAC_y Normal Ldst or multiple word Ldst	Take Debug exception, DBSR update setting DACx, DACy. DAC_OFST set to 3'b010. DSRR0 points to instruction after the 3rd load/store class instruction. **Note:** In this case if x==y, then the resultant state of DBSR and DSRR0 may be indistinguishable from the "no DACy" case.
DVC DACx	DVC DACy, Ldst multiple (lmw, stmw)	Any instruction including ld/st	Take Debug exception, DBSR update setting DACx, DACy. DAC_OFST set to 3'b001. DSRR0 points to the 3rd instruction. **Note:** In this case if x==y, then the resultant state of DBSR and DSRR0 may be indistinguishable from the "no DACy" case.
DVC DACx	Any instruction (no exception)	DSI, with or without DAC, Normal Ldst or multiple word Ldst	Take Debug exception, DBSR update setting DACx. DAC_OFST set to 3'b001. DSRR0 points to the 3rd instruction. No ESR update. **Note:** In this case the 3rd ld/st exception is masked. This behavior is implementation dependent and may differ on other CPUs.
DVC DACx	Any instruction (no exception)	DACy, or DVC DAC_y Normal Ldst or multiple word Ldst	Take Debug exception, DBSR update setting DACx, DACy. DAC_OFST set to 3'b010. DSRR0 points to instruction after the 3rd class instruction. **Note:** In this case if x==y, then the resultant state of DBSR and DSRR0 may be indistinguishable from the "no DACy" case.

Table 1058 through *Table 1061* show some example updates for specific code sequences of dual issuing of load/store class instructions with non-load/store class instructions and the results of DAC and DVC events on selected ones of the load/store instructions.

Table 1058. DAC events and Resultant Updates, Dual-issue case 1

Instruction Sequence: The following pairs dual-issue: (1) load/store (2) alu (3) load/store (4) alu (5) load/store (6) alu	Event(s)	Result
	Instruction (1): DSI, no DAC	Take DSI exception, no DBSR update, Update ESR.
	Instruction (1): DSI, with DACx	Take Debug exception, DBSR update setting DACx and IDE, DAC_OFST set to 3'b000. DSRR0 points to instruction (1). No ESR update.
	Instruction (1): DACx	Take Debug exception, DBSR update setting DACx, DAC_OFST set to 3'b000. DSRR0 points to instruction (2). No ESR update.
	Instruction (1): DVC DACx No other exceptions	Take Debug exception, DBSR update setting DACx, DAC_OFST set to 3'b100. DSRR0 points to instruction (6). No ESR update. No debug event updates for instruction (6)
	Instruction (1): DVC DACx Instruction (3): DSI, with or without DAC	Take Debug exception, DBSR update setting DACx, DAC_OFST set to 3'b001. DSRR0 points to instruction (3). No ESR update. No debug event updates for instructions (3)–(6). **Note:** In this case the 2nd ld/st exception is masked. This behavior is implementation dependent and may differ on other CPUs.
	Instruction (1): DVC DACx Instruction (3): DACy	Take Debug exception, DBSR update setting DACx, DACy. DAC_OFST set to 3'b010. DSRR0 points to instruction (4). No debug event updates for instructions (4)–(6). **Note:** In this case if x==y, then the resultant state of DBSR and DSRR0 may be indistinguishable from the "no DACy" case.
	Instruction (1): DVC DACx Instruction (3): DVC DACy	Take Debug exception, DBSR update setting DACx, DAC_OFST set to 3'b100. DSRR0 points to instruction (6). No ESR update. No debug event updates for instruction (6). **Note:** In this case if x==y, then the resultant state of DBSR and DSRR0 may be indistinguishable from the "no DACy" case.
	Instruction (1): DVC DACx Instruction (3): DVC DACy Instruction (5): DSI, with or without DAC	Take Debug exception, DBSR update setting DACx, DACy. DAC_OFST set to 3'b010. No ESR update. DSRR0 points to instruction (4). No debug event updates for instructions (4)–(6). **Note:** Note: in this case if x==y, then the resultant state of DBSR and DSRR0 may be indistinguishable from the "no DACy" case. **Note:** In this case the 3rd ld/st exception is masked. This behavior is implementation dependent and may differ on other CPUs.
	Instruction (1): DVC DACx Instruction (3): DVC DACy Instruction (5): DACy or DVC DACy	Take Debug exception, DBSR update setting DACx, DACy. DAC_OFST set to 3'b100. No ESR update. DSRR0 points to instruction (6). **Note:** In this case if x==y, then the resultant state of DBSR and DSRR0 may be indistinguishable from the "no DACy" case.

Table 1059. DAC events and Resultant Updates, Dual-issue case 2

Instruction Sequence: The following pairs dual-issue: (1) load/store (2) alu (3) load/store (4) alu (5) alu (6) load/store	Event(s)	Result
	Instruction (1): DSI, no DAC	Take DSI exception, no DBSR update, Update ESR.
	Instruction (1): DSI, with DACx	Take Debug exception, DBSR update setting DACx and IDE, DAC_OFST set to 3'b000. DSRR0 points to instruction (1). No ESR update.
	Instruction (1): DACx	Take Debug exception, DBSR update setting DACx, DAC_OFST set to 3'b000. DSRR0 points to instruction (2). No ESR update.
	Instruction (1): DVC DACx No other exceptions	Take Debug exception, DBSR update setting DACx, DAC_OFST set to 3'b101. DSRR0 points to instruction after instruction (6). No ESR update.
	Instruction (1): DVC DACx Instruction (3): DSI, with or without DAC	Take Debug exception, DBSR update setting DACx, DAC_OFST set to 3'b001. DSRR0 points to instruction (3). No ESR update. No debug event updates for instructions (3)–(6). **Note:** In this case the 2nd ld/st exception is masked. This behavior is implementation dependent and may differ on other CPUs.
	Instruction (1): DVC DACx Instruction (3): DACy	Take Debug exception, DBSR update setting DACx, DACy. DAC_OFST set to 3'b010. DSRR0 points to instruction (4). No debug event updates for instructions (4)–(6). **Note:** In this case if x==y, then the resultant state of DBSR and DSRR0 may be indistinguishable from the "no DACy" case.
	Instruction (1): DVC DACx Instruction (3): DVC DACy	Take Debug exception, DBSR update setting DACx, DAC_OFST set to 3'b101. DSRR0 points to instruction (7). No ESR update. **Note:** In this case if x==y, then the resultant state of DBSR and DSRR0 may be indistinguishable from the "no DACy" case.
	Instruction (1): DVC DACx Instruction (3): DVC DACy Instruction (6): DSI, with or without DAC	Take Debug exception, DBSR update setting DACx, DACy. DAC_OFST set to 3'b010. No ESR update. DSRR0 points to instruction (4). No debug event updates for instruction (4). **Note:** In this case if x==y, then the resultant state of DBSR and DSRR0 may be indistinguishable from the "no DACy" case. In this case the 3rd ld/st exception is masked. This behavior is implementation dependent and may differ on other CPUs.
	Instruction (1): DVC DACx Instruction (3): DVC DACy Instruction (6): DACy or DVC DACy	Take Debug exception, DBSR update setting DACx, DACy. DAC_OFST set to 3'b101. No ESR update. DSRR0 points to instruction (7). No debug event updates for instruction (7). **Note:** In this case if x==y, then the resultant state of DBSR and DSRR0 may be indistinguishable from the "no DACy" case.

Table 1060. DAC events and Resultant Updates, Dual-issue case 3

Instruction Sequence: The following pairs dual-issue: (1) load/store (2) alu (3) alu (4) alu (5) load/store (6) alu	Event(s)	Result
	Instruction (1): DSI, no DAC	Take DSI exception, no DBSR update, Update ESR.
	Instruction (1): DSI, with DACx	Take Debug exception, DBSR update setting DACx and IDE, DAC_OFST set to 3'b000. DSRR0 points to instruction (1). No ESR update.
	Instruction (1): DACx	Take Debug exception, DBSR update setting DACx, DAC_OFST set to 3'b000. DSRR0 points to instruction (2). No ESR update.
	Instruction (1): DVC DACx No other exceptions	Take Debug exception, DBSR update setting DACx, DAC_OFST set to 3'b100. DSRR0 points to instruction (6). No ESR update. No debug event updates for instruction (6).
	Instruction (1): DVC DACx Instruction (5): DSI, with or without DAC	Take Debug exception, DBSR update setting DACx, DAC_OFST set to 3'b011. DSRR0 points to instruction (5). No ESR update. No debug event updates for instructions (5)–(6). **Note:** In this case the 2nd ld/st exception is masked. This behavior is implementation dependent and may differ on other CPUs.
	Instruction (1): DVC DACx Instruction (5): DACy	Take Debug exception, DBSR update setting DACx, DACy. DAC_OFST set to 3'b100. DSRR0 points to instruction (6). No debug event updates for instruction (6). **Note:** In this case if x==y, then the resultant state of DBSR and DSRR0 may be indistinguishable from the "no DACy" case.
	Instruction (1): DVC DACx Instruction (5): DVC DACy	Take Debug exception, DBSR update setting DACx, DAC_OFST set to 3'b100. DSRR0 points to instruction (6). No ESR update. No debug event updates for instruction (6) **Note:** In this case if x==y, then the resultant state of DBSR and DSRR0 may be indistinguishable from the "no DACy" case.

Table 1061. DAC events and Resultant Updates, Dual-issue case 4

Instruction Sequence: The following pairs dual-issue: (1) load/store (2) alu (3) load/store (4) alu (5) alu (6) alu	Event(s)	Result
	Instruction (1): DSI, no DAC	Take DSI exception, no DBSR update, Update ESR.
	Instruction (1): DSI, with DACx	Take Debug exception, DBSR update setting DACx and IDE, DAC_OFST set to 3'b000. DSRR0 points to instruction (1). No ESR update.
	Instruction (1): DACx	Take Debug exception, DBSR update setting DACx, DAC_OFST set to 3'b000. DSRR0 points to instruction (2). No ESR update.
	Instruction (1): DVC DACx No other exceptions	Take Debug exception, DBSR update setting DACx, DAC_OFST set to 3'b011. DSRR0 points to instruction (5). No ESR update. No debug event updates for instructions (5)–(6)
	Instruction (1): DVC DACx Instruction (3): DSI, with or without DAC	Take Debug exception, DBSR update setting DACx, DAC_OFST set to 3'b001. DSRR0 points to instruction (3). No ESR update. No debug event updates for instructions (3)–(6). **Note:** In this case the 2nd ld/st exception is masked. This behavior is implementation dependent and may differ on other CPUs.
	Instruction (1): DVC DACx Instruction (3): DACy	Take Debug exception, DBSR update setting DACx, DACy. DAC_OFST set to 3'b010. DSRR0 points to instruction (4). No debug event updates for instructions (4)–(6). **Note:** In this case if x==y, then the resultant state of DBSR and DSRR0 may be indistinguishable from the "no DACy" case.
	Instruction (1): DVC DACx Instruction (3): DVC DACy	Take Debug exception, DBSR update setting DACx, DAC_OFST set to 3'b011. DSRR0 points to instruction (5). No ESR update. No debug event updates for instructions (5)–(6). **Note:** In this case if x==y, then the resultant state of DBSR and DSRR0 may be indistinguishable from the "no DACy" case.

53.2.3 Linked Instruction Address Compare and Data Address Compare Events

Data Address Compare 1, 2, 3, and 4 debug events may be 'linked' with an Instruction Address Compare event by setting the DAC[1–4]LNK control bits in DBCR2 and DBCR8 to further refine when a Data Address Compare debug event is generated. DAC1 may be linked with IAC1, DAC2 (when not used as a mask or range bounds register) may be linked with IAC3. DAC3 may be linked with IAC5, DAC4 (when not used as a mask or range bounds register) may be linked with IAC7. When linked, a DAC debug event occurs when the same instruction which generates the DAC 'hit' also generates a corresponding linked IAC 'hit'. When linked, the IAC event is not recorded in the Debug Status register, regardless

of whether a corresponding linked DAC event occurs, or whether the IAC event enable is set.

When enabled and execution of a load or store class instruction results in a data access with an address that meets the criteria specified in the DBCRx, DACx, and DVCx Registers, and the instruction also meets the criteria for generating an Instruction Address Compare event, a Linked Data Address Compare debug event occurs. This event can occur and be recorded in DBSR regardless of the setting of MSR_{DE}. The normal DAC1 status bit in the DBSR is used for recording these events. The IAC status bit is not set if the corresponding Instruction Address Compare register is linked.

Linking is enabled using control bits in DBCR2 and DBCR8. Note that linking is only available in EDM or IDM. Attempts to use linking otherwise are ignored.

Note: Linked DAC events will not be recorded if a load multiple word or store multiple word type instruction is interrupted prior to completion by a critical input or external input interrupt.

53.2.4 Trap Debug Event

A Trap debug event (TRAP) occurs if Trap debug events are enabled ($DBCR0_{TRAP}$ = 1), a Trap instruction (**tw**) is executed, and the conditions specified by the instruction for the trap are met. This event can occur and be recorded in DBSR regardless of the setting of MSR_{DE}. When a Trap debug event occurs, the $DBSR_{TRAP}$ bit is set to 1 to record the debug exception.

53.2.5 Branch Taken Debug Event

A Branch Taken debug event (BRT) occurs if Branch Taken debug events are enabled ($DBCR0_{BRT}$ = 1) and execution is attempted of a branch instruction which will be taken (either an unconditional branch, or a conditional branch whose branch condition is true), and MSR_{DE} = 1. Branch Taken debug events are not recognized if MSR_{DE} = 0 at the time of execution of the branch instruction and thus $DBSR_{IDE}$ can not be set by a Branch Taken debug event. When a Branch Taken debug event is recognized, the $DBSR_{BRT}$ bit is set to 1 to record the debug exception, and the address of the branch instruction will be recorded in DSRR0.

53.2.6 Instruction Complete Debug Event

An Instruction Complete debug event (ICMP) occurs if Instruction Complete debug events are enabled ($DBCR0_{ICMP}$ = 1), execution of any instruction is completed, and MSR_{DE} = 1. If execution of an instruction is suppressed due to the instruction causing some other exception which is enabled to generate an interrupt, then the attempted execution of that instruction does not cause an Instruction Complete debug event. The ***se_sc*** instruction does not fall into the category of an instruction whose execution is suppressed, since the instruction actually executes and then generates a System Call interrupt. In this case, the Instruction Complete debug exception will also be set. When an Instruction Complete debug event is recognized, $DBSR_{ICMP}$ is set to 1 to record the debug exception and the address of the next instruction to be executed will be recorded in DSRR0.

Instruction Complete debug events are not recognized if MSR_{DE} = 0 at the time of execution of the instruction, thus $DBSR_{IDE}$ is not generally set by an ICMP debug event.

One circumstance may cause the $DBSR_{ICMP}$ and $DBSR_{IDE}$ bits to be set. This occurs when a EFPU Round exception occurs. Since the instruction is by definition completed (SRR0 points to the following instruction), this interrupt takes higher priority than the Debug

interrupt so as not to be lost, and $DBSR_{IDE}$ is set to indicate the imprecise recognition of a Debug interrupt. In this case, the Debug interrupt will be taken with SRR0 pointing to the instruction following the instruction which generated the EFPU Round exception, and DSRR0 will point to the Round exception handler. In addition to occurring when $DBCR0_{IDM}$ = 1, this circumstance can also occur when $EDBCR0_{EDM}$ = 1 and the event is hardware-owned, in which case $EDBSR0_{IDE}$ will be set.

Note: Instruction complete debug events are not generated by the execution of an instruction which sets MSR_{DE} to '1' while $DBCR0_{ICMP}$ = 1, nor by the execution of an instruction which sets $DBCR0_{ICMP}$ to '1' while MSR_{DE} = 1.

53.2.7 Interrupt Taken Debug Event

An Interrupt Taken debug event (IRPT) occurs if Interrupt Taken debug events are enabled ($DBCR0_{IRPT}$ = 1) and a base-class interrupt occurs. Only base-class interrupts (an interrupt using SRR0/1) cause an Interrupt Taken debug event. This event can occur and be recorded in DBSR regardless of the setting of MSR_{DE}. When an Interrupt Taken debug event occurs, the $DBSR_{IRPT}$ bit is set to 1 to record the debug exception. The value saved in DSRR0 will be the address of the non-critical interrupt handler.

53.2.8 Critical Interrupt Taken Debug Event

A Critical Interrupt Taken debug event (CIRPT) occurs if Critical Interrupt Taken debug events are enabled ($DBCR0_{CIRPT}$ = 1) and a critical interrupt occurs. Only critical class interrupts (an interrupt using CSRR0/1) cause a Critical Interrupt Taken debug event. This event can occur and be recorded in DBSR regardless of the setting of MSR_{DE}. When a Critical Interrupt Taken debug event occurs, the $DBSR_{CIRPT}$ bit is set to 1 to record the debug exception. The value saved in DSRR0 will be the address of the critical interrupt handler.

53.2.9 Return Debug Event

A Return debug event (RET) occurs if Return debug events are enabled ($DBCR0_{RET}$ = 1) and an attempt is made to execute an **se_rfi** instruction. This event can occur and be recorded in DBSR regardless of the setting of MSR_{DE}. When a Return debug event occurs, the $DBSR_{RET}$ bit is set to 1 to record the debug exception.

If MSR_{DE} = 0 at the time of the execution of the **se_rfi** (that is before the MSR is updated by the **se_rfi**), then $DBSR_{IDE}$ is also set to 1 to record the imprecise debug event.

If MSR_{DE} = 1 at the time of the execution of the **se_rfi**, a Debug interrupt will occur provided there exists no higher priority exception which is enabled to cause an interrupt. Debug Save/Restore Register 0 will be set to the address of the **se_rfi** instruction.

53.2.10 Critical Return Debug Event

A Critical Return debug event (CRET) occurs if Critical Return debug events are enabled ($DBCR0_{CRET}$ = 1) and an attempt is made to execute an **se_rfci** instruction. This event can occur and be recorded in DBSR regardless of the setting of MSR_{DE}. When a Critical Return debug event occurs, the $DBSR_{CRET}$ bit is set to 1 to record the debug exception.

If MSR_{DE} = 0 at the time of the execution of the **se_rfci** (that is before the MSR is updated by the **se_rfci**), then $DBSR_{IDE}$ is also set to 1 to record the imprecise debug event.

If MSR_{DE} = 1 at the time of the execution of the **se_rfci**, a Debug interrupt will occur provided there exists no higher priority exception which is enabled to cause an interrupt. Debug Save/Restore Register 0 will be set to the address of the **se_rfci** instruction.

53.2.11 External Debug Event

An External debug event (DEVT1, DEVT2) occurs if External debug events are enabled ($DBCR0_{DEVT1}$ = 1 or $DBCR0_{DEVT2}$ = 1), and the respective **p_devt1** or **p_devt2** input signal transitions to the asserted state while the CPU is not in the Stopped state. This event can occur and be recorded in DBSR regardless of the setting of MSR_{DE}. When an External debug event occurs, $DBSR_{DEVT\{1,2\}}$ is set to '1' to record the debug exception. This debug event is an asynchronous event, but is only sampled when the CPU **m_clk** is active.

53.2.12 Unconditional Debug Event

An Unconditional debug event (UDE) occurs when the Unconditional Debug Event (**p_ude**) input transitions to the asserted state. The Unconditional debug event is the only debug event which does not have a corresponding enable bit for the event in DBCR0. This event can occur and be recorded in DBSR regardless of the setting of MSR_{DE}. When an Unconditional debug event occurs, the $DBSR_{UDE}$ bit is set to '1' to record the debug exception. This debug event is an asynchronous event.

53.2.13 Performance Monitor Interrupt Debug Event

A Performance Monitor Interrupt debug event (PMI) occurs if Performance Monitor Interrupt debug events are enabled ($PMGC0_{UDI}$ = 1), and a performance monitor interrupt event occurs. This event can occur and be recorded in DBSR regardless of the setting of MSR_{DE}. When a Performance Monitor Interrupt debug event occurs, $DBSR_{PMI}$ is set to '1' to record the debug exception. This debug event is an asynchronous event.

53.3 Debug Registers

This section describes debug-related registers that are software accessible. These registers are intended for use by special debug tools and debug software, not by general application code.

Access to these registers (other than DBSR) by software is conditioned by the External Debug Mode control bit ($EDBCR0_{EDM}$) and the settings of debug control register EDBRAC0, which can be set by the hardware debug port. If $EDBCR0_{EDM}$ is set and if the bit in EDBRAC0 corresponding to the resource is cleared, software is prevented from modifying debug register values other than in DBSR, since the resource is not "owned" by software. Software always has ownership of DBSR. Execution of a **mtspr** instruction targeting a debug register or register field not "owned" by software will not cause modifications to occur, and no exception will be signaled. In addition, since the external debugger hardware may be manipulating debug register values, the state of these registers or register fields not "owned" by software is not guaranteed to be consistent if accessed (read) by software with a **mfspr** instruction, other than the $DBCR0_{EDM}$ bit itself and the EDBRAC0 register. Hardware always has full access to all registers and all register fields through the OnCE register access mechanism, and it is up to the debug firmware to properly implement modifications to these registers with read-modify-write operations to implement any control sharing with software. Settings in EDBRAC0 should be considered by the debug

firmware in order to preserve software settings of control registers as appropriate when hardware modifications to the debug registers is performed.

53.3.1 Debug Address and Value Registers

Instruction Address Compare registers IAC1–8 are used to hold instruction addresses for address comparison purposes. In addition, IAC2 and IAC4 may hold mask information for IAC1 and IAC3 respectively and IAC6 and IAC8 may hold mask information for IAC5 and IAC7 respectively, when *Address Bit Match* compare modes are selected. Note that when performing instruction address compares, the low order bit of the instruction address and the corresponding IAC register is ignored for VLE instructions.

Data Address Compare registers DAC1–4 are used to hold data access addresses for address comparison purposes. In addition, DAC2 and DAC4 may hold mask information for DAC1 and DAC3 respectively when *Address Bit Match* compare modes are selected.

Data Value Compare registers DVC1 and DVC2 are used to hold data values for data comparison purposes. DVC1 and DVC2 are 64-bit registers. Data value comparisons are used to qualify Data Address compare 1 and 2 debug events. Data value comparisons are not available for Data address compare 3–4 events. DVC1 is associated with DAC1, and DVC2 is associated with DAC2. The most significant byte of the DVC1(2) register (labeled B0 in *Figure 1098*) corresponds to the byte data value transferred to/from memory byte offset 0, 8, …, and the least significant byte of the register (labeled B7 in *Figure 1099*) corresponds to byte offset 7, F, …. When enabled for performing data value comparisons, each enabled byte in DVC1(2) is compared with the memory value transferred on the corresponding active byte lane of the data memory interface to determine if a match occurs. Inactive byte lanes do not participate in the comparison, they are implicitly masked. The Byte Strobe Assertion for Transfers table in the Core (e200z420n3) Core Complex Overview chapter shows active byte lanes for data transfers. Software must also program the DVC1(2) register byte positions based on the alignment of the access. Misaligned accesses are not fully supported, since the data address and data value comparisons are only performed on the initial access in the case of a misaligned access; thus, accesses which cross a 64-bit boundary cannot be fully matched. For address and size combinations which involve two transfers, only the initial transfer is used for data address and value matching.

DVC1 and DVC2 may be read or written using **mtspr** and **mfspr** instructions. The DVC1U and DVC2U SPR numbers (601,602) are used for accessing the upper 32-bit portion of the DVC register. The DVC1 and DVC2 SPR numbers (318,319) are used for accessing the lower 32-bit portion of the DVC register.

Offset: Refer to SPR - 318 (DVC1)
Refer to SPR - 319 (DVC2)
Access: Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	B0								B1							
Reset	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	B2								B3							
Reset	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—

Figure 1098. Debug Address and Value Registers (DVC1, DVC2)

Offset: Refer to SPR - 601 (DVC1U)
Refer to SPR - 602 (DVC2U)
Access: Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	B4								B5							
Reset	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	B6								B7							
Reset	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—

Figure 1099. Debug Address and Value Registers - Upper portion (DVC1U, DVC2U)

53.3.2 Debug Control and Status Registers

Debug Control Registers (DBCR0–8 and EDBRAC0) are used to enable debug events, reset the processor, and set the debug mode of the processor. The Debug Status register (DBSR) records debug exceptions while Internal Debug Mode is enabled. The Debug Data Effective Address register (DDEAR) records the effective address of a Data Address Compare event while Internal Debug Mode is enabled.

e200z420n3 requires that a context synchronizing instruction follow a **mtspr** DBCR0–8 or DBSR to ensure that any alterations enabling/disabling debug events are effective. The context synchronizing instruction may or may not be affected by the alteration. Typically, an **se_isync** instruction is used to create a synchronization boundary beyond which it can be guaranteed that the newly written control values are in effect.

For watchpoint generation, configuration settings contained in DBCR1–8 are used, even though the corresponding event(s) may be disabled (via DBCR0) from setting DBSR flags.

53.3.2.1 Debug Control Register 0 (DBCR0)

Debug Control Register 0 is used to enable debug modes and controls which debug events are allowed to set DBSR or EDBSR0 flags. e200z420n3 adds some implementation specific bits to this register, as seen in *Figure 1100*.

Offset: Refer to SPR - 308 Access: Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	EDM	IDM	RST		ICMP	BRT	IRPT	TRAP	IAC1	IAC2	IAC3	IAC4	DAC1		DAC2	
W																
Reset(1)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	RET	IAC5	IAC6	IAC7	IAC8	DEVT1	DEVT2	0		CIRPT	CRET	0				
W																
Reset(1)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. $DBCR0_{EDM}$ is affected by **j_trst_b** or **m_por** assertion, and remains reset while in the Test_Logic_Reset state, but is not affected by **p_reset_b**. All other bits are reset by processor reset **p_reset_b** if $DBCR0_{EDM}$ = 0, as well as unconditionally by **m_por**. If $DBCR0_{EDM}$ = 1, EDBRAC0 masks off hardware-owned resources (other than RST) from reset by **p_reset_b,** and only software-owned resources indicated by EDBRAC0 and the $DBCR0_{RST}$ field will be reset by **p_reset_b**. The $DBCR0_{RST}$ field will always be reset by **p_reset_b** regardless of the value of $DBCR0_{EDM}$.

Figure 1100. Debug Control Register 0 Register (DBCR0)

Table 1062 provides bit definitions for Debug Control Register 0.

Table 1062. DBCR0 Field descriptions

Field	Description
0 EDM	External Debug Mode This bit is read-only by software. 0 External debug mode disabled. Internal debug events not mapped into external debug events. 1 External debug mode enabled. Hardware-owned debug events will not cause the CPU to vector to interrupt code. Software is not permitted to write to debug registers {DBCR0–8, IAC1–8, DAC1–4, DVC1–2[U]} unless permitted by settings in EDBRAC0. Hardware-owned events will set status bits in EDBSR0. When external debug mode is enabled, hardware-owned resources in debug registers are not affected by processor reset **p_reset_b**. This allows the debugger to set up hardware debug events which remain active across a processor reset. Programming: It is recommended that debug status bits in the Debug Status Registers be cleared before disabling external debug mode to avoid any internal imprecise debug interrupts. Software may use this bit to determine if external debug has control over the debug registers. The hardware debugger must set the EDM bit to '1' before other bits in this register (and other debug registers) may be altered. On the initial setting of this bit to '1', all other bits are unchanged. This bit is only writable through the OnCE port.
1 IDM	Internal Debug Mode 0 Debug exceptions are disabled. Debug events do not affect DBSR. 1 Debug exceptions are enabled. Enabled debug events owned by software will update the DBSR. If MSR_{DE} = 1, the occurrence of a debug event, or the recording of an earlier debug event in the Debug Status Register when MSR_{DE} was cleared, will cause a Debug interrupt.

Table 1062. DBCR0 Field descriptions (continued)

Field	Description
2:3 RST	Reset Control 00 No function 01 **p_dbrstc[1]** pin asserted by Debug Reset Control. Allows external device to initiate processor or system reset 10 **p_dbrstc[0]** pin asserted by Debug Reset Control. Allows external device to initiate processor or system reset. 11 Reserved
4 ICMP	Instruction Complete Debug Event Enable 0 ICMP debug events are disabled 1 ICMP debug events are enabled
5 BRT	Branch Taken Debug Event Enable 0 BRT debug events are disabled 1 BRT debug events are enabled
6 IRPT	Interrupt Taken Debug Event Enable 0 IRPT debug events are disabled 1 IRPT debug events are enabled
7 TRAP	Trap Taken Debug Event Enable 0 TRAP debug events are disabled 1 TRAP debug events are enabled
8 IAC1	Instruction Address Compare 1 Debug Event Enable 0 IAC1 debug events are disabled 1 IAC1 debug events are enabled
9 IAC2	Instruction Address Compare 2 Debug Event Enable 0 IAC2 debug events are disabled 1 IAC2 debug events are enabled
10 IAC3	Instruction Address Compare 3 Debug Event Enable 0 IAC3 debug events are disabled 1 IAC3 debug events are enabled
11 IAC4	Instruction Address Compare 4 Debug Event Enable 0 IAC4 debug events are disabled 1 IAC4 debug events are enabled
12:13 DAC1	Data Address Compare 1 Debug Event Enable 00 DAC1 debug events are disabled 01 DAC1 debug events are enabled only for store-type data storage accesses 10 DAC1 debug events are enabled only for load-type data storage accesses 11 DAC1 debug events are enabled for load-type or store-type data storage accesses
14:15 DAC2	Data Address Compare 2 Debug Event Enable 00 DAC2 debug events are disabled 01 DAC2 debug events are enabled only for store-type data storage accesses 10 DAC2 debug events are enabled only for load-type data storage accesses 11 DAC2 debug events are enabled for load-type or store-type data storage accesses
16 RET	Return Debug Event Enable 0 RET debug events are disabled 1 RET debug events are enabled

Table 1062. DBCR0 Field descriptions (continued)

Field	Description
17 IAC5	Instruction Address Compare 5 Debug Event Enable 0 IAC5 debug events are disabled 1 IAC5 debug events are enabled
18 IAC6	Instruction Address Compare 6 Debug Event Enable 0 IAC6 debug events are disabled 1 IAC6 debug events are enabled
19 IAC7	Instruction Address Compare 7 Debug Event Enable 0 IAC7 debug events are disabled 1 IAC7 debug events are enabled
20 IAC8	Instruction Address Compare 8 Debug Event Enable 0 IAC8 debug events are disabled 1 IAC8 debug events are enabled
21 DEVT1	External Debug Event 1 Enable 0 DEVT1 debug events are disabled 1 DEVT1 debug events are enabled
22 DEVT2	External Debug Event 2 Enable 0 DEVT2 debug events are disabled 1 DEVT2 debug events are enabled
25 CIRPT	Critical Interrupt Taken Debug Event Enable 0 CIRPT debug events are disabled 1 CIRPT debug events are enabled
26 CRET	Critical Return Debug Event Enable 0 CRET debug events are disabled 1 CRET debug events are enabled

53.3.2.2 Debug Control Register 1 (DBCR1)

Debug Control Register 1 is used to configure Instruction Address Compare operation. The DBCR1 register is shown in *Figure 1101*.

Offset: Refer to SPR - 309 Access: Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	IAC1US		IAC1ER		IAC2US		IAC2ER		IAC12M		0					
W																
Reset(1)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	IAC3US		IAC3ER		IAC4US		IAC4ER		IAC34M		0					
W																
Reset(1)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. Reset by processor reset **p_reset_b** if $EDBCR0_{EDM}$ = 0, as well as unconditionally by **m_por**. If $EDBCR0_{EDM}$ = 1, EDBRAC0 masks off hardware-owned resources from reset by **p_reset_b** and only software-owned resources indicated by EDBRAC0 will be reset by **p_reset_b.**

Figure 1101. Debug Control Register 1 (DBCR1)

Table 1063 provides bit definitions for Debug Control Register 1.

Table 1063. DBCR1 field descriptions

Field	Description
0:1 IAC1US	Instruction Address Compare 1 User/Supervisor Mode 00 IAC1 debug events not affected by MSR_{PR} 01 Reserved 10 IAC1 debug events can only occur if MSR_{PR} = 0 (Supervisor mode) 11 IAC1 debug events can only occur if MSR_{PR} = 1. (User mode)
2:3 IAC1ER	Instruction Address Compare 1 Effective/Real Mode 00 IAC1 debug events are based on effective address 01 Unimplemented (Book E real address compare), no match can occur 10 IAC1 debug events are based on effective address and can only occur if MSR_{IS} = 0 11 IAC1 debug events are based on effective address and can only occur if MSR_{IS} = 1
4:5 IAC2US	Instruction Address Compare 2 User/Supervisor Mode 00 IAC2 debug events not affected by MSR_{PR} 01 Reserved 10 IAC2 debug events can only occur if MSR_{PR} = 0 (Supervisor mode) 11 IAC2 debug events can only occur if MSR_{PR} = 1. (User mode)
6:7 IAC2ER	Instruction Address Compare 2 Effective/Real Mode 00 IAC2 debug events are based on effective address 01 Unimplemented (Book E real address compare), no match can occur 10 IAC2 debug events are based on effective address and can only occur if MSR_{IS} = 0 11 IAC2 debug events are based on effective address and can only occur if MSR_{IS} = 1
8:9 IAC12M	Instruction Address Compare 1/2 Mode 00 Exact address compare. IAC1 debug events can only occur if the address of the instruction fetch is equal to the value specified in IAC1. IAC2 debug events can only occur if the address of the instruction fetch is equal to the value specified in IAC2. 01 Address bit match. IAC1 debug events can occur only if the address of the instruction fetch, ANDed with the contents of IAC2 are equal to the contents of IAC1, also ANDed with the contents of IAC2. IAC2 debug events do not occur. IAC1US and IAC1ER settings are used. 10 Inclusive address range compare. IAC1 debug events can occur only if the address of the instruction fetch is greater than or equal to the value specified in IAC1 and less than the value specified in IAC2. IAC2 debug events do not occur. IAC1US and IAC1ER settings are used. 11 Exclusive address range compare. IAC1 debug events can occur only if the address of the instruction fetch is less than the value specified in IAC1 or is greater than or equal to the value specified in IAC2. IAC2 debug events do not occur. IAC1US and IAC1ER settings are used.
16:17 IAC3US	Instruction Address Compare 3 User/Supervisor Mode 00 IAC3 debug events not affected by MSR_{PR} 01 Reserved 10 IAC3 debug events can only occur if MSR_{PR} = 0 (Supervisor mode) 11 IAC3 debug events can only occur if MSR_{PR} = 1 (User mode)
18:19 IAC3ER	Instruction Address Compare 3 Effective/Real Mode 00 IAC3 debug events are based on effective address 01 Unimplemented (Book E real address compare), no match can occur 10 IAC3 debug events are based on effective address and can only occur if MSR_{IS} = 0 11 IAC3 debug events are based on effective address and can only occur if MSR_{IS} = 1

Table 1063. DBCR1 field descriptions (continued)

Field	Description
20:21 IAC4US	Instruction Address Compare 4 User/Supervisor Mode 00 IAC4 debug events not affected by MSR_{PR} 01 Reserved 10 IAC4 debug events can only occur if MSR_{PR} = 0 (Supervisor mode). 11 IAC4 debug events can only occur if MSR_{PR} = 1. (User mode)
22:23 IAC4ER	Instruction Address Compare 4 Effective/Real Mode 00 IAC4 debug events are based on effective address 01 Unimplemented (Book E real address compare), no match can occur 10 IAC4 debug events are based on effective address and can only occur if MSR_{IS} = 0 11 IAC4 debug events are based on effective address and can only occur if MSR_{IS} = 1
24:25 IAC34M	Instruction Address Compare 3/4 Mode 00 Exact address compare. IAC3 debug events can only occur if the address of the instruction fetch is equal to the value specified in IAC3. IAC4 debug events can only occur if the address of the instruction fetch is equal to the value specified in IAC4. 01 Address bit match. IAC3 debug events can occur only if the address of the instruction fetch, ANDed with the contents of IAC4 are equal to the contents of IAC3, also ANDed with the contents of IAC4. IAC4 debug events do not occur. IAC3US and IAC3ER settings are used. 10 Inclusive address range compare. IAC3 debug events can occur only if the address of the instruction fetch is greater than or equal to the value specified in IAC3 and less than the value specified in IAC4. IAC4 debug events do not occur. IAC3US and IAC3ER settings are used. 11 Exclusive address range compare. IAC3 debug events can occur only if the address of the instruction fetch is less than the value specified in IAC3 or is greater than or equal to the value specified in IAC4. IAC4 debug events do not occur. IAC3US and IAC3ER settings are used.

53.3.2.3 Debug Control Register 2 (DBCR2)

Debug Control Register 2 is used to configure Data Address Compare and Data Value Compare operation.The DBCR2 register is shown in *Figure 1102*.

Offset: Refer to SPR - 310 Access: Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	DAC1US		DAC1ER		DAC2US		DAC2ER		DAC12M		DAC1LNK	DAC2LNK	DVC1M		DVC2M	
Reset(1)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	DVC1BE								DVC2BE							
Reset(1)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. Reset by processor reset **p_reset_b** if $EDBCR0_{EDM}$ = 0, as well as unconditionally by **m_por**. If $EDBCR0_{EDM}$ = 1, EDBRAC0 masks off hardware-owned resources from reset by **p_reset_b** and only software-owned resources indicated by EDBRAC0 will be reset by **p_reset_b**.

Figure 1102. Debug Control Register 2 (DBCR2)

Table 1064 provides bit definitions for Debug Control Register 2.

Table 1064. DBCR2 field descriptions

Field	Description
0:1 DAC1US	Data Address Compare 1 User/Supervisor Mode 00 DAC1 debug events not affected by MSR_{PR} 01 Reserved 10 DAC1 debug events can only occur if MSR_{PR} = 0 (Supervisor mode) 11 DAC1 debug events can only occur if MSR_{PR} = 1 (User mode)
2:3 DAC1ER	Data Address Compare 1 Effective/Real Mode 00 DAC1 debug events are based on effective address 01 Unimplemented (Book E real address compare), no match can occur 10 DAC1 debug events are based on effective address and can only occur if MSR_{DS} = 0 11 DAC1 debug events are based on effective address and can only occur if MSR_{DS} = 1
4:5 DAC2US	Data Address Compare 2 User/Supervisor Mode 00 DAC2 debug events not affected by MSR_{PR} 01 Reserved 10 DAC2 debug events can only occur if MSR_{PR} = 0 (Supervisor mode) 11 DAC2 debug events can only occur if MSR_{PR} = 1. (User mode)
6:7 DAC2ER	Data Address Compare 2 Effective/Real Mode 00 DAC2 debug events are based on effective address 01 Unimplemented (Book E real address compare), no match can occur 10 DAC2 debug events are based on effective address and can only occur if MSR_{DS} = 0 11 DAC2 debug events are based on effective address and can only occur if MSR_{DS} = 1
8:9 DAC12M	Data Address Compare 1/2 Mode 00 Exact address compare. DAC1 debug events can only occur if the address of the data access is equal to the value specified in DAC1. DAC2 debug events can only occur if the address of the data access is equal to the value specified in DAC2. 01 Address bit match. DAC1 debug events can occur only if the address of the data access ANDed with the contents of DAC2, are equal to the contents of DAC1 also ANDed with the contents of DAC2. DAC2 debug events do not occur. DAC1US and DAC1ER settings are used. 10 Inclusive address range compare. DAC1 debug events can occur only if the address of the data access is greater than or equal to the value specified in DAC1 and less than the value specified in DAC2. DAC2 debug events do not occur. DAC1US and DAC1ER settings are used. 11 Exclusive address range compare. DAC1 debug events can occur only if the address of the data access is less than the value specified in DAC1 or is greater than or equal to the value specified in DAC2. DAC2 debug events do not occur. DAC1US and DAC1ER settings are used.
10 DAC1LNK	Data Address Compare 1 Linked 0 No effect 1 DAC1 debug events are linked to IAC1 debug events. IAC1 debug events do not affect DBSR When linked to IAC1, DAC1 debug events are conditioned based on whether the instruction also generated an IAC1 debug event. Note that linking is only available in EDM or IDM.

Table 1064. DBCR2 field descriptions (continued)

Field	Description
11 DAC2LNK	Data Address Compare 2 Linked 0 No effect 1 DAC 2 debug events are linked to IAC3 debug events. IAC3 debug events do not affect DBSR When linked to IAC3, DAC2 debug events are conditioned based on whether the instruction also generated an IAC3 debug event. DAC2 can only be linked if DAC12M specifies *Exact Address Compare* since DAC2 debug events are not generated in the other compare modes. Note that linking is only available in EDM or IDM.
12:13 DVC1M	Data Value Compare 1 Mode When $DBCR4_{DVC1C}$ = 0: 00 DAC1 debug events not affected by data value compares. 01 DAC1 debug events can only occur when all bytes specified in the DVC1BE field match the corresponding data byte values for active byte lanes of the memory access. 10 DAC1 debug events can only occur when any byte specified in the DVC1BE field matches the corresponding data byte value for active byte lanes of the memory access. 11 DAC1 debug events can only occur when all bytes specified in the DVC1BE field within at least one of the halfwords of the data value of the memory access matches the corresponding DVC1 value. Inactive byte lanes of the memory access are automatically masked. When $DBCR4_{DVC1C}$ = 1: 00 Reserved 01 DAC1 debug events can only occur when any byte specified in the DVC1BE field does not match the corresponding data byte value for active byte lanes of the memory access. If all active bytes match, then no event will be generated. 10 DAC1 debug events can only occur when all bytes specified in the DVC1BE field do not match the corresponding data byte values for active byte lanes of the memory access. If any active byte match occurs, no event will be generated. 11 Reserved Inactive byte lanes of the memory access are automatically masked.

Table 1064. DBCR2 field descriptions (continued)

Field	Description
14:15 DVC2M	Data Value Compare 2 Mode When $DBCR4_{DVC2C}$ = 0: 00 DAC2 debug events not affected by data value compares. 01 DAC2 debug events can only occur when all bytes specified in the DVC2BE field match the corresponding data byte values for active byte lanes of the memory access. 10 DAC2 debug events can only occur when any byte specified in the DVC2BE field matches the corresponding data byte value for active byte lanes of the memory access. 11 DAC2 debug events can only occur when all bytes specified in the DVC2BE field within at least one of the halfwords of the data value of the memory access matches the corresponding DVC2 value. Inactive byte lanes of the memory access are automatically masked. When $DBCR4_{DVC2C}$ = 1: 00 Reserved 01 DAC2 debug events can only occur when any byte specified in the DVC2BE field does not match the corresponding data byte value for active byte lanes of the memory access. If all active bytes match, then no event will be generated. 10 DAC2 debug events can only occur when all bytes specified in the DVC2BE field do not match the corresponding data byte values for active byte lanes of the memory access. If any active byte match occurs, no event will be generated. 11 Reserved Inactive byte lanes of the memory access are automatically masked.

Table 1064. DBCR2 field descriptions (continued)

Field	Description
16:23 DVC1BE	Data Value Compare 1 Byte Enables Specifies which bytes in the aligned doubleword value associated with the memory access are compared to the corresponding bytes in DVC1. Inactive byte lanes of a memory access smaller than 64 bits are automatically masked by hardware. If all bits in the DVC1BE field are clear, then a match will occur regardless of the data. Misaligned accesses which cross a doubleword boundary are not fully supported. 1xxxxxxx Byte lane 0 is enabled for comparison with the value in bits 0:7 of DVC1. x1xxxxxx Byte lane 1 is enabled for comparison with the value in bits 8:15 of DVC1. xx1xxxxx Byte lane 2 is enabled for comparison with the value in bits 16:23 of DVC1. xxx1xxxx Byte lane 3 is enabled for comparison with the value in bits 24:31 of DVC1. xxxx1xxx Byte lane 4 is enabled for comparison with the value in bits 32:39 of DVC1. xxxxx1xx Byte lane 5 is enabled for comparison with the value in bits 40:47 of DVC1. xxxxxx1x Byte lane 6 is enabled for comparison with the value in bits 48:55 of DVC1. xxxxxxx1 Byte lane 7 is enabled for comparison with the value in bits 56:63 of DVC1.
24:31 24:31 DVC2BE	Data Value Compare2 Byte Enables Specifies which bytes in the aligned doubleword value associated with the memory access are compared to the corresponding bytes in DVC2. Inactive byte lanes of a memory access smaller than 64 bits are automatically masked by hardware. If all bits in the DVC1BE field are clear, then a match will occur regardless of the data. Misaligned accesses which cross a doubleword boundary are not fully supported. 1xxxxxxx Byte lane 0 is enabled for comparison with the value in bits 0:7 of DVC2. x1xxxxxx Byte lane 1 is enabled for comparison with the value in bits 8:15 of DVC2. xx1xxxxx Byte lane 2 is enabled for comparison with the value in bits 16:23 of DVC2. xxx1xxxx Byte lane 3 is enabled for comparison with the value in bits 24:31 of DVC2. xxxx1xxx Byte lane 4 is enabled for comparison with the value in bits 32:39 of DVC2. xxxxx1xx Byte lane 5 is enabled for comparison with the value in bits 40:47 of DVC2. xxxxxx1x Byte lane 6 is enabled for comparison with the value in bits 48:55 of DVC2. xxxxxxx1 Byte lane 7 is enabled for comparison with the value in bits 56:63 of DVC2.

53.3.2.4 Debug Control Register 4 (DBCR4)

Debug Control Register 4 is used to extend data address and value compare matching functionality. DBCR4 is shown in *Figure 1103*.

Offset: Refer to SPR - 563 Access: Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	DVC1C	0	DVC2C	0								0	DAC1XMH	0	DAC2XMH
W																
Reset(1)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	DAC1XM				DAC2XM				DAC1CFG				DAC2CFG			
W																
Reset(1)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. DBCR4 is reset by processor reset **p_reset_b** if $EDBCR0_{EDM}$ = 0, as well as unconditionally by **m_por**. If $EDBCR0_{EDM}$ = 1, EDBRAC0 masks off hardware-owned resources from reset by **p_reset_b** and only software-owned resources indicated by EDBRAC0 will be reset by **p_reset_b**.

Figure 1103. 32-bit register (REG NAME)

Table 1065 provides bit definitions for Debug Control Register 4.

Table 1065. DBCR4 Field descriptions

Field	Description
1 DVC1C	Data Value Compare 1 Control 0 Normal DVC1 operation. 1 Inverted polarity DVC1 operation DVC1C controls whether DVC1 data value comparisons utilize the normal compare operation, or an alternate “inverted compare” operation. In inverted polarity mode, data value compares perform a not-equal comparison. See details in the DBCR2 register definition.
3 DVC2C	Data Value Compare 2 Control 0 Normal DVC2 operation. 1 Inverted polarity DVC2 operation DVC2C controls whether DVC2 data value comparisons utilize the normal compare operation, or an alternate “inverted compare” operation. In inverted polarity mode, data value compares perform a not-equal comparison. See details in the DBCR2 register definition.
13 DAC1XMH	Data Address Compare 1 Extended Mask Control High 0 DAC1XM masks 0–15 low-order address bits 1 DAC1XM masks 16–31 low-order address bits DAC1XMH extends the range of the DAC1XM field.
15 DAC2XMH	Data Address Compare 2 Extended Mask Control High 0 DAC2XM masks 0–15 low-order address bits 1 DAC2XM masks 16–31 low-order address bits DAC2XMH extends the range of the DAC2XM field.

Table 1065. DBCR4 Field descriptions (continued)

Field	Description
16:19 DAC1XM	Data Address Compare 1 Extended Mask Control Value of DAC1XMH || DAC1XM: 00000 No additional masking when DBCR2[DAC12M] = 00 00001 Exact Match Bit Mask. One to 31 low-order bits are masked in DAC1 when comparing the storage address with the value in DAC1 for exact address compare (DBCR2[DAC12M] = 00). Address ranges of 2 bytes to 2GB are supported ... 11111 Exact Match Bit Mask. One to 31 low-order bits are masked in DAC1 when comparing the storage address with the value in DAC1 for exact address compare (DBCR2[DAC12M] = 00). Address ranges of 2 bytes to 2GB are supported DAC1XM allows for binary power of 2 address range compares for DAC1 without requiring the use of DAC2.
20:23 DAC2XM	Data Address Compare 2 Extended Mask Control Value of DAC2XMH || DAC2XM: 00000 No additional masking when DBCR2[DAC12M] = 00 00001 Exact Match Bit Mask. One to 31 low-order bits are masked in DAC2 when comparing the storage address with the value in DAC2 for exact address compare (DBCR2[DAC12M] = 00). Address ranges of 2 bytes to 2GB are supported ... 11111 Exact Match Bit Mask. One to 31 low-order bits are masked in DAC2 when comparing the storage address with the value in DAC2 for exact address compare (DBCR2[DAC12M] = 00). Address ranges of 2 bytes to 2GB are supported DAC2XM allows for binary power of 2 address range compares for DAC2.

Table 1065. DBCR4 Field descriptions (continued)

Field	Description
24:27 DAC1CFG	Data Address Compare 1 Configuration 0000 DAC1 debug watchpoints are enabled for load-type or store-type data storage accesses when $DBCR0_{DAC1}$ = 00 0001 DAC1 debug watchpoints are enabled only for store-type data storage accesses when $DBCR0_{DAC1}$ = 00 0010 DAC1 debug watchpoints are enabled only for load-type data storage accesses when $DBCR0_{DAC1}$ = 00 0011 Reserved … 0111 Reserved 1000 DAC1 address comparisons are used for stack limit checking. DAC1 address comparisons are qualified with use of GPR R1 in <EA> calculation for most load or store instructions. No debug events occur for DAC1. When a qualified DAC1 match occurs, a machine check exception is generated for supervisor-mode accesses or a DSI is generated for user-mode accesses, and a DAC1 watchpoint is generated. $DBCR0_{DAC1}$ and $DBCR2_{DVC1M}$ settings are ignored. 1001 Reserved … 1111 Reserved DAC1CFG controls whether DAC1 data address comparisons utilize the normal PowerISA operation for generating both debug events and watchpoints, a watchpoint-only function, or a stack limit checking function (with watchpoint). **Note:** Unlike the DAC3–4CFG fields, debug event enabling for DAC1 is controlled by the DAC1 field in DBCR0. The DAC1CFG encodings 0000–0010 are used to control watchpoint generation when $DBCR0_{DAC1}$ = 0; when $DBCR0_{DAC1}$!= 00, DAC1 watchpoints will fire whenever a DAC1 debug event occurs.
28:31 DAC2CFG	Data Address Compare 2 Configuration 0000 DAC2 debug watchpoints are enabled for load-type or store-type data storage accesses when $DBCR0_{DAC2}$ = 00 0001 DAC2 debug watchpoints are enabled only for store-type data storage accesses when $DBCR0_{DAC2}$ = 00 0010 DAC2 debug watchpoints are enabled only for load-type data storage accesses when $DBCR0_{DAC2}$ = 00 0011 Reserved … 1111 Reserved DAC2CFG controls whether DAC2 data address comparisons utilize the normal compare operation for generating both debug events and watchpoints, or a watchpoint-only function. **Note:** Unlike the DAC3–4CFG fields, debug event enabling for DAC2 is controlled by the DAC2 field in DBCR0. The DAC2CFG encodings 0000–0010 are used to control watchpoint generation when $DBCR0_{DAC2}$ = 00. When $DBCR0_{DAC2}$!= 00, DAC2 watchpoints will fire whenever a DAC2 debug event occurs.

53.3.2.5 Debug Control Register 5 (DBCR5)

Debug Control Register 5 is used to configure Instruction Address Compare operation for IAC5–8. The DBCR5 register is shown in *Figure 1104*

Offset: Refer to SPR - 564 Access: Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	IAC5US		IAC5ER		IAC6US		IAC6ER		IAC56M		0					
W																
Reset(1)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	IAC7US		IAC7ER		IAC8US		IAC8ER		IAC78M		0					
W																
Reset(1)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. Reset by processor reset **p_reset_b** if $EDBCR0_{EDM}$ = 0, as well as unconditionally by **m_por**. If $EDBCR0_{EDM}$ = 1, EDBRAC0 masks off hardware-owned resources from reset by **p_reset_b** and only software-owned resources indicated by EDBRAC0 will be reset by **p_reset_b**.

Figure 1104. Debug Control Register 5 (DBCR5)

Table 1063 provides bit definitions for Debug Control Register 5.

Table 1066. DBCR5 field descriptions

Field	Description
0:1 IAC5US	Instruction Address Compare 5 User/Supervisor Mode 00 IAC5 debug events not affected by MSR_{PR} 01 Reserved 10 IAC5 debug events can only occur if MSR_{PR} = 0 (Supervisor mode) 11 IAC5 debug events can only occur if MSR_{PR} = 1. (User mode)
2:3 IAC5ER	Instruction Address Compare 5 Effective/Real Mode 00 IAC5 debug events are based on effective address 01 Unimplemented (Book E real address compare), no match can occur 10 IAC5 debug events are based on effective address and can only occur if MSR_{IS} = 0 11 IAC5 debug events are based on effective address and can only occur if MSR_{IS} = 1
4:5 IAC6US	Instruction Address Compare 6 User/Supervisor Mode 00 IAC6 debug events not affected by MSR_{PR} 01 Reserved 10 IAC6 debug events can only occur if MSR_{PR} = 0 (Supervisor mode) 11 IAC6 debug events can only occur if MSR_{PR} = 1. (User mode)
6:7 IAC6ER	Instruction Address Compare 6 Effective/Real Mode 00 IAC6 debug events are based on effective address 01 Unimplemented (Book E real address compare), no match can occur 10 IAC6 debug events are based on effective address and can only occur if MSR_{IS} = 0 11 IAC6 debug events are based on effective address and can only occur if MSR_{IS} = 1

Table 1066. DBCR5 field descriptions (continued)

Field	Description
8:9 IAC56M	Instruction Address Compare 5/6 Mode 00 Exact address compare. IAC5 debug events can only occur if the address of the instruction fetch is equal to the value specified in IAC5. IAC6 debug events can only occur if the address of the instruction fetch is equal to the value specified in IAC6. 01 Address bit match. IAC5 debug events can occur only if the address of the instruction fetch, ANDed with the contents of IAC6 are equal to the contents of IAC5, also ANDed with the contents of IAC6. IAC6 debug events do not occur. IAC5US and IAC5ER settings are used. 10 Inclusive address range compare. IAC5 debug events can occur only if the address of the instruction fetch is greater than or equal to the value specified in IAC5 and less than the value specified in IAC6. IAC6 debug events do not occur. IAC5US and IAC5ER settings are used. 11 Exclusive address range compare. IAC5 debug events can occur only if the address of the instruction fetch is less than the value specified in IAC5 or is greater than or equal to the value specified in IAC6. IAC6 debug events do not occur. IAC5US and IAC5ER settings are used.
15:17 IAC7US	Instruction Address Compare 7 User/Supervisor Mode 00 IAC7 debug events not affected by MSR_{PR} 01 Reserved 10 IAC7 debug events can only occur if MSR_{PR} = 0 (Supervisor mode) 11 IAC7 debug events can only occur if MSR_{PR} = 1 (User mode)
18:19 IAC7ER	Instruction Address Compare 7 Effective/Real Mode 00 IAC7 debug events are based on effective address 01 Unimplemented (Book E real address compare), no match can occur 10 IAC7 debug events are based on effective address and can only occur if MSR_{IS} = 0 11 IAC7 debug events are based on effective address and can only occur if MSR_{IS} = 1
20:21 IAC8US	Instruction Address Compare 8 User/Supervisor Mode 00 IAC8 debug events not affected by MSR_{PR} 01 Reserved 10 IAC8 debug events can only occur if MSR_{PR} = 0 (Supervisor mode). 11 IAC8 debug events can only occur if MSR_{PR} = 1. (User mode)

Table 1066. DBCR5 field descriptions (continued)

Field	Description
22:23 IAC8ER	Instruction Address Compare 8 Effective/Real Mode 00 IAC8 debug events are based on effective address 01 Unimplemented (Book E real address compare), no match can occur 10 IAC8 debug events are based on effective address and can only occur if MSR_{IS} = 0 11 IAC8 debug events are based on effective address and can only occur if MSR_{IS} = 1
24:25 IAC78M	Instruction Address Compare 7/8 Mode 00 Exact address compare. IAC7 debug events can only occur if the address of the instruction fetch is equal to the value specified in IAC7. IAC8 debug events can only occur if the address of the instruction fetch is equal to the value specified in IAC8. 01 Address bit match. IAC7 debug events can occur only if the address of the instruction fetch, ANDed with the contents of IAC8 are equal to the contents of IAC7, also ANDed with the contents of IAC8. IAC8 debug events do not occur. IAC7US and IAC7ER settings are used. 10 Inclusive address range compare. IAC7 debug events can occur only if the address of the instruction fetch is greater than or equal to the value specified in IAC7 and less than the value specified in IAC8. IAC8 debug events do not occur. IAC7US and IAC7ER settings are used. 11 Exclusive address range compare. IAC7 debug events can occur only if the address of the instruction fetch is less than the value specified in IAC7 or is greater than or equal to the value specified in IAC8. IAC8 debug events do not occur. IAC7US and IAC7ER settings are used.

53.3.2.6 Debug Control Register 6 (DBCR6)

Debug Control Register 6 is used to extend instruction address compare matching functionality. DBCR6 is shown in *Figure 1105*.

Offset: Refer to SPR - 603 Access: Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	IAC1XM				IAC2XM				IAC3XM				IAC4XM			
Reset(1)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	IAC5XM				IAC6XM				IAC7XM				IAC8XM			
Reset(1)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. DBCR6 is reset by processor reset **p_reset_b** if $EDBCR0_{EDM}$ = 0, as well as unconditionally by **m_por**. If $EDBCR0_{EDM}$ = 1, EDBRAC0 masks off hardware-owned resources from reset by **p_reset_b** and only software-owned resources indicated by EDBRAC0 will be reset by **p_reset_b**.

Figure 1105. Debug Control Register 6 (DBCR6)

Table 1067 provides bit definitions for Debug Control Register 6.

Table 1067. DBCR6 field descriptions

Field	Description
0:3 IAC1XM	Instruction Address Compare 1 Extended Mask Control 0000 No additional masking when DBCR1[IAC12M] = 00 0001 Exact Match Bit Mask. Number of low order bits masked in IAC1 when comparing the storage address with the value in IAC1 for exact address compare (DBCR1[IAC12M] = 00). Ranges up to 4KB are supported … 1100 Exact Match Bit Mask. Number of low order bits masked in IAC1 when comparing the storage address with the value in IAC1 for exact address compare (DBCR1[IAC12M] = 00). Ranges up to 4KB are supported 1101 Reserved 1110 Reserved 1111 Reserved IAC1XM allows for binary power of 2 address range compares for IAC1 without requiring the use of IAC2.
4:7 IAC2XM	Instruction Address Compare 2 Extended Mask Control 0000 No additional masking when DBCR1[IAC12M] = 00 0001 Exact Match Bit Mask. Number of low order bits masked in IAC2 when comparing the storage address with the value in IAC2 for exact address compare (DBCR1[IAC12M] = 00). Ranges up to 4KB are supported … 1100 Exact Match Bit Mask. Number of low order bits masked in IAC2 when comparing the storage address with the value in IAC2 for exact address compare (DBCR1[IAC12M] = 00). Ranges up to 4KB are supported 1101 Reserved 1110 Reserved 1111 Reserved IAC2XM allows for binary power of 2 address range compares for IAC2 without requiring the use of IAC1.
8:11 IAC3XM	Instruction Address Compare 3 Extended Mask Control 0000 No additional masking when DBCR1[IAC34M] = 00 0001 Exact Match Bit Mask. Number of low order bits masked in IAC3 when comparing the storage address with the value in IAC3 for exact address compare (DBCR1[IAC34M] = 00). Ranges up to 4KB are supported … 1100 Exact Match Bit Mask. Number of low order bits masked in IAC3 when comparing the storage address with the value in IAC3 for exact address compare (DBCR1[IAC34M] = 00). Ranges up to 4KB are supported 1101 Reserved 1110 Reserved 1111 Reserved IAC3XM allows for binary power of 2 address range compares for IAC1 without requiring the use of IAC2.

Table 1067. DBCR6 field descriptions (continued)

Field	Description
12:15 IAC4XM	Instruction Address Compare 4 Extended Mask Control 0000 No additional masking when DBCR1[IAC34M] = 00 0001 Exact Match Bit Mask. Number of low order bits masked in IAC4 when comparing the storage address with the value in IAC4 for exact address compare (DBCR1[IAC34M] = 00). Ranges up to 4KB are supported. … 1100 Exact Match Bit Mask. Number of low order bits masked in IAC4 when comparing the storage address with the value in IAC4 for exact address compare (DBCR1[IAC34M] = 00). Ranges up to 4KB are supported 1101 Reserved 1110 Reserved 1111 Reserved IAC4XM allows for binary power of 2 address range compares for IAC4 without requiring the use of IAC3.
16:19 IAC5XM	Instruction Address Compare 5 Extended Mask Control 0000 No additional masking when DBCR5[IAC56M] = 00 0001 Exact Match Bit Mask. Number of low order bits masked in IAC5 when comparing the storage address with the value in IAC5 for exact address compare (DBCR5[IAC56M] = 00). Ranges up to 4KB are supported. … 1100 Exact Match Bit Mask. Number of low order bits masked in IAC5 when comparing the storage address with the value in IAC5 for exact address compare (DBCR5[IAC56M] = 00). Ranges up to 4KB are supported 1101 Reserved 1110 Reserved 1111 Reserved IAC5XM allows for binary power of 2 address range compares for IAC5 without requiring the use of IAC6.
20:23 IAC6XM	Instruction Address Compare 6 Extended Mask Control 0000 No additional masking when DBCR5[IAC56M] = 00 0001 Exact Match Bit Mask. Number of low order bits masked in IAC6 when comparing the storage address with the value in IAC6 for exact address compare (DBCR5[IAC56M] = 00). Ranges up to 4KB are supported … 1100 Exact Match Bit Mask. Number of low order bits masked in IAC6 when comparing the storage address with the value in IAC6 for exact address compare (DBCR5[IAC56M] = 00). Ranges up to 4KB are supported 1101 Reserved 1110 Reserved 1111 Reserved IAC6XM allows for binary power of 2 address range compares for IAC6 without requiring the use of IAC5.

Table 1067. DBCR6 field descriptions (continued)

Field	Description
24:27 24:27 IAC7XM	Instruction Address Compare 7 Extended Mask Control 0000 No additional masking when DBCR5[IAC78M] = 00 0001 Exact Match Bit Mask. Number of low order bits masked in IAC7 when comparing the storage address with the value in IAC7 for exact address compare (DBCR5[IAC78M] = 00). Ranges up to 4KB are supported … 1100 Exact Match Bit Mask. Number of low order bits masked in IAC7 when comparing the storage address with the value in IAC7 for exact address compare (DBCR5[IAC78M] = 00). Ranges up to 4KB are supported 1101 Reserved 1110 Reserved 1111 Reserved IAC7XM allows for binary power of 2 address range compares for IAC7 without requiring the use of IAC8.
28:31 IAC8XM	Instruction Address Compare 8 Extended Mask Control 0000 No additional masking when DBCR5[IAC78M] = 00 0001 Exact Match Bit Mask. Number of low order bits masked in IAC8 when comparing the storage address with the value in IAC8 for exact address compare (DBCR5[IAC78M] = 00). Ranges up to 4KB are supported … 1100 Exact Match Bit Mask. Number of low order bits masked in IAC8 when comparing the storage address with the value in IAC8 for exact address compare (DBCR5[IAC78M] = 00). Ranges up to 4KB are supported 1101 Reserved 1110 Reserved 1111 Reserved IAC8XM allows for binary power of 2 address range compares for IAC8 without requiring the use of IAC7.

53.3.2.7 Debug Control Register 7 (DBCR7)

Debug Control Register 7 is used to enable and configure Data Address Compare 3 and 4 functionality. DBCR7 is shown in *Figure 1106*.

Offset: Refer to SPR - 596 Access: Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0												0	DAC3XMH	0	DAC4XMH
W																
Reset(1)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	DAC3XM				DAC4XM				DAC3CFG				DAC4CFG			
W																
Reset(1)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. DBCR7 is reset by processor reset **p_reset_b** if $EDBCR0_{EDM}$ = 0, as well as unconditionally by **m_por**. If $EDBCR0_{EDM}$ = 1, EDBRAC0 masks off hardware-owned resources from reset by **p_reset_b** and only software-owned resources indicated by EDBRAC0 will be reset by **p_reset_b**.

Figure 1106. Debug Control Register 7 (DBCR7)

Table 1068 provides bit definitions for Debug Control Register 7.

Table 1068. DBCR7 Bit Definitions

Field	Description
13 DAC3XMH	Data Address Compare 3 Extended Mask Control High 0 DAC3XM masks 0–15 low-order address bits 1 DAC3XM masks 16–31 low-order address bits DAC3XMH extends the range of the DAC3XM field.
15 DAC4XMH	Data Address Compare 4 Extended Mask Control High 0 DAC4XM masks 0–15 low-order address bits 1 DAC4XM masks 16–31 low-order address bits DAC4XMH extends the range of the DAC4XM field.
16:19 DAC3XM	Data Address Compare 3 Extended Mask Control Value of DAC3XMH || DAC3XM: 00000 No additional masking when DBCR8[DAC34M] = 00 00001 Exact Match Bit Mask. One to 31 low-order bits are masked in DAC3 when comparing the storage address with the value in DAC3 for exact address compare (DBCR8[DAC34M] = 00). Address ranges of 2 bytes to 2GB are supported ... 11111 Exact Match Bit Mask. One to 31 low-order bits are masked in DAC3 when comparing the storage address with the value in DAC3 for exact address compare (DBCR8[DAC34M] = 00). Address ranges of 2 bytes to 2GB are supported DAC3XM allows for binary power of 2 address range compares for DAC3 without requiring the use of DAC4.

Table 1068. DBCR7 Bit Definitions (continued)

Field	Description
20:13 DAC4XM	Data Address Compare 4 Extended Mask Control Value of DAC4XMH || DAC4XM: 00000 No additional masking when DBCR8[DAC34M] = 00 00001 Exact Match Bit Mask. One to 31 low-order bits are masked in DAC4 when comparing the storage address with the value in DAC4 for exact address compare (DBCR8[DAC34M] = 00). Address ranges of 2 bytes to 2GB are supported ... 11111 Exact Match Bit Mask. One to 31 low-order bits are masked in DAC4 when comparing the storage address with the value in DAC4 for exact address compare (DBCR8[DAC34M] = 00). Address ranges of 2 bytes to 2GB are supported DAC4XM allows for binary power of 2 address range compares for DAC4 without requiring the use of DAC3.

Table 1068. DBCR7 Bit Definitions (continued)

Field	Description
24:27 DAC3CFG	Data Address Compare 3 Configuration 0000 DAC3 debug watchpoints are enabled for load-type or store-type data storage accesses 0001 DAC3 debug watchpoints are enabled only for store-type data storage accesses 0010 DAC3 debug watchpoints are enabled only for load-type data storage accesses 0011 Reserved 0100 Reserved 0101 DAC3 debug events and watchpoints are enabled only for store-type data storage accesses 0110 DAC3 debug events and watchpoints are enabled only for load-type data storage accesses 0111 DAC3 debug events and watchpoints are enabled for load-type or store-type data storage accesses 1000 DAC3 address comparisons are used for stack limit checking. DAC3 address comparisons are qualified with use of GPR R1 in <EA> calculation for most load or store instructions. No debug events occur for DAC3. When a qualified match occurs, a machine check exception is generated for supervisor-mode accesses or a DSI is generated for user-mode accesses, and a DAC3 watchpoints is generated 1001 Reserved ... 1111 Reserved DAC3CFG controls whether DAC3 data address comparisons utilize the normal compare operation for generating both debug events and watchpoints, a watchpoint-only function, or a stack limit checking function (with watchpoint).
28:31 DAC4CFG	Data Address Compare 4 Configuration 0000 DAC4 debug watchpoints are enabled for load-type or store-type data storage accesses 0001 DAC4 debug watchpoints are enabled only for store-type data storage accesses 0010 DAC4 debug watchpoints are enabled only for load-type data storage accesses 0011 Reserved 0100 Reserved 0101 DAC4 debug events and watchpoints are enabled only for store-type data storage accesses 0110 DAC4 debug events and watchpoints are enabled only for load-type data storage accesses 0111 DAC4 debug events and watchpoints are enabled for load-type or store-type data storage accesses 1000 Reserved ... 1111 Reserved DAC4CFG controls whether DAC4 data address comparisons utilize the normal compare operation for generating both debug events and watchpoints, or a watchpoint-only function.

53.3.2.8 Debug Control Register 8 (DBCR8)

Debug Control Register 8 is used to configure Data Address Compare 3 and 4 operation. The DBCR8 register is shown in *Figure 1107*.

Offset: Refer to SPR - 597 Access: Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DAC3US		DAC3ER		DAC4US		DAC4ER		DAC34M		DAC3LNK	DAC4LNK	0			
W																
Reset(1)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0															
W																
Reset(1)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. Reset by processor reset **p_reset_b** if $EDBCR0_{EDM}$ = 0, as well as unconditionally by **m_por**. If $EDBCR0_{EDM}$ = 1, EDBRAC0 masks off hardware-owned resources from reset by **p_reset_b** and only software-owned resources indicated by EDBRAC0 will be reset by **p_reset_b**.

Figure 1107. Debug Control Register 8 (DBCR8)

Table 1069 provides bit definitions for Debug Control Register 8.

Table 1069. DBCR8 Field descriptions

Field	Description
0:1 DAC3US	Data Address Compare 3 User/Supervisor Mode 00 DAC3 debug events not affected by MSR_{PR} 01 Reserved 10 DAC3 debug events can only occur if MSR_{PR} = 0 (Supervisor mode) 11 DAC3 debug events can only occur if MSR_{PR} = 1. (User mode)
2:3 DAC3ER	Data Address Compare 3 Effective/Real Mode 00 DAC3 debug events are based on effective address 01 Unimplemented (Book E real address compare), no match can occur 10 DAC3 debug events are based on effective address and can only occur if MSR_{DS} = 0 11 DAC3 debug events are based on effective address and can only occur if MSR_{DS} = 1
:4:5DAC4US	Data Address Compare 4 User/Supervisor Mode 00 DAC4 debug events not affected by MSR_{PR} 01 Reserved 10 DAC4 debug events can only occur if MSR_{PR} = 0 (Supervisor mode) 11 DAC4 debug events can only occur if MSR_{PR} = 1 (User mode)
6:7 DAC4ER	Data Address Compare 4 Effective/Real Mode 00 DAC4 debug events are based on effective address 01 Unimplemented (Book E real address compare), no match can occur 10 DAC4 debug events are based on effective address and can only occur if MSR_{DS} = 0 11 DAC4 debug events are based on effective address and can only occur if MSR_{DS} = 1

Table 1069. DBCR8 Field descriptions (continued)

Field	Description
8:9 DAC34M	Data Address Compare 3/4 Mode 00 Exact address compare. DAC3 debug events can only occur if the address of the data access is equal to the value specified in DAC3. DAC4 debug events can only occur if the address of the data access is equal to the value specified in DAC4 01 Address bit match. DAC3 debug events can occur only if the address of the data access ANDed with the contents of DAC4, are equal to the contents of DAC3 also ANDed with the contents of DAC4. DAC4 debug events do not occur. DAC3US and DAC3ER settings are used 10 Inclusive address range compare. DAC3 debug events can occur only if the address of the data access is greater than or equal to the value specified in DAC3 and less than the value specified in DAC4. DAC4 debug events do not occur. DAC3US and DAC3ER settings are used 11 Exclusive address range compare. DAC3 debug events can occur only if the address of the data access is less than the value specified in DAC3 or is greater than or equal to the value specified in DAC4. DAC4 debug events do not occur. DAC3US and DAC3ER settings are used
10 DAC3LNK	Data Address Compare 3 Linked 0 No effect 1 DAC3 debug events are linked to IAC5 debug events. IAC5 debug events do not affect DBSR When linked to IAC5, DAC3 debug events are conditioned based on whether the instruction also generated an IAC5 debug event. Note that linking is only available in EDM or IDM.
11 DAC4LNK	Data Address Compare 4 Linked 0 No effect 1 DAC4 debug events are linked to IAC7 debug events. IAC7 debug events do not affect DBSR When linked to IAC7, DAC4 debug events are conditioned based on whether the instruction also generated an IAC7 debug event. Note that linking is only available in EDM or IDM.

53.3.2.9 Debug Status Register (DBSR)

The Debug Status Register (DBSR) contains status on debug events and the most recent processor reset. The Debug Status Register is set via hardware, and read and cleared via software. Bits in the Debug Status Register can be cleared using **mtspr** *DBSR,RS*. Clearing is done by writing to the Debug Status Register with a 1 in any bit position that is to be cleared and 0 in all other bit positions. The write data to the Debug Status Register is not direct data, but a mask. A '1' causes the bit to be cleared, and a '0' has no effect. Debug Status bits are set by Debug events only while Internal Debug Mode is enabled ($DBCR0_{IDM}$ = 1). When debug interrupts are enabled (MSR_{DE} = 1 $DBCR0_{IDM}$ = 1 and $EDBCR0_{EDM}$ = 0, or MSR_{DE} = 1, $DBCR0_{IDM}$ = 1, $EDBCR0_{EDM}$ = 1 and software is allocated resource(s) via EDBRAC0), a set bit in DBSR other than MRR, DAC_OFST, or VLES will cause a debug interrupt to be generated. The debug interrupt handler is responsible for clearing DBSR bits prior to returning to normal execution. When resource sharing is enabled, ($EDBCR0_{EDM}$ = 1 and $EDBRAC0_{IDM}$ = 1), only software-owned resources may be modified by software, and status bits associated with hardware-owned resources will not be set by hardware in DBSR. The DBSR register is shown in *Figure 1108*.

Offset: Refer to SPR - 304 Access: Read/Clear

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	IDE	UDE	MRR		ICMP	BRT	IRPT	TRAP	IAC	0			DACR	DACW	0	IDE
W																
Reset	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	RET	0				DEVT1	DEVT2	PMI	MPU	CIRPT	CRET	VLES	DAC_OFST			0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1108. Debug Status Register (DBSR)

Table 1070 provides bit definitions for the Debug Status Register.

Table 1070. DBSR field descriptions

Field	Description
0 IDE	Imprecise Debug Event Set to 1 if MSR_{DE} = 0 and $DBCR0_{IDM}$ = 1 and a debug event causes its respective Debug Status Register bit to be set to 1. It may also be set to '1' if an imprecise debug event occurs due to a DAC event on a load or store which is terminated with error, or if an ICMP event occurs in conjunction with a EFPU FP round exception.
1 UDE	Unconditional Debug Event Set to 1 if an Unconditional debug event occurred.
2 MRR	Most Recent Reset 00 No reset occurred since these bits were last cleared by software 01 A hard reset occurred since these bits were last cleared by software 10 Reserved 11 Reserved
3 ICMP	Instruction Complete Debug Event Set to 1 if an Instruction Complete debug event occurred.
4 BRT	Branch Taken Debug Event Set to 1 if an Branch Taken debug event occurred.
6 IRPT	Interrupt Taken Debug Event Set to 1 if an Interrupt Taken debug event occurred.
7 TRAP	Trap Taken Debug Event Set to 1 if a Trap Taken debug event occurred.
8 IAC	Instruction Address Compare Debug Event Set to 1 if an IAC debug event occurred.
12 DACR	Data Address Compare Read Debug Event Set to 1 if a read-type DAC debug event occurred

Table 1070. DBSR field descriptions (continued)

Field	Description
13 DACW	Data Address Compare Write Debug Event Set to 1 if a write-type DAC debug event occurred
15 DNI	DNI Debug Event Set to 1 if a DNI debug event occurred
16 RET	Return Debug Event Set to 1 if a Return debug event occurred
21 21 DEVT1	External Debug Event 1 Debug Event Set to 1 if a DEVT1 debug event occurred
22 DEVT2	External Debug Event 2 Debug Event Set to 1 if a DEVT2 debug event occurred
23 PMI	Performance Monitor Interrupt Debug Event Set to 1 if a Performance Monitor Interrupt event occurred with $PMGC0_{UDI}$ = 1
24 MPU	Memory Protection Unit Debug Event Set to 1 if a MPU debug event occurred. For MPU debug events, the IAC, DACR, or DACW status bit will also be set, depending on the type of access which matched a region descriptor enabled to generate debug events, in order to further define the type of MPU debug event. Note that software MPU debug events on data accesses normally occur after the data access is performed and the instruction completes, thus act like a normal DAC debug event. The instruction may not complete if a DSI occurs. In this case, IDE will be set to indicate this occurrence.
25 CIRPT	Critical Interrupt Taken Debug Event Set to 1 if a Critical Interrupt Taken debug event occurred.
26 CRET	Critical Return Debug Event Set to 1 if a Critical Return debug event occurred
27 VLES	VLE Status Set to 1 if an ICMP, BRT, TRAP, RET, CRET, IAC, or DAC debug event occurred on a PowerISA VLE Instruction. Undefined for IRPT, CIRPT, DEVT[1,2], and UDE events
28:31 DAC_OFST	Data Address Compare Offset Indicates offset-1 of saved DSRR0 value from the address of the load or store instruction which took a DAC Debug exception, unless a simultaneous DSI error occurs, in which case this field is set to 3‘b000 and $DBSR_{IDE}$ is set to 1. Normally set to 3‘b000 by a non-DVC DAC. A DVC DAC may set this field to any value.

53.3.2.10 Debug Data Effective Address Register (DDEAR)

The Debug Data Effective Address Register (DDEAR) contains address information for data address compare debug events (DAC or MPU DAC). DDEAR is update by hardware with the effective address of the load, store, or cache control operation when a data address compare event is recorded in DBSR if the previous values of the $DBSR_{DAC\{R,W\}}$ bits are zero. Once a DDEAR update is performed, these bits must be cleared by software prior to another DDEAR hardware update occurring. A subsequent DAC or MPU DAC event will not update the DDEAR register if either of the $DBSR_{DAC\{R,W\}}$ bits are set, in order to capture the first event address.

The DDEAR register is shown in *Figure 1109*.

Offset: Refer to SPR - 600 Access: Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Data Effective Address[0:15]															
W																
Reset	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	Data Effective Address[16:31]															
W																
Reset	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—

Figure 1109. Debug Data Effective Address Register (DDEAR)

53.3.3 External Debug Resource Allocation Control Register (EDBRAC0)

The External Debug Resource Allocation Control Register (EDBRAC0) controls resource allocation when $EDBCR0_{EDM}$ is set to '1'. EDBRAC0 provides a mechanism for the hardware debugger to share certain debug resources with software. Individual resources are allocated based on the settings of EDBRAC0 when $EDBCR0_{EDM}$ = 1. EDBRAC0 settings are ignored when $EDBCR0_{EDM}$ = 0.

Hardware-owned resources which generate debug events update EDBSR0 instead of DBSR and cause entry into debug mode if the event is not masked in EDBSRMSK0, while software-owned resources which generate debug events if $DBCR0_{IDM}$ = 1 update DBSR, causing debug interrupts to occur if MSR_{DE} = 1. EDBRAC0 is controlled via the OnCE port hardware, and is read-only to software.

The DBSR status register is always owned by software. Debug status bits in DBSR are set by software-owned debug events only while Internal Debug Mode is enabled. When debug interrupts are enabled (MSR_{DE} = 1 $DBCR0_{IDM}$ = 1 and $EDBCR0_{EDM}$ = 0, or MSR_{DE} = 1, $DBCR0_{IDM}$ = 1 and $EDBCR0_{EDM}$ = 1 and software is allocated resource(s) via EDBRAC0), a set bit in DBSR by an event which is software-owned (other than MRR, DAC_OFST, or VLES) will cause a debug interrupt to be generated.

Debug status bits in EDBSR0 are set by hardware-owned debug events only while External Debug Mode is enabled ($EDBCR0_{EDM}$ = 1). When $EDBCR0_{EDM}$ = 1, a set bit in EDBSR0 by an event which is hardware-owned (other than IDE, DAC_OFST, or VLES) will cause entry into debug mode unless entry is masked via EDBSRMSK0.

If $EDBCR0_{EDM}$ = 1, DBSR status bits corresponding to hardware-owned debug events are masked from being set by hardware.

Software-owned resources may be modified by software, but only the corresponding control bits in DBCR0–8 or MPU0 CSR0 are affected by execution of a **mtspr**, thus only a portion of these registers may be affected, depending on the allocation settings in EDBRAC0. The debug interrupt handler is still responsible for clearing DBSR bits for software-owned resources prior to returning to normal execution. Hardware always has full access to all registers and register fields through the OnCE register access mechanism, and it is up to the debug firmware to properly implement modifications to these registers with read-modify-write operations to implement any control sharing with software. Settings in EDBRAC0 should be considered by the debug firmware in order to preserve software settings of control

and status registers as appropriate when hardware modifications to the debug registers is performed.

The EDBRAC0 register is shown in *Figure 1110*.

Offset: Refer to SPR - 638 Access: Read-only by Software

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	IDM	RST	UDE	ICMP	BRT	IRPT	TRAP	IAC1	IAC2	IAC3	IAC4	DAC1	DAC34	DAC2	0
W																
Reset(1)	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	RET	IAC5	IAC6	IAC7	IAC8	DEVT1	DEVT2	PMI	MPU	CIRPT	CRET	DNI	DQM	0		
W																
Reset(1)	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—

1. Unaffected by **p_reset_b**, reset to 0x00000180 by **m_por** or while in the test-logic-reset OnCE controller state

Figure 1110. External Debug Resource Allocation Control Register (EDBRAC0)

Table 1070 provides bit definitions for the Debug External Resource Control Register. Note that EDBRAC0 controls are disabled when EDBCR0EDM = 0.

Table 1071. EDBRAC0 register field descriptions

Field	Description
1 IDM	Internal Debug Mode control 0 Internal Debug mode may not be enabled by software. $DBCR0_{IDM}$ is owned exclusively by hardware. **mtspr** DBCR0–8 and other debug registers is always ignored. No resource sharing occurs, regardless of the settings of other fields in EDBRAC0. Hardware exclusively owns all resources 1 Internal Debug mode may be enabled by software. $DBCR0_{IDM}$ is owned by software. $DBCR0_{IDM}$ is software readable/writable When $EDBRAC0_{IDM}$ = 1, software writes to hardware-owned bits in DBCR0–8 via **mtspr** are ignored.
2 RST	Reset Field Control 0 $DBCR0_{RST}$ owned exclusively by hardware debug. No **mtspr** access by software to $DBCR0_{RST}$ field 1 $DBCR0_{RST}$ accessible by software debug. $DBCR0_{RST}$ is software readable/writable
3 UDE	Unconditional Debug Event 0 Event owned by hardware debug 1 Event owned by software debug
4 ICMP	Instruction Complete Debug Event 0 Event owned by hardware debug. No **mtspr** access by software to $DBCR0_{ICMP}$ field 1 Event owned by software debug. $DBCR0_{ICMP}$ is software readable/writable

Table 1071. EDBRAC0 register field descriptions (continued)

Field	Description
5 BRT	Branch Taken Debug Event 0 Event owned by hardware debug. No **mtspr** access by software to $DBCR0_{BRT}$ field 1 Event owned by software debug. $DBCR0_{BRT}$ is software readable/writable
6 IRPT	Interrupt Taken Debug Event 0 Event owned by hardware debug. No **mtspr** access by software to $DBCR0_{IRPT}$ field 1 Event owned by software debug. $DBCR0_{IRPT}$ is software readable/writable
7 TRAP	Trap Taken Debug Event 0 Event owned by hardware debug. No **mtspr** access by software to $DBCR0_{TRAP}$ field 1 Event owned by software debug. $DBCR0_{TRAP}$ is software readable/writable
8 IAC1	Instruction Address Compare 1 Debug Event 0 Event owned by hardware debug. No **mtspr** access by software to IAC1 control and status fields 1 Event owned by software debug. IAC1 control fields are software readable/writable
6 IAC2	Instruction Address Compare 2 Debug Event 0 Event owned by hardware debug. No **mtspr** access by software to IAC2 control and status fields 1 Event owned by software debug. IAC2 control fields are software readable/writable
10 IAC3	Instruction Address Compare 3 Debug Event 0 Event owned by hardware debug. No **mtspr** access by software to IAC3 control and status fields 1 Event owned by software debug. IAC3 control fields are software readable/writable
11 IAC4	Instruction Address Compare 4 Debug Event 0 Event owned by hardware debug. No **mtspr** access by software to IAC4 control and status fields 1 Event owned by software debug. IAC4 control fields are software readable/writable
12 DAC1	Data Address Compare 1 Debug Event 0 Event owned by hardware debug. No **mtspr** access by software to DAC1 control and status fields 1 Event owned by software debug. DAC1 control fields are software readable/writable
13 DAC34	Data Address Compare 3 and 4 Debug Events 0 Events owned by hardware debug. No **mtspr** access by software to DAC3 and DAC4 control and status fields 1 Event owned by software debug. DAC3 and DAC4 control fields are software readable/writable
14 DAC2	Data Address Compare 2 Debug Event 0 Event owned by hardware debug. No **mtspr** access by software to DAC2 control and status fields 1 Event owned by software debug. DAC2 control fields are software readable/writable
16 RET	Return Debug Event 0 Event owned by hardware debug. No **mtspr** access by software to $DBCR0_{RET}$ field 1 Event owned by software debug. $DBCR0_{RET}$ is software readable/writable
17 IAC5	Instruction Address Compare 5 Debug Event 0 Event owned by hardware debug. No **mtspr** access by software to IAC5 control and status fields 1 Event owned by software debug. IAC5 control fields are software readable/writable

Table 1071. EDBRAC0 register field descriptions (continued)

Field	Description
18 IAC6	Instruction Address Compare 6 Debug Event 0 Event owned by hardware debug. No **mtspr** access by software to IAC6 control and status fields 1 Event owned by software debug. IAC6 control fields are software readable/writable
19 IAC7	Instruction Address Compare 7 Debug Event 0 Event owned by hardware debug. No **mtspr** access by software to IAC7 control and status fields 1 Event owned by software debug. IAC7 control fields are software readable/writable
20 IAC8	Instruction Address Compare 8 Debug Event 0 Event owned by hardware debug. No **mtspr** access by software to IAC8 control and status fields 1 Event owned by software debug. IAC8 control are software readable/writable
21 DEVT1	External Debug Event Input 1 Debug Event 0 Event owned by hardware debug. No **mtspr** access by software to $DBCR0_{DEVT1}$ field 1 Event owned by software debug. $DBCR0_{DEVT1}$ is software readable/writable
22 DEVT2	External Debug Event Input 2 Debug Event 0 Event owned by hardware debug. No **mtspr** access by software to $DBCR0_{DEVT2}$ field 1 Event owned by software debug. $DBCR0_{DEVT2}$ is software readable/writable
23 PMI	Performance Monitor Interrupt Debug Event 0 Event owned by hardware debug. No **mtpmr** access by software to the PMRs. Performance monitor interrupts set $EDBSR0_{PMI}$ regardless of the setting of $PMGC0_{UDI}$ 1 Event owned by software debug. PMRs are software readable/writable **Note:** This bit is reset to ‘1’.
24 MPU	Memory Protection Unit Debug Event 0 Event owned by hardware debug. No **mpuwe** access by software to region descriptors which have the DEBUG control bit set to ‘1’ or to the $MPU0CSR0_{DRDEN,DWDEN,IDEN}$ control bits, unless the CPU is in a debug session (**jd_debug_b** is asserted). MPU debug events set $EDBSR0_{MPU}$, and if not masked by $EDBSRMSK0_{MPU}$, one of $EDBSR0_{IAC}$, $EDBSR0_{DACR}$, or $EDBSR0_{DACW}$. MPU flash invalidates do not affect DEBUG = 1 entries unless the CPU is in a debug session (**jd_debug_b** is asserted) 1 Event owned by software debug. All region descriptors and the $MPU0CSR0_{DRDEN,DWDEN,IDEN}$ control bits are software readable/writable **Note:** This bit is reset to ‘1’.
25 CIRPT	Critical Interrupt Taken Debug Event 0 Event owned by hardware debug. No **mtspr** access by software to $DBCR0_{CIRPT}$ field 1 Event owned by software debug. $DBCR0_{CIRPT}$ is software readable/writable
26 CRET	Critical Return Debug Event 0 Event owned by hardware debug. No **mtspr** access by software to $DBCR0_{CRET}$ field 1 Event owned by software debug. $DBCR0_{CRET}$ is software readable/writable

Table 1071. EDBRAC0 register field descriptions (continued)

Field	Description
27 DNI	DNI Instruction Debug Control 0 DNI resource owned by hardware debug. Execution of an **e_dni** or **se_dni** instruction results in entry into debug mode 1 DNI resource owned by software debug. Execution of an **e_dni** or **se_dni** instruction results in either a debug interrupt ($DBCR0_{IDM}$ = 1 and MSR_{DE} = 1) or a nop ($DBCR0_{IDM}$ = 0 or MSR_{DE} = 0) **Note:** DNI events are not blocked during a debug session
28 DQM	Data Acquisition Messaging Registers 0 $DEVENT_{DQTAG}$ and DDAM register are exclusively owned by hardware debug. No **mtspr** access by software to $DEVENT_{DQTAG}$ field or DDAM register. Attempted access by software is ignored 1 $DEVENT_{DQTAG}$ and DDAM register are owned by software. Software has read/write access to $DEVENT_{DQTAG}$ field and DDAM register

Table 1072 shows which resources are controlled by EDBRAC0 settings.

Table 1072. EDBRAC0 Resource Control

$EDBCR0_{EDM}$	$EDBRAC0_{IDM}$	$EDBRAC0_{RST}$	$EDBRAC0_{UDE}$	$EDBRAC0_{ICMP}$	$EDBRAC0_{BRT}$	$EDBRAC0_{IRPT}$	$EDBRAC0_{TRAP}$	$EDBRAC0_{IAC1}$	$EDBRAC0_{IAC2}$	$EDBRAC0_{IAC3}$	$EDBRAC0_{IAC4}$	$EDBRAC0_{IAC5}$	$EDBRAC0_{IAC6}$	$EDBRAC0_{IAC7}$	$EDBRAC0_{IAC8}$	$EDBRAC0_{DAC1}$	$EDBRAC0_{DAC34}$	$EDBRAC0_{DAC2}$	$EDBRAC0_{RET}$	$EDBRAC0_{DEVT1}$	$EDBRAC0_{DEVT2}$	$EDBRAC0_{PMI}$	$EDBRAC0_{MPU}$	$EDBRAC0_{CIRT}$	$EDBRAC0_{CRET}$	$EDBRAC0_{DNI}$	$EDBRAC0_{DQM}$	Software Accessible via mtspr, affected by p_reset_b
0	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	All debug registers
1	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	$DBCR0_{IDM}$
1	1	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	$DBCR0_{RST}$
1	1	-	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	$DBCR0_{UDE}$
1	1	-	-	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	$DBCR0_{ICMP}$
1	1	-	-	-	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	$DBCR0_{BRT}$
1	1	-	-	-	-	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	$DBCR0_{IRPT}$
1	1	-	-	-	-	-	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	$DBCR0_{TRAP}$
1	1	-	-	-	-	-	-	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	IAC1, $DBCR0_{IAC1}$, $DBCR1_{IAC1US}$, IAC1ER, $DBCR6_{IAC1XM}$
1	1	-	-	-	-	-	-	-	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	IAC2, $DBCR0_{IAC2}$, $DBCR1_{IAC2US}$, IAC2ER, $DBCR6_{IAC2XM}$
1	1	-	-	-	-	-	-	1	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	$DBCR1_{IAC12M}$

Table 1072. EDBRAC0 Resource Control (continued)

$EDBCR0_{EDM}$	$EDBRAC0_{IDM}$	$EDBRAC0_{RST}$	$EDBRAC0_{UDE}$	$EDBRAC0_{ICMP}$	$EDBRAC0_{BRT}$	$EDBRAC0_{IRPT}$	$EDBRAC0_{TRAP}$	$EDBRAC0_{IAC1}$	$EDBRAC0_{IAC2}$	$EDBRAC0_{IAC3}$	$EDBRAC0_{IAC4}$	$EDBRAC0_{IAC5}$	$EDBRAC0_{IAC6}$	$EDBRAC0_{IAC7}$	$EDBRAC0_{IAC8}$	$EDBRAC0_{DAC1}$	$EDBRAC0_{DAC34}$	$EDBRAC0_{DAC2}$	$EDBRAC0_{RET}$	$EDBRAC0_{DEVT1}$	$EDBRAC0_{DEVT2}$	$EDBRAC0_{PMI}$	$EDBRAC0_{MPU}$	$EDBRAC0_{CIRT}$	$EDBRAC0_{CRET}$	$EDBRAC0_{DNI}$	$EDBRAC0_{DQM}$	Software Accessible via mtspr, affected by p_reset_b
1	1	-	-	-	-	-	-	-	-	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	IAC3, $DBCR0_{IAC3}$, $DBCR1_{IAC3US}$, IAC3ER, $DBCR6_{IAC3XM}$
1	1	-	-	-	-	-	-	-	-	-	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	IAC4, $DBCR0_{IAC4}$, $DBCR1_{IAC4US}$, IAC4ER, $DBCR6_{IAC4XM}$
1	1	-	-	-	-	-	-	-	-	1	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	$DBCR1_{IAC34M}$
1	1	-	-	-	-	-	-	-	-	-	-	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	IAC5, $DBCR0_{IAC5}$, $DBCR5_{IAC5US}$, IAC5ER, $DBCR6_{IAC5XM}$
1	1	-	-	-	-	-	-	-	-	-	-	-	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	IAC6, $DBCR0_{IAC6}$, $DBCR5_{IAC6US}$, IAC6ER, $DBCR6_{IAC6XM}$
1	1	-	-	-	-	-	-	-	-	-	-	1	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	$DBCR5_{IAC56M}$
1	1	-	-	-	-	-	-	-	-	-	-	-	-	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	IAC7, $DBCR0_{IAC7}$, $DBCR5_{IAC7US}$, IAC7ER, $DBCR6_{IAC7XM}$
1	1	-	-	-	-	-	-	-	-	-	-	-	-	-	1	-	-	-	-	-	-	-	-	-	-	-	-	-	IAC8, $DBCR0_{IAC8}$, $DBCR5_{IAC8US}$, IAC8ER, $DBCR6_{IAC8XM}$
1	1	-	-	-	-	-	-	-	-	-	-	-	-	1	1	-	-	-	-	-	-	-	-	-	-	-	-	-	$DBCR5_{IAC78M}$
1	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1	-	-	-	-	-	-	-	-	-	-	-	-	DAC1, DVC1, DVC1U, $DBCR0_{DAC1}$, $DBCR2_{DAC1US}$, DAC1ER, $DBCR2_{DVC1M}$, DVC1BE $DBCR4_{DVC1C}$, DAC1XM

Table 1072. EDBRAC0 Resource Control (continued)

EDBCR0$_{EDM}$	EDBRAC0$_{IDM}$	EDBRAC0$_{RST}$	EDBRAC0$_{UDE}$	EDBRAC0$_{ICMP}$	EDBRAC0$_{BRT}$	EDBRAC0$_{IRPT}$	EDBRAC0$_{TRAP}$	EDBRAC0$_{IAC1}$	EDBRAC0$_{IAC2}$	EDBRAC0$_{IAC3}$	EDBRAC0$_{IAC4}$	EDBRAC0$_{IAC5}$	EDBRAC0$_{IAC6}$	EDBRAC0$_{IAC7}$	EDBRAC0$_{IAC8}$	EDBRAC0$_{DAC1}$	EDBRAC0$_{DAC34}$	EDBRAC0$_{DAC2}$	EDBRAC0$_{RET}$	EDBRAC0$_{DEVT1}$	EDBRAC0$_{DEVT2}$	EDBRAC0$_{PMI}$	EDBRAC0$_{MPU}$	EDBRAC0$_{CIRT}$	EDBRAC0$_{CRET}$	EDBRAC0$_{DNI}$	EDBRAC0$_{DQM}$	Software Accessible via mtspr, affected by p_reset_b
1	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1	-	-	-	-	-	-	-	-	-	-	DAC3, DAC4, DBCR7$_{DAC\{3,4\}}$, DAC{3,4}CFG, DAC{3,4}XM, DAC{3,4}XMH DBCR8$_{DAC\{3,4\}}$US, DAC{3,4}ER, DAC{3,4}M
1	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1	-	-	-	-	-	-	-	-	-	DAC2, DVC2, DVC2U, DBCR0$_{DAC2}$, DBCR2$_{DAC2US}$, DAC2ER, DBCR2$_{DVC2M}$, DVC2BE DBCR4$_{DVC2C}$, DAC2XM
1	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	—
1	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1	-	1	-	-	-	-	-	-	-	-	-	DBCR2$_{DAC12M}$
1	1	-	-	-	-	-	-	1	-	-	-	-	-	-	-	1	-	-	-	-	-	-	-	-	-	-	-	DBCR2$_{DAC1LNK}$
1	1	-	-	-	-	-	-	-	-	1	-	-	-	-	-	-	-	1	-	-	-	-	-	-	-	-	-	DBCR2$_{DAC2LNK}$
1	1	-	-	-	-	-	-	-	-	-	-	1	-	-	-	-	1	-	-	-	-	-	-	-	-	-	-	DBCR8$_{DAC3LNK}$
1	1	-	-	-	-	-	-	-	-	-	-	-	1	-	-	-	1	-	-	-	-	-	-	-	-	-	-	DBCR8$_{DAC4LNK}$
1	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1	-	-	-	-	-	-	-	-	DBCR0$_{RET}$
1	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1	-	-	-	-	-	-	-	DBCR0$_{DEVT1}$
1	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1	-	-	-	-	-	-	DBCR0$_{DEVT2}$
1	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1	-	-	-	-	-	All PMRs
1	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1	-	-	-	-	MPU entries with DEBUG bit set, MPU0CSR0$_{DR}$DEN,DWDEN,IDEN
1	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1	-	-	-	DBCR0$_{CIRPT}$
1	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1	-	-	DBCR0$_{CRET}$

Table 1072. EDBRAC0 Resource Control (continued)

$EDBCR0_{EDM}$	$EDBRAC0_{IDM}$	$EDBRAC0_{RST}$	$EDBRAC0_{UDE}$	$EDBRAC0_{ICMP}$	$EDBRAC0_{BRT}$	$EDBRAC0_{IRPT}$	$EDBRAC0_{TRAP}$	$EDBRAC0_{IAC1}$	$EDBRAC0_{IAC2}$	$EDBRAC0_{IAC3}$	$EDBRAC0_{IAC4}$	$EDBRAC0_{IAC5}$	$EDBRAC0_{IAC6}$	$EDBRAC0_{IAC7}$	$EDBRAC0_{IAC8}$	$EDBRAC0_{DAC1}$	$EDBRAC0_{DAC34}$	$EDBRAC0_{DAC2}$	$EDBRAC0_{RET}$	$EDBRAC0_{DEVT1}$	$EDBRAC0_{DEVT2}$	$EDBRAC0_{PMI}$	$EDBRAC0_{MPU}$	$EDBRAC0_{CIRT}$	$EDBRAC0_{CRET}$	$EDBRAC0_{DNI}$	$EDBRAC0_{DQM}$	Software Accessible via mtspr, affected by p_reset_b
1	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-		-	-	-	-	-	-	1	-	—
1	(1) -	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1	$DEVENT_{DQTAG}$, DDAM

1. IDM not required to be set to enable software access.

EDBRAC0 also controls which bits or fields in DBCR0–8, MPU0CSR0, and the PMRs are reset by assertion of **p_reset_b** when $EDBCR0_{EDM}$ = 1. Only software-owned bits or fields as shown in *Table 1072* are affected in this case, except that $DBCR0_{RST}$ and $DBSR_{MRR}$ are updated by assertion of **p_reset_b** regardless of the value of $EDBCR0_{EDM}$ or EDBRAC0.

53.3.4 Debug Event Select Register (DEVENT)

The Debug Event Select Register allows instrumented software to internally generate signals when a **mtspr** instruction is executed and this register is accessed. The values written to this register determine which of the **p_devnt_out[0:7]** processor output signals are asserted upon access. Writing a '1' to any of these bit positions will cause a one clock pulse to be generated on the corresponding output. For **p_devnt_out[0:3]**, a corresponding **jd_watchpt[x]** output is asserted as well to indicate a watchpoint has occurred. These signals may be used for internal core debug resources as well as for SoC level cross-triggering. See the SoC User's Manual for more information on SoC use cases.

The $DEVENT_{DEVNT}$ register field value is undefined on a read; it may or may not remain set to the last value written. Since it is unconditionally shared by hardware debug and software, software should not rely on any value remaining.

The upper 8 bits of the DEVENT register also provide the DQTAG used to identify channels within Data Acquisition Messages. See the "Data Acquisition ID Tag Field" section in the Core (e200z420n3) Nexus 3 Module chapter for more detail on the DQTAG.

The DEVENT register is shown in *Figure 1111*.

Offset: Refer to SPR - 975 Access: Read-only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DQTAG								0							
W																
Reset(1)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0								DEVNT							
W																
Reset(1)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. Reset by processor reset **p_reset_b** if $EDBCR0_{EDM}$ = 0, as well as unconditionally by **m_por**. If $EDBCR0_{EDM}$ = 1, EDBRAC0 masks off hardware-owned resources from reset by **p_reset_b** and only software-owned resources indicated by EDBRAC0 will be reset by **p_reset_b.** Note that DEVNT field is shared by hardware and software but is always reset by **p_reset_b**.

Figure 1111. Debug Event Select Register (DEVENT)

Table 1073 provides bit definitions for the Debug Event Register.

Table 1073. DEVENT field descriptions

Field	Description
0:7 DQTAG	Data Acquisition Message IDTAG channel identifier (supplied to Nexus 3)
8:23	Reserved, should be cleared.
24:31 DEVNT	Debug Event Signals 00000000 No signal is asserted xxxxxxx1 **p_devnt_out[0]** and **jd_watchpt[12]** are asserted for one clock xxxxxx1x **p_devnt_out[1]** and **jd_watchpt[13]** are asserted for one clock xxxxx1xx **p_devnt_out[2]** and **jd_watchpt[20]** are asserted for one clock xxxx1xxx **p_devnt_out[3]** and **jd_watchpt[21]** are asserted for one clock xxx1xxxx **p_devnt_out[4]** is asserted for one clock xx1xxxxx **p_devnt_out[5]** is asserted for one clock x1xxxxxx **p_devnt_out[6]** is asserted for one clock 1xxxxxxx **p_devnt_out[7]** is asserted for one clock

53.3.5 Debug Data Acquisition Message Register (DDAM)

The Debug Data Acquisition Message Register allows instrumented software to generate real-time Data Acquisition Messages (as defined by Nexus 3) via a **mtspr** instruction to this register. See the “Data Acquisition Messaging” section in the Core (e200z420n3) Nexus 3 Module chapter for details.

The DDAM register is shown in *Figure 1112*.

Offset: Refer to SPR - 576 Access: Read-only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DDAM[0:15]															
W																
Reset(1)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	DDAM[16:31]															
W																
Reset(1)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. Reset by processor reset **p_reset_b** if $EDBCR0_{EDM}$ = 0, as well as unconditionally by **m_por**. If $EDBCR0_{EDM}$ = 1, EDBRAC0 masks off hardware-owned resources from reset by **p_reset_b** and only software-owned resources indicated by EDBRAC0 will be reset by **p_reset_b**.

Figure 1112. Debug Data Acquisition Message Register (DDAM)

Table 1074 provides bit definitions for the Debug Data Acquisition Message Register.

Table 1074. DDAM field descriptions

Field	Description
0:31 DDAM	Value to be transmitted in a Data Acquisition Message (DQM) (supplied to Nexus 3 with strobe)

53.4 Using Debug Resources for Stack Limit Checking

The DAC1,2 and DAC3,4 resources can be used for stack overflow/underflow detection when not being used as a hardware or software debug resource. Stack limit checking is available regardless of EDM or IDM mode, and when resources used for stack limit checking are owned by software, will utilize a DSI or machine check exception. Software-owned stack limit checking does not require IDM to be set. Hardware owned stack limit checking requires EDM to be set.

When stack limit checking is enabled, and DAC resources used for stack limit checking are owned by software, DAC events are not generated for resources configured to perform stack limit checking, and no DBSR DAC status flag will be set due to a detected stack limit violation. Instead, depending on the processor mode, a data storage interrupt or a machine check exception is signaled. When stack limit checking is enabled, and DAC resources used for stack limit checking are owned by hardware, DAC events will be generated for resources configured to perform stack limit checking, and the EDBSR0 DAC status flag will be set due to a detected stack limit violation, causing entry into debug halted mode in the same way as a DAC exception normally does. The only difference is that qualification of the access address is performed as discussed in the next paragraph.

Stack limit checking is implemented in the same way as range compares using DAC1,2 or 3,4, or extended masking using DAC1 or DAC3, but qualify a load or store access address with the use of GPR R1 as the base or index register used to compute an effective address when a load or store instruction is executed. No stack limit checking is performed for other instructions which may calculate an EA using GPR R1 such as cache, MMU/MPU management instructions, or instructions which indicate GPR R1 is used to hold a decoration value (for decorated load/store type instructions). When DAC resources

configured to perform stack limit checking are not owned by hardware, if a stack limit violation occurs when performing the load or store, the access is aborted, and an error report machine check is generated, with MCSRR0 pointing to the address of the load or store access which generated the stack overflow/underflow. If DAC resources configured to perform stack limit checking are owned by hardware, then a normal DAC event is generated (but qualified with use of GPR R1), and debug mode entry will occur in the same manner as for a non-stack limit DAC event.

Enabling of this functionality is described in more detail in *Table 1065*. Note that IDM is not required to be set for software-owned stack limit checking.

In contrast to the normal case of DAC address matching resulting in a debug event, the stack limit check address compare logic operates differently for stack limit checking. When using resources for stack limit checking, a DACn hit to a resource enabled for performing stack limit checking on a stack access indicates a stack limit violation.

When stack limit checking is enabled by the setting of the $DBCR4_{DAC1CFG}$ field to '1000' or the $DBCR8_{DAC3CFG}$ field to '1000' (or both), and the corresponding DACn resources are owned by software, DACn events are not generated and the DBSR DAC status flag will not be set due to a detected stack limit hit. Instead, if stack limit checking is enabled for supervisor mode stack accesses in DAC1 or DAC3, and a compare hit occurs for a supervisor mode stack access (A load or store using GPR R1 in the <EA> calculation), a machine check exception is signaled. If stack limit checking is enabled for user mode accesses, a DSI exception is signaled when a stack limit checking enabled DAC1 or DAC3 compare hit occurs for a user mode access. A watchpoint for DAC1 or DAC3 will be generated when the stack limit violation is detected, even though the instruction does not complete. Stack limit checking for supervisor mode stack accesses is considered enabled when either the $DBCR4_{DAC1CFG}$ field is set to '1000' with $DBCR2_{DAC1US}$ set to '00' or '10', or the $DBCR7_{DAC3CFG}$ field is set to '1000' with $DBCR8_{DAC3US}$ set to '00' or '10', or both. Stack limit checking for user mode stack accesses is considered enabled when either the $DBCR4_{DAC1CFG}$ field is set to '1000' with $DBCR2_{DAC1US}$ set to '00' or '01', or the $DBCR7_{DAC3CFG}$ field is set to '1000' with $DBCR8_{DAC3US}$ set to '00' or '01', or both. Note that unlike regular debug DAC events, both halves of a misaligned access are checked for limit violations.

When stack limit checking is enabled for a stack access, and DACn resources are owned by hardware, the EDBSR0 DAC status flag will be set due to a detected stack limit violation, to cause entry into debug halted mode or to generate a watchpoint, or both, in the same way as a DAC event normally does, that is after the access has completed. The only difference is that qualification of the access address is performed as discussed in the next paragraph. If the access is aborted due to a DSI or other exception such as machine check condition, the $EDBSR0_{IDE}$ status bit will also be set to indicate that the data access instruction was not completed.

Stack limit checking is implemented in the same way as address compares using DACn, but qualify a load or store access address with the use of GPR R1 as the base or index register used to compute an effective address when a load or store instruction is executed. No stack limit checking is performed for other instructions which may calculate an EA using GPR R1 such as cache, MMU/MPU management instructions, or instructions which indicate GPR R1 is used to hold a decoration value (for decorated load/store type instructions). When DACn resources are not owned by hardware, if a stack limit violation occurs (a hit to a stack limit enabled DAC is detected) when performing the load or store, the access request is aborted and the access is not performed. Instead, for supervisor mode accesses, an error report machine check exception is generated, with MCSRR0 pointing to the address of the load or store instruction which generated the stack overflow/underflow, and for user mode

accesses, a DSI exception is generated, with SRR0 pointing to the address of the load or store instruction which generated the stack overflow/underflow. If all stack limit check enabled DACn resources are owned by hardware, then a DAC event is generated (but qualified with use of GPR R1), and debug mode entry will occur in the same manner as for a non-stack limit DAC event. In this case, the instruction and access will normally have completed, in the same manner as a normal DAC event.

Independent limit checks for supervisor and user accesses may be implemented by allocating independent DACn resources to each, or a single limit may be applied using a single DACn resource. Typically, a DAC1,2 pair operating in exclusive address range mode is utilized for stack limit checking for a single shared limit. For separate limits, DAC3,4 may also be utilized. If more than one DACn resource is utilized, a DAC hit on any resource utilized for stack limit checking will cause the corresponding stack limit exception action to occur. If both a hardware-owned and a software-owned resource generate a stack limit exception for a given load or store, the software resource will have priority, since it is detected prior to completion of the access, and the access is aborted, thus the hardware event will not occur.

Enabling of this functionality is described in more detail in the description of the DACnCFG fields in *Table 1065* and *Table 1068* in *Section 53.3.2*. Note that for DAC1 and DAC2, access type (read, write) control is part of DBCR0.

53.5 External Debug Support

External debug support is supplied through the OnCE controller serial interface which allows access to internal CPU registers and other system state while the CPU is halted in debug mode. All debug resources including DBCR0–8, DBSR, IAC1–8, DAC1–4, and DVC1–2 are accessible through the serial OnCE interface in external debug mode. Setting the $EDBCR0_{EDM}$/$DBCR0_{EDM}$ bit to '1' through the OnCE interface enables external debug mode, and unless otherwise permitted by the settings in EDBRAC0, disables software updates to the debug control registers. When $EDBCR0_{EDM}$ is set, debug events enabled to set respective status bits will also cause the CPU to enter Debug Mode if the event is not masked in EDBSRMSK0, as opposed to generating Debug Interrupts, unless the specific events are allocated to software via the settings in EDBRAC0. In Debug Mode, the CPU is halted at a recoverable boundary, and an external Debug Control Module may control CPU operation through the On-Chip Emulation logic (OnCE).

Note that the descriptions of events in the subsections of *Section 53.2* refer to setting DBSR status bits, however, when resources are owned by hardware, the events for those resources set the respective status bits in EDBSR0 instead of DBSR.

Note: On the initial setting of $EDBCR0_{EDM}$ to '1', other bits in DBCR0 will remain unchanged. After $EDBCR0_{EDM}$ has been set, all debug register resources may be subsequently controlled through the OnCE interface. The CPU should be placed into debug mode via the OCR_{DR} control bit prior to writing EDM to '1'. This gives the debugger the opportunity to cleanly write to the DBCRx registers and the DBSR to clear out any residual state / control information which could cause unintended operation.

*It is intended for the CPU to remain in external debug mode ($EDBCR0_{EDM}$ = 1) in order to single step or perform other debug mode entry/ reentry via the OCR_{DR}, by performing go+noexit commands, or by assertion of the **jd_de_b** signal.*

*$EDBCR0_{EDM}$ operation will be blocked if OnCE operation is disabled (**jd_en_once** negated) regardless of whether it is set or cleared. This means that if $EDBCR0_{EDM}$ was previously*

*set, and then **jd_en_once** is negated (this should not occur), entry into debug mode will be blocked, and all hardware debug events are blocked. Watchpoints are not blocked.*

Due to clock domain design, the CPU clock (**m_clk**) must be active in order to perform writes to debug registers other than the OnCE Command register (OCMD), the OnCE Control register (OCR), External Debug Control Register 0 (EDBCR0), External Debug Status register 0 (EDBSR0), External Debug Status Register Mask 0 (EDBSRMSK0), or the $EDBCR0_{EDM}$ bit. Register read data is synchronized back to the **j_tclk** clock domain. The OnCE Control register provides the capability of signaling the system level clock controller that the CPU clock should be activated if not already active.

Updates to the DBCRx, DBSR, and most other debug registers via the OnCE interface should be performed with the CPU in debug mode to guarantee proper operation. Due to the various points in the CPU pipeline where control is sampled and event handshaking is performed, it is possible that modifications to these registers while the CPU is running may result in early or late entry into debug mode, and may have incorrect status posted in the DBSR register.

If resource sharing is enabled via EDBRAC0, updates to the EDBRAC0, DBCRx, and DBSR registers must be performed with the CPU in debug mode, since simultaneous updates of register portions could otherwise be attempted, and such updates are not guaranteed to properly occur. The results of such an attempt are undefined.

53.5.1 External Debug Registers

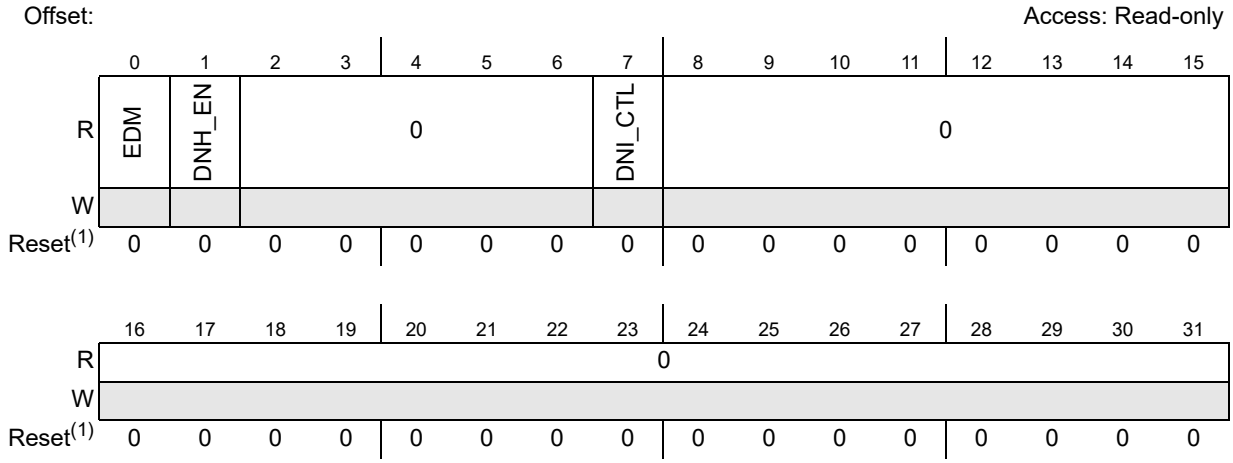
The external debug registers are used for controlling several debug aspects of the core and reporting status while in External Debug Mode.

53.5.1.1 External Debug Control Register 0 (EDBCR0)

EDBCR0 is a control register accessible to an external debugger through the OnCE/Jtag port. An external development tool can write to this register in order to enable external debug mode or to enable Debugger Notify Halt instructions (**e_dnh**, **se_dnh**), as well as to control aspects of Debugger Notify Interrupt instructions (**e_dni**, **se_dni**).

EDBCR0 is not accessible by software, However, the state of $EDBCR0_{EDM}$ is reflected as a read-only bit in $DBCR0_{EDM}$ to software. There is only one physical EDM bit implemented; it is reflected in both the DBCR0 and EDBCR0 registers, and may be written and read using either register by the hardware debugger. For future compatibility, EDBCR0 updates are preferred.

EDBCR0 is shown in *Figure 1113*.



1. EDBCR0 is affected (reset) by **j_trst_b** or **m_por** assertion, and remains reset while in the Test_Logic_Reset state, but is not affected by **p_reset_b**.

Figure 1113. External Debug Control Register 0 (EDBCR0)

Table 1075 provides bit definitions for External Debug Control Register 0.

Table 1075. EDBCR0 field descriptions

Field	Description
0 EDM	External Debug Mode. This bit is also reflected in DBCR0 0 External debug mode disabled. Internal debug events not mapped into external debug events. 1 External debug mode enabled. Hardware-owned events will not cause the CPU to vector to interrupt code. Software is not permitted to write to debug registers {DBCRx, IAC1-8, DAC1-4, DVC1-2[U]} unless permitted by settings in EDBRAC0. When external debug mode is enabled, hardware-owned resources in debug registers are not affected by processor reset **p_reset_b**. This allows the debugger to set up hardware debug events which remain active across a processor reset.
1 DNH_EN	**dnh** Instruction Enable 0 execution of **e_dnh** and **se_dnh** instructions cause illegal instruction exceptions to occur. 1 execution of **e_dnh** and **se_dnh** instructions cause entry into debug mode and a debug halt occurs, regardless of the value of EDM.
7 DNI_CTL	**dni** Instruction Control 0 When the **dni** resource is owned by hardware, the MSR_{DE} bit is cared, and when MSR_{DE} = 0, execution of **e_dni** and **se_dni** instructions are nop'ed and no entry into debug mode occurs. 1 When the **dni** resource is owned by hardware, the MSR_{DE} bit is don't-cared, and execution of **e_dni** and **se_dni** instructions cause entry into debug mode and a debug halt occurs, regardless of the value of MSR_{DE}. Note that this control bit is only used when the dni resource is owned by hardware via control in $EDBRAC0_{DNI}$, and thus also only when EDM = 1

53.5.1.2 External Debug Status Register 0 (EDBSR0)

The External Debug Status Register 0 (EDBSR0) contains status on debug events owned by hardware. The External Debug Status Register 0 is set via hardware, and read and

cleared via OnCE access by the debugger. Clearing is done by writing to the External Debug Status Register via the OnCE port, with a '1' in any bit position that is to be cleared and '0' in all other bit positions. The write data to EDBSR0 is not direct data, but a mask. A '1' causes the bit to be cleared, and a '0' has no effect. The EDBSR0 register is shown in *Figure 1114*.

Offset: Access: Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	IDE	UDE	DNH	0	ICMP	BRT	IRPT	TRAP	IAC	0			DACR	DACW	0	DNI
W																
Reset(1)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	RET	0				DEVT1	DEVT2	PMI	MPU	CIRPT	CRET	VLES	DAC_OFST			0
W																
Reset(1)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. Reset by **j_trst_b** or **m_por** assertion, and remains reset while in the Test_Logic_Reset state or while $EDBCR0_{EDM}$ = 0.

Figure 1114. External Debug Status Register 0 (EDBSR0)

Table 1076 provides bit definitions for External Debug Status Register 0.

Table 1076. EDBSR0 field descriptions

Field	Description
0 IDE	Imprecise Debug Event Set to 1 if $EDBCR0_{EDM}$ = 1 and an imprecise debug event occurs for a hardware-owned DAC event due to a load or store which is terminated with error, or if a hardware-owned ICMP event occurs in conjunction with a EFPU round exception. This bit will not be set for imprecise debug events which are masked via settings in EDBSRMSK0.
1 UDE	Unconditional Debug Event Set to 1 if a hardware-owned Unconditional debug event occurred.
2 DNH	Debugger Notify Halt Event Set to 1 if a debugger notify halt instruction was executed and caused a debug halt.
4 ICMP	Instruction Complete Debug Event Set to 1 if a hardware-owned Instruction Complete debug event occurred.
5 BRT	Branch Taken Debug Event Set to 1 if a hardware-owned Branch Taken debug event occurred.
6 IRPT	Interrupt Taken Debug Event Set to 1 if a hardware-owned Interrupt Taken debug event occurred.
7 TRAP	Trap Taken Debug Event Set to 1 if a hardware-owned Trap Taken debug event occurred.
8 IAC	Instruction Address Compare 1 Debug Event Set to 1 if a hardware-owned IAC debug event occurred.

Table 1076. EDBSR0 field descriptions (continued)

Field	Description
12 DACR	Data Address Compare Read Debug Event Set to 1 if a hardware-owned read-type DAC debug event occurred
13 DACW	Data Address Compare Write Debug Event Set to 1 if a hardware-owned write-type DAC1 debug event occurred
15 DNI	DNI Debug Event Set to 1 if a hardware-owned DNI debug event occurred
16 RET	Return Debug Event Set to 1 if a hardware-owned Return debug event occurred
21 DEVT1	External Debug Event 1 Debug Event Set to 1 if a hardware-owned DEVT1 debug event occurred
22 DEVT2	External Debug Event 2 Debug Event Set to 1 if a hardware-owned DEVT2 debug event occurred
23 PMI	Performance Monitor Interrupt Debug Event Set to 1 if a Performance Monitor Interrupt event occurred with $EDBRAC0_{PMI}$ = 0 regardless of the setting of $PMGC0_{UDI}$
24 MPU	Memory Protection Unit Debug Event Set to 1 if a hardware-owned MPU debug event occurred. For MPU debug events, if $EDBSRMSK0_{MPU}$ is cleared, the IAC, DACR, or DACW status bit will also be set, depending on the type of access which matched a region descriptor enabled to generate debug events, in order to further define the type of MPU debug event. If $EDBSRMSK0_{MPU}$ is set at the time a MPU debug event occurs, the IAC, DACR, and DACW status bits will not be set by MPU debug events, in order to properly mask the MPU debug event. Note that hardware MPU debug events on data accesses normally occur after the data access is performed and the instruction completes, thus act like a normal DAC debug event. The instruction may not complete if a DSI occurs. In this case, IDE will be set to indicate this occurrence.
25 CIRPT	Critical Interrupt Taken Debug Event Set to 1 if a hardware-owned Critical Interrupt Taken debug event occurred.
26 CRET	Critical Return Debug Event Set to 1 if a hardware-owned Critical Return debug event occurred
27 VLES	VLE Status Set to 1 if a hardware-owned ICMP, BRT, TRAP, RET, CRET, IAC, or DAC debug event occurred on a PowerISA VLE Instruction. Also set for execution of an e_dnh or se_dnh instruction when enabled by $EDBCR0_{DNH_EN}$. Undefined for IRPT, CIRPT, DEVT[1,2], and UDE events
28:31 DAC_OFST	Data Address Compare Offset Indicates offset-1 of saved DSRR0 value from the address of the load or store instruction which took a hardware-owned DAC Debug exception, unless a simultaneous DSI error occurs, in which case this field is set to 3‘b000 and $EDBSR0_{IDE}$ is set to 1. Normally set to 3‘b000 by a non-DVC DAC. A DVC DAC may set this field to any value.

53.5.1.3 External Debug Status Register Mask 0 (EDBSRMSK0)

The External Debug Status Register Mask 0 (EDBSRMSK0) is used to mask debug events set in EDBSR0 from causing entry into debug halted mode. A '1' stored in any mask bit prevents debug mode entry caused by the corresponding bit being set in EDBSR0. The mask has no effect on DBSR actions or on the setting of EDBSR0 status bits by hardware-owned events, except that the IDE bit will not be set by imprecise hardware-owned debug events which are masked. EDBSRMSK0 may be used to allow debug events owned by hardware to be configured for watchpoint generation purposes without causing debug mode entry when the watchpoint occurs. EDBSRMSK0 is read and written via OnCE access by the debugger. No software access is provided. The EDBSRMSK0 register is shown in *Figure 1115*.

Offset: Access: Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	UDE	DNH	0	ICMP	BRT	IRPT	TRAP	IAC	0			DACR	DACW	0	DNI
W																
Reset(1)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	RET	0				DEVT1	DEVT2	PMI	MPU	CIRPT	CRET	0				
W																
Reset(1)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. Reset by **j_trst_b** or **m_por** assertion, and remains reset while in the Test_Logic_Reset state or while $EDBCR0_{EDM}$ = 0.

Figure 1115. External Debug Status Register Mask 0 (EDBSRMSK0)

Table 1077 provides bit definitions for External Debug Status Register Mask 0.

Table 1077. EDBSRMSK0 field descriptions

Field	Description
1 UDE	Unconditional Debug Event Set to 1 to mask debug mode entry by $EDBSR0_{UDE}$
2 DNH	Debugger Notify Halt Event Set to 1 to mask debug mode entry by $EDBSR0_{DNH}$
4 ICMP	Instruction Complete Debug Event Set to 1 to mask debug mode entry by $EDBSR0_{ICMP}$
5 BRT	Branch Taken Debug Event Set to 1 to mask debug mode entry by $EDBSR0_{BRT}$
6 IRPT	Interrupt Taken Debug Event Set to 1 to mask debug mode entry by $EDBSR0_{IRPT}$
7 TRAP	Trap Taken Debug Event Set to 1 to mask debug mode entry by $EDBSR0_{TRAP}$

Table 1077. EDBSRMSK0 field descriptions (continued)

Field	Description
8 IAC	Instruction Address Compare Debug Event Set to 1 to mask debug mode entry by $EDBSR0_{IAC}$
12 DACR	Data Address Compare 1 Read Debug Event Set to 1 to mask debug mode entry by $EDBSR0_{DACR}$
13 DACW	Data Address Compare 1 Write Debug Event Set to 1 to mask debug mode entry by $EDBSR0_{DACW}$
15 DNI	DNI Debug Event Set to 1 to mask debug mode entry by $EDBSR0_{DNI}$
16 RET	Return Debug Event Set to 1 to mask debug mode entry by $EDBSR0_{RET}$
21 DEVT1	External Debug Event 1 Debug Event Set to 1 to mask debug mode entry by $EDBSR0_{DEVT1}$
22 DEVT2	External Debug Event 2 Debug Event Set to 1 to mask debug mode entry by $EDBSR0_{DEVT2}$
23 PMI	Performance Monitor Interrupt Debug Event Set to 1 to mask debug mode entry by $EDBSR0_{PMI}$
24 MPU	Memory Protection Unit Debug Event Set to 1 to mask debug mode entry by $EDBSR0_{MPU}$
25 CIRPT	Critical Interrupt Taken Debug Event Set to 1 to mask debug mode entry by $EDBSR0_{CIRPT}$
26 CRET	Critical Return Debug Event Set to 1 to mask debug mode entry by $EDBSR0_{CRET}$

53.5.1.4 External Debug Data Effective Address Register (EDDEAR)

The External Debug Data Effective Address Register (EDDEAR) contains address information for hardware-owned data address compare debug events, including MPU DAC events. EDDEAR is update by hardware with the effective address of the load, store, or cache control operation when an unmasked data address compare event is recorded in EDBSR0 if the previous values of $EDBSR0_{DAC\{R,W\}}$ bits which are unmasked in EDBSRMSK0 are zero. Once an EDDEAR update is performed, these unmasked bits must be cleared by the hardware debugger prior to another EDDEAR hardware update occurring. A subsequent hardware-owned unmasked DAC event will not update the EDDEAR register if either of the $EDBSR0_{DAC\{R,W\}}$ bits are set and are not masked by EDBSRMSK0, in order to capture the first unmasked event address. EDDEAR is read and written via OnCE access by the debugger. No software access is provided.

The EDDEAR register is shown in *Figure 1116*.

Offset: Access: Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Data Effective Address[0:15]															
W																
Reset	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	Data Effective Address[16:31]															
W																
Reset	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—

Figure 1116. External Debug Data Effective Address Register (EDDEAR)

53.5.2 OnCE Introduction

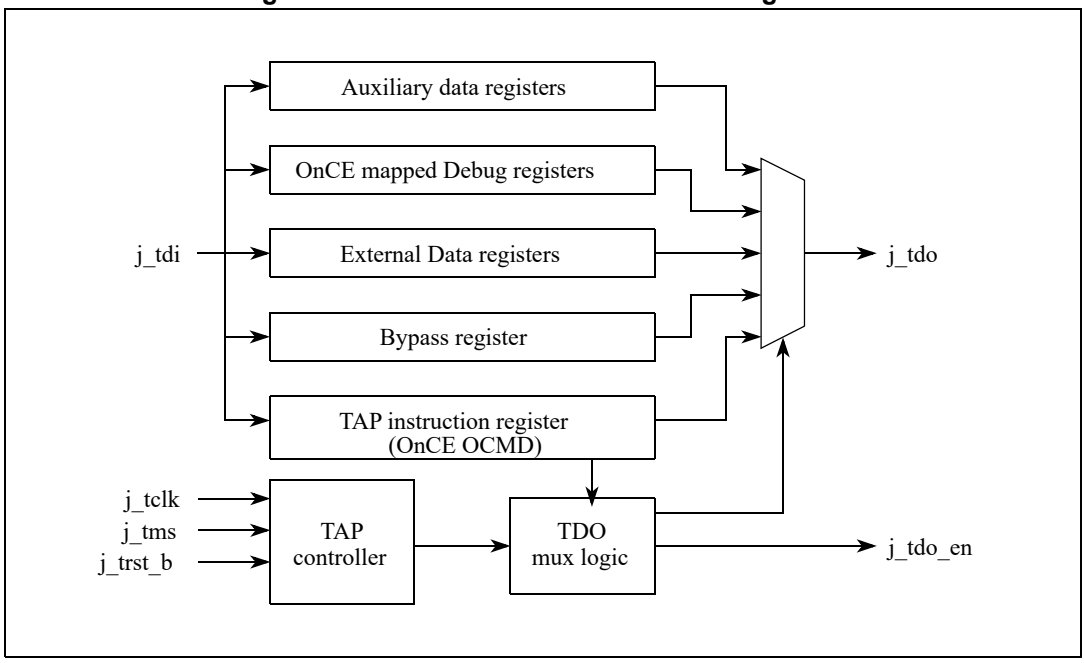
The e200z420n3 on-chip emulation circuitry (OnCE™/Nexus Class 1 interface) provides a means of interacting with the e200z420n3 core and integrated system so that a user may examine registers, memory, or on-chip peripherals facilitating hardware/software development. OnCE operation is controlled via an industry standard IEEE 1149.1 TAP controller. By using public instructions, the external hardware debugger can freeze or halt the CPU, read and write internal state, and resume normal execution. The core does not contain IEEE 1149.1 standard boundary cells on its interface, as it is a building block for further integration. It does not support the JTAG related boundary scan instruction functionality, although JTAG public instructions may be decoded and signaled to external logic.

The OnCE logic provides for Nexus Class 1 static debug capability (utilizing the same set of resources available to software while in internal debug mode).

In order to enable full OnCE operation, the **jd_enable_once** input signal must be asserted. In some system integrations, this is automatic, since the input will be tied asserted. Other integrations may require the execution of the Enable OnCE command via the TAP and appropriate entry of serial data. Exact requirements will be documented by the integrated product specification. The **jd_enable_once** input signal should not change state during a debug session, or undefined activity may occur.

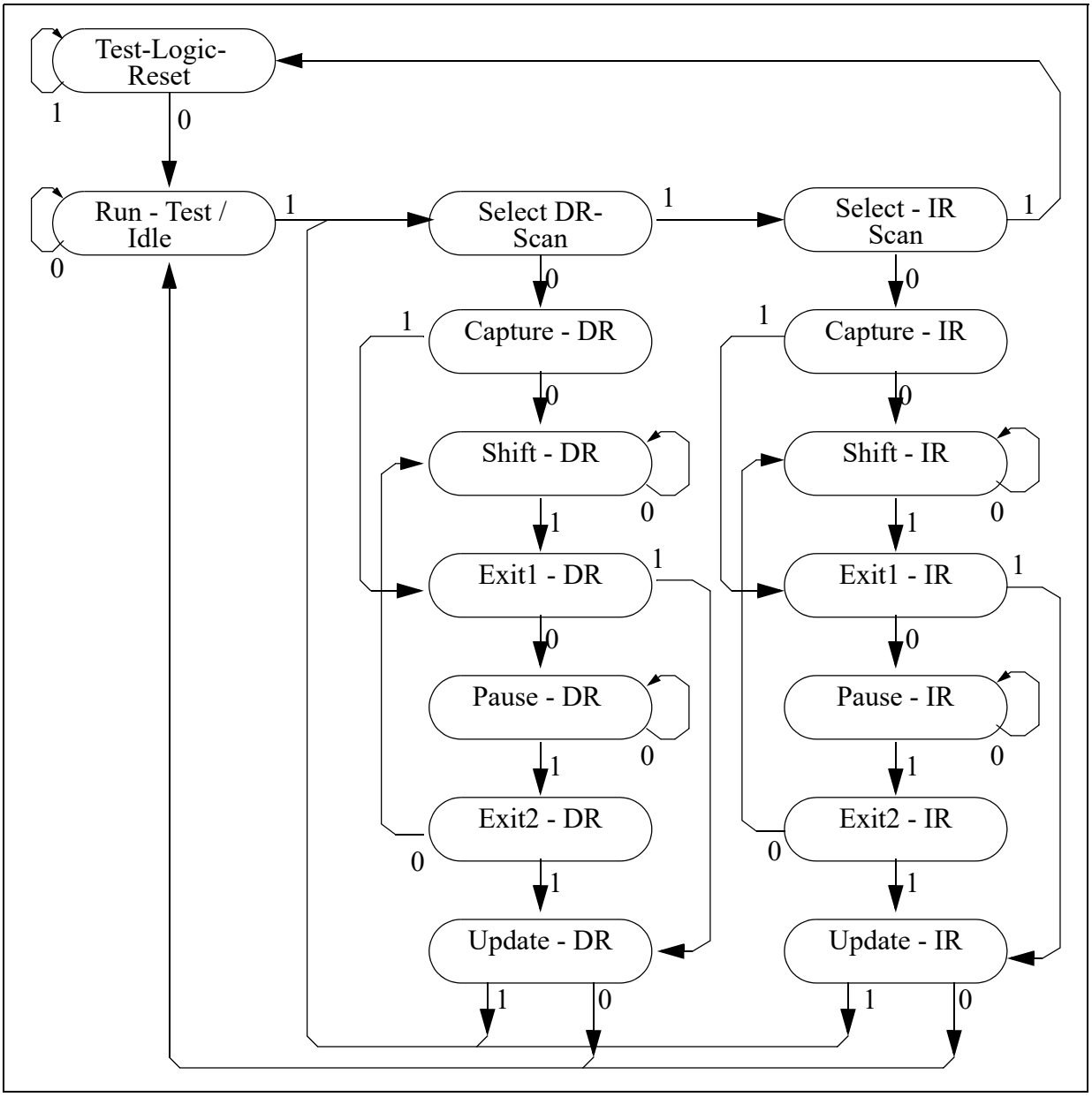
The following figures show the TAP controller state model and the TAP registers implemented by the OnCE logic.

Figure 1117. OnCE TAP Controller and Registers



The OnCE controller is implemented as a 16-state FSM, with a one-to-one correspondence to the states defined for the JTAG TAP controller.

Figure 1118. OnCE controller implementation



Access to processor registers and the contents of memory locations are performed by enabling external debug mode (setting $EDBCR0_{EDM}$ to '1'), placing the processor into debug mode, followed by scanning instructions and data into and out of the CPU Scan Chain (CPUSCR); execution of scanned instructions by the CPU is used as the method to access required data. Memory locations may be read by scanning a load instruction into the CPU core which will reference the desired memory location, executing the load instruction, and then scanning out the result of the load. Other resources are accessed in a similar manner.

The initial entry by the CPU into the debug state (or mode) from normal, Waiting, Stopped, or Halted states (all indicated via the OnCE Status Register (OSR), *Section 53.5.6.1*) by assertion of one or more debug requests, begins a *debug session*. The **jd_debug_b** output signal indicates that a debug session is in progress, and the OSR will indicate the CPU is in

the debug state. Instructions may the be single-stepped by scanning new values into the CPUSCR, and performing a OnCE go+noexit command (See *Section 53.5.6.2*). The CPU will then temporarily exit the debug state (but not the debug session) to execute the instruction, and will then return to the debug state (again indicated via the OnCE Status Register (OSR)). The debug session remains in force until the final OnCE go+exit command is executed, at which time the CPU will return to the previous state it was in (unless a new debug request is pending). A scan into the CPUSCR is required prior to executing each go+exit or go+noexit OnCE command.

53.5.3 JTAG/OnCE Pins

The JTAG/OnCE pin interface is used to transfer OnCE instructions and data to the OnCE control block. Depending on the particular resource being accessed, the CPU may need to be placed in the Debug mode. For resources outside of the CPU block and contained in the OnCE block, the processor is not disturbed, and may continue execution. If a processor resource is required, an internal debug request (**dbg_dbgrq**) may be asserted to the CPU by the OnCE controller, and causes the CPU to finish the current instruction being executed, save the instruction pipeline information, enter Debug Mode, and wait for further commands. Asserting **dbg_dbgrq** will cause the chip to exit the low power mode enabled by the setting of MSR_{WE}, as well as temporarily exiting the Waiting, Stopped or Halted power management states.

Table 1078 details the primary JTAG/OnCE interface signals.

Table 1078. JTAG/OnCE Primary Interface Signals

Signal name	Type	Description
j_trst_b	I	JTAG test reset
j_tclk	I	JTAG test clock
j_tms	I	JTAG test mode select
j_tdi	I	JTAG test data input
j_tdo	O	Test data out to master controller or pad
j_tdo_en(1)	O	Enables TDO output buffer

1. j_tdo_en is asserted when the TAP controller is in the shift_DR or shift_IR state.

A full description of JTAG pins is provided in the JTAG Support Signals section of the Core (e200z420n3) Core Complex Overview.

53.5.4 OnCE Internal Interface Signals

The following paragraphs describe the OnCE interface signals to other internal blocks associated with the OnCE controller.

53.5.4.1 CPU Debug Request (dbg_dbgrq)

The **dbg_dbgrq** signal is asserted by the OnCE control logic to request the CPU to enter the debug state. It may be asserted for a number of different conditions, and causes the CPU to finish the current instruction being executed, save the instruction pipeline information, enter the debug mode, and wait for further commands.

53.5.4.2 CPU Debug Acknowledge (cpu_dbgack)

The **cpu_dbgack** signal is asserted by the CPU upon entering the debug state. This signal is used as part of the handshake mechanism between the OnCE control logic and the rest of the CPU. The CPU core may enter debug mode either through a software or hardware event.

53.5.4.3 CPU Address, Attributes

The CPU address and attribute information are used by the Nexus3 unit with information for real-time address trace information.

53.5.4.4 CPU Data

The CPU data buses are used to supply the Nexus3 debug unit with information for real-time data trace capability.

53.5.5 OnCE Interface Signals

The following paragraphs describe additional OnCE interface signals to other external blocks such as a Nexus Controller and external blocks which may need information pertaining to debug operation.

53.5.5.1 OnCE Enable (jd_en_once)

The OnCE enable signal **jd_en_once** is used to enable the OnCE controller to allow certain instructions and operations to be executed. Assertion of this signal will enable the full OnCE command set, as well as operation of control signals and OnCE Control register functions. When this signal is disabled, only the Bypass, ID and Enable_OnCE commands are executed by the OnCE unit, and all other commands default to a “Bypass” command. The OnCE Status register (OSR) is not visible when OnCE operation is disabled. In addition, OnCE Control register (OCR) functions are disabled, as is the operation of the **jd_de_b** input. Secure systems may choose to leave the **jd_en_once** signal negated until a security check has been performed. Other systems should tie this signal asserted to enable full OnCE operation. The **j_en_once_regsel** output signal is provided to assist external logic performing security checks.

The **jd_en_once** input must only change state during the Test-Logic-Reset, Run-Test/Idle, or Update_DR TAP states. A new value will take affect after one additional **j_tclk** cycle of synchronization. In addition, **jd_enable_once** input signal must not change state during a debug session, or undefined activity may occur.

53.5.5.2 OnCE Debug Request/Event (jd_de_b, jd_de_en)

If implemented at the SoC level, a system level bidirectional open drain debug event pin **DE_b** (not part of the e200z420n3 interface) provides a fast means of entering the Debug Mode of operation from an external command controller (when input) as well as a fast means of acknowledging the entering of the Debug Mode of operation to an external command controller (when output). The assertion of this pin by a command controller causes the CPU core to finish the current instruction being executed, save the instruction pipeline information, enter Debug Mode, and wait for commands to be entered. If **DE_b** was used to enter the Debug Mode then **DE_b** must be negated after the OnCE controller responds with an acknowledge and before sending the first OnCE command. The assertion of this pin by the CPU Core acknowledges that it has entered the Debug Mode and is waiting for commands to be entered.

To support operation of this system pin, the OnCE logic supplies the **jd_de_en** output and samples the **jd_de_b** input when OnCE is enabled (**jd_en_once** asserted). Assertion of **jd_de_b** will cause the OnCE logic to place the CPU into Debug Mode. Once Debug Mode has been entered, the **jd_de_en** output will be asserted for three **j_tclk** periods to signal an acknowledge. **jd_de_en** can be used to enable the open-drain pulldown of the system level **DE_b** pin.

For systems which do not implement a system level bidirectional open drain debug event pin **DE_b**, the **jd_de_en** and **jd_de_b** signals may still be used to handshake debug entry.

53.5.5.3 OnCE Debug Output (jd_debug_b)

The OnCE Debug output **jd_debug_b** is used to indicate to on-chip resources that a debug session is in progress. Peripherals and other units may use this signal to modify normal operation for the duration of a debug session, which may involve the CPU executing a sequence of instructions solely for the purpose of visibility/system control which are not part of the normal instruction stream the CPU would have executed had it not been placed in debug mode. This signal is asserted the first time the CPU enters the debug state, and remains asserted until the CPU is released by a write to the OnCE Command Register with the GO and EX bits set, and a register specified as either “No Register Selected” or the CPUSCR. This signal will remain asserted even though the CPU may enter and exit the debug state for each instruction executed under control of the OnCE controller. See *Section 53.5.6.2* for more information on the function of the GO and EX bits. This signal is not normally used by the CPU.

53.5.5.4 CPU Clock On Input (jd_mclk_on)

The CPU Clock On input **jd_mclk_on** is used to indicate that the CPU’s **m_clk** input is active. This input signal is expected to be driven by system logic external to the e200z420n3 core, is synchronized to the **j_tclk** (scan clock) clock domain, and is presented as a status flag on the **j_tdo** output during the Shift_IR state. External firmware may use this signal to ensure proper scan sequences will occur to access debug resources in the **m_clk** clock domain.

53.5.5.5 Watchpoint Events (jd_watchpt[0:31])

The **jd_watchpt[0:31]** signals may be asserted by the OnCE control logic to signal that a watchpoint condition has occurred. Watchpoints do not directly cause the CPU to be affected. They are provided to allow external visibility only or for triggering purposes. Watchpoint events are conditioned by the settings in the DBCRxx registers, as well as by the DEVENT register, the DTC/DTSA/DTEA registers, the Performance Monitor control register settings, and generation of MPU debug events. Refer to *Table 1084* for details of the signal assignments. Note that assertion of most watchpoint outputs is conditioned on being in EDM or IDM. The Performance monitor, DEVENT, and DTC watchpoints are not conditioned however, and may assert regardless of the state of EDM or IDM. In addition, DAC1 or DAC3 watchpoints may be generated for stack limit violation occurrences when those resources are configured to perform stack limit checking, regardless of the state of EDM or IDM.

53.5.5.6 Update DR w/go+exit (j_ocmd_go_exit)

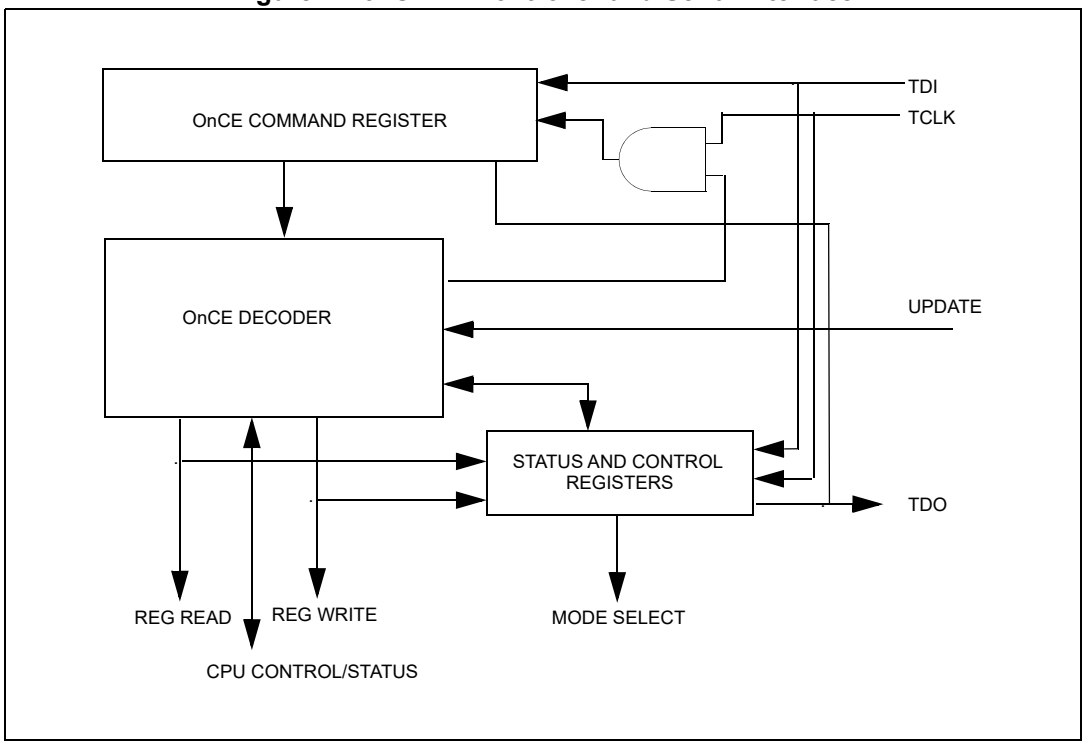
This signal indicates the TAP controller is in the Update_DR state and that the go and exit bits in the OnCE Command register are high, and RS indicates “no register selected”. This signal will assert regardless of whether the CPU is currently in debug mode. It may be

monitored by external logic to cause a synchronous exit from debug mode of other modules (but not other CPUs) in the system. A debugger can place all of the CPU tap controllers in parallel, scan in a go+exit command with “no register selected”, and sequence through the Update_DR state to cause any CPU currently in debug mode to exit. CPUs which are not it debug mode will ignore the command.

53.5.6 OnCE Controller and Serial Interface

The OnCE Controller contains the OnCE command register, the OnCE decoder, and the status/control register. *Figure 1119* is a block diagram of the OnCE controller. In operation, the OnCE Command register acts as the IR for the TAP controller, and all other OnCE resources are treated as data registers (DR) by the TAP controller. The Command register is loaded by serially shifting in commands during the TAP controller Shift-IR state, and is loaded during the Update-IR state. The Command register selects a resource to be accessed as a data register (DR) during the TAP controller Capture-DR, Shift-DR and Update-DR states.

Figure 1119. OnCE Controller and Serial Interface



53.5.6.1 OnCE Status Register

Status information regarding the state of the CPU is latched into the OnCE Status register when the OnCE controller state machine enters the Capture-IR state. When OnCE operation is enabled, this information is provided on the **j_tdo** output in serial fashion when the Shift_IR state is entered following a Capture-IR. Information is shifted out least significant bit first.

Offset: Access: Read-only

	0	1	2	3	4	5	6	7	8	9
R	MCLK	ERR	0	RESET	HALT	STOP	DEBUG	WAIT	0	1
W										

Reset:

Figure 1120. OnCE Status Register

Table 1079 provides bit definitions for the Once Status Register.

Table 1079. OnCE Status Register field descriptions

Field	Description
0 MCLK	**m_clk** Status Bit 0 Inactive state 1 Active state This status bit reflects the logic level on the **jd_mclk_on** input signal after capture by **j_tclk**.
1 ERR	ERROR This bit is used to indicate that an error condition occurred during attempted execution of the last single-stepped instruction (GO+NoExit with CPUSCR or No Register Selected in OCMD), and that the instruction may not have been properly executed. This could occur if an Interrupt (all classes including External, Critical, machine check, Storage and Alignment, Program) occurred while attempting to perform the instruction single step. In this case, the CPUSCR will contain information related to the first instruction of the Interrupt handler, and no portion of the handler will have been executed.
3 RESET	RESET Mode This bit reflects the inverted logic level on the CPU **p_reset_b** input after capture by **j_tclk**.
4 HALT	HALT Mode This bit reflects the logic level on the CPU **p_halted** output after capture by **j_tclk**.
5 STOP	STOP Mode This bit reflects the logic level on the CPU **p_stopped** output after capture by **j_tclk**.
6 DEBUG	Debug Mode This bit is asserted once the CPU is in debug mode. It is negated once the CPU exits debug mode (even during a debug session)
7 WAIT	Waiting Mode This bit reflects the logic level on the CPU **p_waiting** output after capture by **j_tclk**.
8 0	Reserved, set to 0 for 1149.1 compliance
9 1	Reserved, set to 1 for 1149.1 compliance

53.5.6.2 OnCE Command Register (OCMD)

The OnCE Command Register (OCMD) is a 10-bit shift register that receives its serial data from the TDI pin and serves as the instruction register (IR). It holds the 10-bit commands to be used as input for the OnCE Decoder. The Command Register is shown in *Figure 1121*. The OCMD is updated when the TAP controller enters the Update-IR state. It contains fields

for controlling access to a resource, as well as controlling single-step operation and exit from OnCE mode.

Although the OCMD is updated during the Update-IR TAP controller state, the corresponding resource is accessed in the DR scan sequence of the TAP controller, and as such, the Update-DR state must be transitioned through in order for an access to occur. In addition, the Update-DR state must also be transitioned through in order for the single-step and/or exit functionality to be performed, even though the command appears to have no data resource requirement associated with it.

Offset: Access: Read-only

	0	1	2	3	4	5	6	7	8	9
R W	R/W	GO	EX	RS[0:6]						
Reset(1)	—	—	—	—	—	—	—	—	—	—

1. 10'b1000000010 on assertion of **j_trst_b** or **m_por**, or while in the Test_Logic_Reset state.

Figure 1121. OnCE Command Register (OCMD)

Table 1080 provides bit definitions for the Once Command Register.

Table 1080. OnCE Command Register field descriptions

Field	Description
0 R/W	Read/Write Command Bit The R/W bit specifies the direction of data transfer. The table below describes the options defined by the R/W bit. **Note:** The R/W bit generally ignored for read-only or write-only registers, although the PC FIFO pointer is only guaranteed to be update when R/W = 1. In addition, it is ignored for all bypass operations. When performing writes, most registers are sampled in the Capture-DR state into a 32-bit shift register, and subsequently shifted out on **j_tdo** during the first 32 clocks of Shift-DR. 0 Write the data associated with the command into the register specified by RS[0:6] 1 Read the data contained in the register specified by RS[0:6]
1 GO	Go Command Bit If the GO bit is set, and the CPU is currently in debug mode, the CPU will execute the instruction which resides in the IR register in the CPUSCR. To execute the instruction, the processor leaves the debug mode, executes the instruction, and if the EX bit is cleared, returns to the debug mode immediately after executing the instruction. The processor goes on to normal operation if the EX bit is set, and no other debug request source is asserted. The GO command is executed only if the operation is a read/write to CPUSCR or a read/write to "No Register Selected". Otherwise the GO bit is ignored.The processor will leave the debug mode after the TAP controller Update-DR state is entered. On a GO+NoExit operation, returning to debug mode is treated as a debug event, thus exceptions such as machine checks and interrupts may take priority and prevent execution of the intended instruction. Debug firmware should mask these exceptions as appropriate. The OSR_{ERR} bit indicates such an occurrence. If the CPU is not currently in debug mode, then the GO command will be ignored. Note that asynchronous interrupts are blocked on a GO+Exit operation until the first instruction to be executed begins execution. See *Section 53.5.9.6*. 0 Inactive (no action taken) 1 Execute instruction in IR

Table 1080. OnCE Command Register field descriptions (continued)

Field	Description
2 EX	Exit Command Bit 0 Remain in debug mode 1 Leave debug mode If the EX bit is set, the processor will leave the debug mode and resume normal operation until another debug request is generated. The Exit command is executed only if the Go command is issued, and the operation is a read/write to CPUSCR or a read/write to “No Register Selected”. Otherwise the EX bit is ignored. The processor will leave the debug mode after the TAP controller Update-DR state is entered. Note that if the DR bit in the OnCE control register is set or remains set, or if a bit in EDBSR0 is set and $EDBCR0_{EDM}$ = 1 (external debug mode is enabled), or if another debug request source is asserted, then the processor may return to the debug mode *without* execution of an instruction, even though the EX bit was set. Note that asynchronous interrupts are blocked on a GO+Exit operation until the first instruction to be executed begins execution. See *Section 53.5.9.6*.
3:9 RS	Register Select The Register Select bits define which register is source (destination) for the read (write) operation. *Table 1081* indicates the OnCE register addresses. Attempted writes to read-only registers are ignored.

Table 1081 indicates the OnCE register addresses.

Table 1081. OnCE Register Addressing

RS[0:6]	Register Selected
000 0000	Reserved
000 0001	Reserved
000 0010	JTAG ID (read-only)
000 0011–000 1111	Reserved
001 0000	CPU Scan Register (CPUSCR)
001 0001	No Register Selected (Bypass)
001 0010	OnCE Control Register (OCR)
001 0011	Reserved
001 0100–001 0111	Reserved
001 1000	Data Address Compare 3 (DAC3)
001 1001	Data Address Compare 4 (DAC4)
001 1010–001 1111	Reserved
010 0000	Instruction Address Compare 1 (IAC1)
010 0001	Instruction Address Compare 2 (IAC2)
010 0010	Instruction Address Compare 3 (IAC3)
010 0011	Instruction Address Compare 4 (IAC4)
010 0100	Data Address Compare 1 (DAC1)

Table 1081. OnCE Register Addressing (continued)

RS[0:6]	Register Selected
010 0101	Data Address Compare 2 (DAC2)
010 0110	Data Value Compare 1 (DVC1) - *all 64 bits of the DVC register are accessed
010 0111	Data Value Compare 2 (DVC2) - *all 64 bits of the DVC register are accessed
010 1000	Instruction Address Compare 5 (IAC5)
010 1001	Instruction Address Compare 6 (IAC6)
010 1010	Instruction Address Compare 7 (IAC7)
010 1011	Instruction Address Compare 8 (IAC8)
010 1100	Debug Data Effective Address (DDEAR)
010 1101	External Debug Data Effective Address (EDDEAR)
010 1110	External Debug Control Register 0 (EDBCR0)
010 1111	External Debug Status Register 0 (EDBSR0)
011 0000	Debug Status Register (DBSR)
011 0001	Debug Control Register 0 (DBCR0)
011 0010	Debug Control Register 1 (DBCR1)
011 0011	Debug Control Register 2 (DBCR2)
011 0100	Reserved (do not access)
011 0101	Debug Control Register 4 (DBCR4)
011 0110	Debug Control Register 5 (DBCR5)
011 0111	Debug Control Register 6 (DBCR6)
011 1000	Debug Control Register 7 (DBCR7)
011 1001	Debug Control Register 8 (DBCR8)
011 1010	Reserved (do not access)
011 1011	Reserved (do not access)
011 1100	External Debug Status Register MASK 0 (EDBSRMSK0)
011 1101	Debug Data Acquisition Message Register (DDAM)
011 1110	Debug Event Control (DEVENT)
011 1111	External Debug Resource Allocation Control 0 (EDBRAC0)
100 0000	Debug L1 Cache Control/Status 0 (DBL1CCSR0)
100 0001–110 1100	Reserved (do not access)
110 1101	MPU0 Control/Status 0 (MPU0CSR0)
110 1110	Performance Monitor Register Access
110 1111	Reserved for Shared Nexus Control Register Select
111 0000–111 1001	General Purpose register selects [0:9]
111 1010	Cache Debug Access Control Register (CDACNTL)
111 1011	Cache Debug Access Data Register (CDADATA)

Table 1081. OnCE Register Addressing (continued)

RS[0:6]	Register Selected
111 1100	Nexus3-Access (see *Chapter 61: e200z420n3 Nexus 3 Module*)
111 1101	LSRL Select (see Test Specification)
111 1110	Enable_OnCE(1)
111 1111	Bypass

1. Causes assertion of the j_en_once_regsel output.

The Once Decoder receives as input the 10-bit command from the OCMD, and status signals from the processor, and generates all the strobes required for reading and writing the selected OnCE registers.

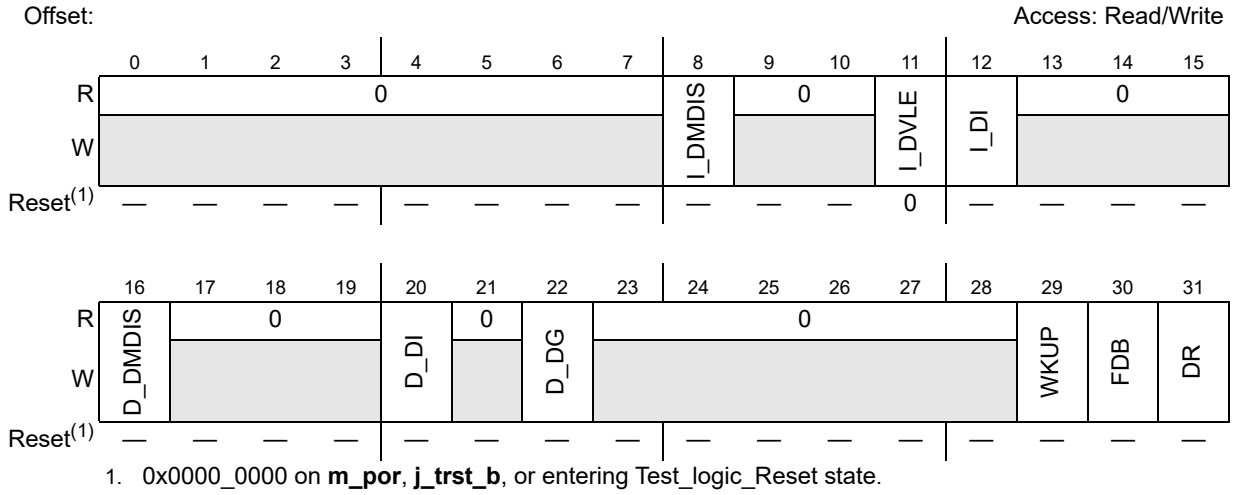
Single stepping of instructions is performed by placing the CPU in debug mode, scanning in appropriate information into the CPUSCR, and setting the Go bit (with the EX bit cleared) with the RS field indicating either the CPUSCR or No Register Selected. After executing a single instruction, the CPU will re-enter debug mode and await further commands. During single-stepping, exception conditions may occur if not properly masked by debug firmware (such as interrupts, machine checks, bus error conditions) and may prevent the desired instruction from being successfully executed. The OSR_{ERR} bit is set to indicate this condition. In these cases, values in the CPUSCR will correspond to the first instruction of the exception handler.

Additionally, the $EDBCR0_{EDM}$ bit is forced to '1' internally while single-stepping to prevent Debug events from generating Debug interrupts. Also, during a debug session, the DBSR register is frozen from updates due to debug events other than execution of a DNI-type instruction, regardless of $EDBCR0_{EDM}$. DBSR may still be modified during a debug session via a single-stepped **mtspr** instruction, or via OnCE access.

If the CPU is not currently in debug mode, go+exit and go+noexit commands are ignored, although for a go+exit command with "No Register Selected", the j_ocmd_go_exit output will be asserted.

53.5.6.3 OnCE Control Register (OCR)

The OnCE Control Register is a 32-bit register used to force the e200z420n3 core into debug mode and to enable / disable sections of the OnCE control logic. It also provides control over the MPU during a debug session (see *Section 53.7*). The control bits are read/write. These bits are only effective while OnCE is enabled (**jd_en_once** asserted). The OCR is shown in *Figure 1122*.



1. 0x0000_0000 on **m_por**, **j_trst_b**, or entering Test_logic_Reset state.

Figure 1122. OnCE Control Register (OCR)

Table 1082 provides bit definitions for the OnCE Control Register.

Table 1082. OnCE Control Register Field descriptions

Field	Description
8 I_DMDIS	Instruction Side Debug MPU Disable Control Bit (I_DMDIS) 0 MPU not disabled for debug sessions 1 MPU disabled for debug sessions This bit may be used to control whether the MPU is enabled normally, or whether the MPU is disabled during a debug session for Instruction Accesses. When enabled, the MPU functions normally. When disabled, for Instruction Accesses, the I bit is taken from the OCR bit I_DI. The SX and UX access permission control bits are set to‘1’ to allow full access. When disabled, no MPU exceptions are generated for Instruction accesses. External access errors can still occur. MPU entries with DEBUG = 1 are not affected by this control bit.
11 I_DVLE	Instruction Side Debug ‘VLE’ Attribute Bit (I_DVLE) This bit is used to provide the ‘VLE’ attribute bit to be used during a debug session. **Note:** This bit is ignored, since VLE is always enabled.
12 I_DI	Instruction Side Debug ‘I’ Attribute Bit (I_DI) This bit is used to provide the ‘I’ attribute bit to be used for Instruction accesses for Instruction accesses during a debug session.
16 D_DMDIS	Data Side Debug MPU Disable Control Bit (D_DMDIS) 0 MPU not disabled for debug sessions 1 MPU disabled for debug sessions This bit may be used to control whether the MPU is enabled normally, or whether the MPU is disabled during a debug session for Data Accesses. When enabled, the MPU functions normally. When disabled, for Data Accesses, the MPU I and G bits are taken from the OCR bits D_DI and D_DG. The SR, SW, UR, and UW access permission control bits are set to‘1’ to allow full access. When disabled, no MPU exceptions are generated for Data accesses. External access errors can still occur. MPU entries with DEBUG = 1 are not affected by this control bit.

Table 1082. OnCE Control Register Field descriptions (continued)

Field	Description
20 D_DI	Data Side Debug 'I' Attribute Bit (D_DI) This bit is used to provide the 'I' attribute bit to be used for Data accesses during a debug session.
22 D_DG	Data Side Debug 'G' Attribute Bit (D_DG) This bit is used to provide the 'G' attribute bit to be used for Data accesses during a debug session.
29 WKUP	Wakeup Request Bit (WKUP) This control bit may be used to force the **p_wakeup** output signal to be asserted. This control function may be used by debug firmware to request that the chip-level clock controller restore the **m_clk** input to normal operation regardless of whether the CPU is in a low power state to ensure that debug resources may be properly accessed by external hardware through scan sequences.
30 FDB	Force Breakpoint Debug Mode Bit (FDB) This control bit is used to determine whether the processor is operating in breakpoint debug enable mode or not. The processor may be placed in breakpoint debug enable mode by setting this bit. In breakpoint debug enable mode, execution of the '**bkpt**' pseudo- instruction will cause the processor to enter debug mode, as if the **jd_de_b** input had been asserted. This bit is qualified with $EDBCR0_{EDM}$, which must be set for FDB to take effect. Note that this bit has no effect on **e_dnh** or **se_dnh** instruction operation.
31 DR	CPU Debug Request Control Bit This control bit is used to unconditionally request the CPU to enter the Debug Mode. The CPU will indicate that Debug Mode has been entered via the data scanned out in the shift-IR state. 0 No Debug Mode request 1 Unconditional Debug Mode request When the DR bit is set the processor will enter Debug mode at the next instruction boundary.

53.5.7 Access to Debug Resources

Resources contained in the OnCE Module which do not require the processor core to be halted for access may be accessed while the e200z420n3 core is running, and will not interfere with processor execution. Accesses to other resources such as the CPUSCR require the e200z420n3 core to be placed in debug mode to avoid synchronization hazards. Debug firmware may ensure that it is safe to access these resources by determining the state of the e200z420n3 core prior to access. Note that a scan operation to update the CPUSCR is required prior to exiting debug mode if debug mode has been entered.

Some cases of write accesses other than accesses to the OnCE Command and Control registers, or the EDM bit of DBCR0 require **m_clk** to be running for proper operation. The OnCE control register provides a means of signaling this need to a system level clock control module.

In addition, since the CPU may cause multiple bits of certain registers to change state, reads of certain registers while the CPU is running (such as DBSR) may not have consistent bit settings unless read twice with the same value indicated. In order to guarantee that the

contents are consistent, the CPU should be placed into debug mode, or multiple reads should be performed until consistent values have been obtained on consecutive reads.

Table 1083 provides a list of access requirements for OnCE registers.

Table 1083. OnCE Register Access Requirements

Register name	Access requirements					Notes
	Requires jd_en_once to be asserted	Requires EDBCR0 $_{EDM}$ = 1 for write access	Requires m_clk active for Write Access	Requires CPU to be halted for Read Access	Requires CPU to be halted for Write Access	
Enable_OnCE	N	N	N	N	—	—
Bypass	N	N	N	N	N	—
CPUSCR	Y	Y	Y	Y	Y	—
DAC1–4	Y	Y	Y	N	*(1)	—
DBCR0	Y	Y	Y	N	*(1)	*EDBCR0$_{EDM}$ access only requires **jd_en_once** asserted
DBCR1–8	Y	Y	Y	N	*(1)	—
DEVENT	Y	Y	Y	N	*(1)	—
EDBRAC0 (DBERC0)	Y	N	Y	N	*(1)	—
DBSR	Y	Y	Y	N(2)	*(1)	—
EDBCR0	Y	N	N	N	N	—
EDBSR0	Y	N	Y	N	N	—
EDBSRMSK0	Y	N	N	N	N	—
IAC1–8	Y	Y	Y	N	*(1)	—
JTAG ID	N	N	—	N	—	Read-only
MPU0CSR0	Y	N	Y	N	N	—
OCR	Y	N	N	N	N	—
OSR	Y	N	—	N	—	Read-only, accessed by scanning out IR while **jd_en_once** is asserted
Cache Debug Access Control (CDACNTL)	Y	N	Y	N	N	CPU gives lowest priority to a Cache access request when it is not debug halted
Cache Debug Access Data (CDADATA)	Y	N	Y	N	N	CPU gives lowest priority to a Cache access request when it is not debug halted
Nexus3-Access	Y	N	N	N	N	—
PMR-Access	Y	N	N	N	N	—
PMR Registers	Y	Y	Y	N	N	—

Table 1083. OnCE Register Access Requirements (continued)

Register name	Access requirements					Notes
	Requires jd_en_once to be asserted	Requires EDBCR0 EDM = 1 for write access	Requires m_clk active for Write Access	Requires CPU to be halted for Read Access	Requires CPU to be halted for Write Access	
External GPRs	Y	N	N	N	N	—
LSRL Select	Y	N	?	?	?	System Test logic implementation determines LSRL functionality

1. Writes to these registers while the CPU is running may have unpredictable results due to the pipelined nature of operation, and the fact that updates are not synchronized to a particular clock, instruction, or bus cycle boundary, therefore it is strongly recommended to ensure the processor is first placed into debug mode before updates to these registers are performed.
2. Reads of these registers while the CPU is running may not give data that is self-consistent due to synchronization across clock domains.

53.5.8 Methods of Entering Debug Mode

The OnCE Status Register indicates that the CPU has entered the debug mode via the DEBUG status bit. The following sections describe how Debug Mode is entered assuming the OnCE circuitry has been enabled. OnCE operation is enabled by the assertion of the **jd_en_once** input See *Section 53.5.5.1*.

53.5.8.1 External Debug Request During RESET

Holding the **jd_de_b** signal asserted during the assertion of **p_reset_b**, and continuing to hold it asserted following the negation of **p_reset_b** causes the e200z420n3 core to enter Debug Mode. After receiving an acknowledge via the OnCE Status Register DEBUG bit, the external command controller should negate the **jd_de_b** signal before sending the first command. Note that in this case the e200z420n3 core does not execute an instruction before entering Debug Mode, although the first instruction to be executed will be fetched prior to entering Debug Mode in order to properly initialize the CPUSCR.

53.5.8.2 Debug Request During RESET

Asserting a debug request by setting the DR bit in the OCR during the assertion of **p_reset_b** and then negating **p_reset_b** causes the chip to enter debug mode. In this case the CPU will fetch the first instruction of the reset exception handler in order to properly initialize the CPUSCR, but does not execute an instruction before entering debug mode.

53.5.8.3 Debug Request During Normal Activity

Asserting a debug request by setting the DR bit in the OCR during normal chip activity causes the chip to finish the execution of the current instruction and then enter the debug mode. Note that in this case the chip completes the execution of the current instruction and stops after the newly fetched instruction enters the CPU instruction register. This process is the same for any newly fetched instruction including instructions fetched by the interrupt processing, or those that will be aborted by the interrupt processing.

53.5.8.4 Debug Request During Waiting, Halted or Stopped State

Asserting a debug request by setting the DR bit in the OCR when the chip is in the Waiting state (**p_waiting** asserted), Halted state (**p_halted** asserted) or Stopped state (**p_stopped** asserted) causes the CPU to exit the state and enter the debug mode once the CPU clock **m_clk** has been restored. Note that in this case, the CPU will negate the **p_waiting, p_halted** and **p_stopped** outputs. Once the debug session has ended, the CPU will return to the state it was in prior to entering debug mode.

To signal the chip-level clock generator to re-enable **m_clk**, the **p_wakeup** output will be asserted whenever the debug block is asserting a debug request to the CPU due to OCR_{DR} being set, or **jd_de_b** assertion, and will remain set from then until the debug session ends (**jd_debug_b** goes from asserted to negated). In addition, the status of the **jd_mclk_on** input (after synchronization to the **j_tclk** clock domain) may be sampled along with other status bits from the **j_tdo** output during the Shift_IR TAP controller state. This status may be used if necessary by external debug firmware to ensure proper scan sequences occur to registers in the **m_clk** clock domain.

53.5.8.5 Software Request During Normal Activity

Upon executing a '**bkpt'** pseudo-instruction (for e200z420n3, defined to be an all 0's instruction opcode) when the OCR register's (FDB) bit is set (debug mode enable control bit is true), and $EDBCR0_{EDM}$ = 1, the CPU enters the debug mode after the instruction following the '**bkpt'** pseudo-instruction has entered the instruction register.

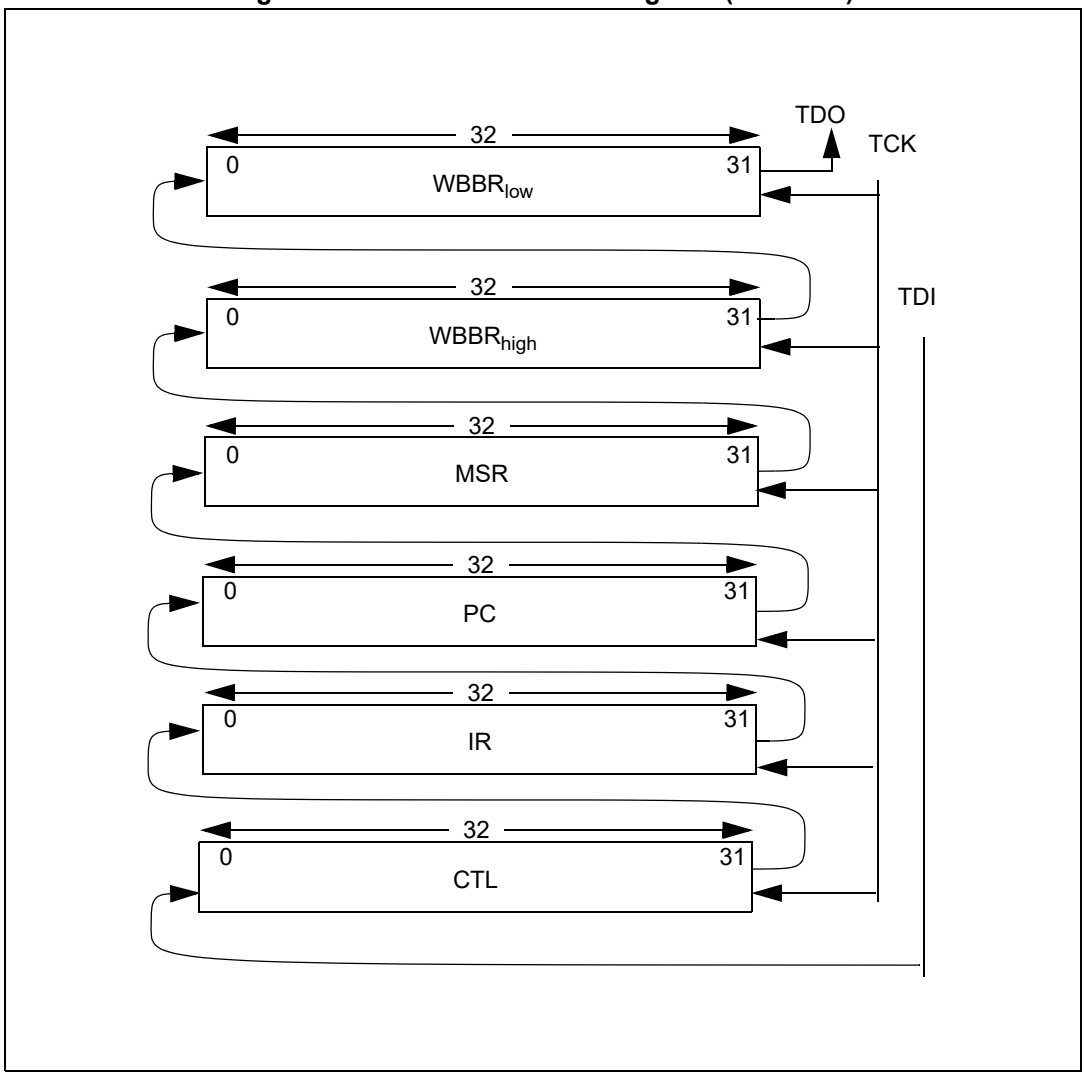
53.5.8.6 Debug Notify Halt Instructions

The **e_dnh** and **se_dnh** instructions allow software to transition the core from a running state to a debug halted state if enabled by $EDBCR0_{DNH_EN}$, and provide the external debugger with bits reserved in the instruction itself to pass additional information. Entry into debug mode is ***not*** conditioned on $EDBCR0_{EDM}$, allowing for debug of software debug handlers as well as other software. For e200z420n3, when the CPU enters a debug halted state due to a **e_dnh** or **se_dnh** instruction, the instruction will be stored in the CPUSCR[IR] portion, and the CPUSCR[PC] value will point to the instruction. The external debugger should update the CPUSCR prior to exiting the debug halted state to point past the **e_dnh** or **se_dnh** instruction.

53.5.9 CPU Status and Control Scan Chain Register (CPUSCR)

A number of on-chip registers store the CPU pipeline status and are configured in a single scan chain for access by the OnCE controller. The CPUSCR register contains these processor resources, which are used to restore the pipeline and resume normal chip activity upon return from the debug mode, as well as a mechanism for the emulator software to access processor and memory contents. *Figure 1123* shows the block diagram of the pipeline information registers contained in the CPUSCR. Once debug mode has been entered, it is required to scan in and update this register prior to exiting debug mode.

Figure 1123. CPU Scan Chain Register (CPUSCR)



53.5.9.1 Instruction Register (IR)

The Instruction Register (IR) provides a mechanism for controlling the debug session by serving as a means for forcing in selected instructions, and then causing them to be executed in a controlled manner by the debug control block. The opcode of the next instruction to be executed when entering debug mode is contained in this register when the scan-out of this chain begins. This value should be saved for later restoration if continuation of the normal instruction stream is desired.

On scan-in, in preparation for exiting debug mode, this register is filled with an instruction opcode selected by debug control software. By selecting appropriate instructions and controlling the execution of those instructions, the results of execution may be used to examine or change memory locations and processor registers. The debug control module external to the processor core will control execution by providing a single-step capability. Once the debug session is complete and normal processing is to be resumed, this register may be loaded with the value originally scanned out.

53.5.9.2 Control State Register (CTL)

The Control State Register (CTL) is a 32-bit register that stores the value of certain internal CPU state variables before the debug mode is entered. This register is affected by the operations performed during the debug session and should normally be restored by the external command controller when returning to normal mode. In addition to saved internal state variables, two of the bits are used by emulation firmware to control the debug process. In certain circumstances, emulation firmware must modify the content of this register as well as the PC and IR values in the CPUSCR before exiting debug mode. These cases are described below.

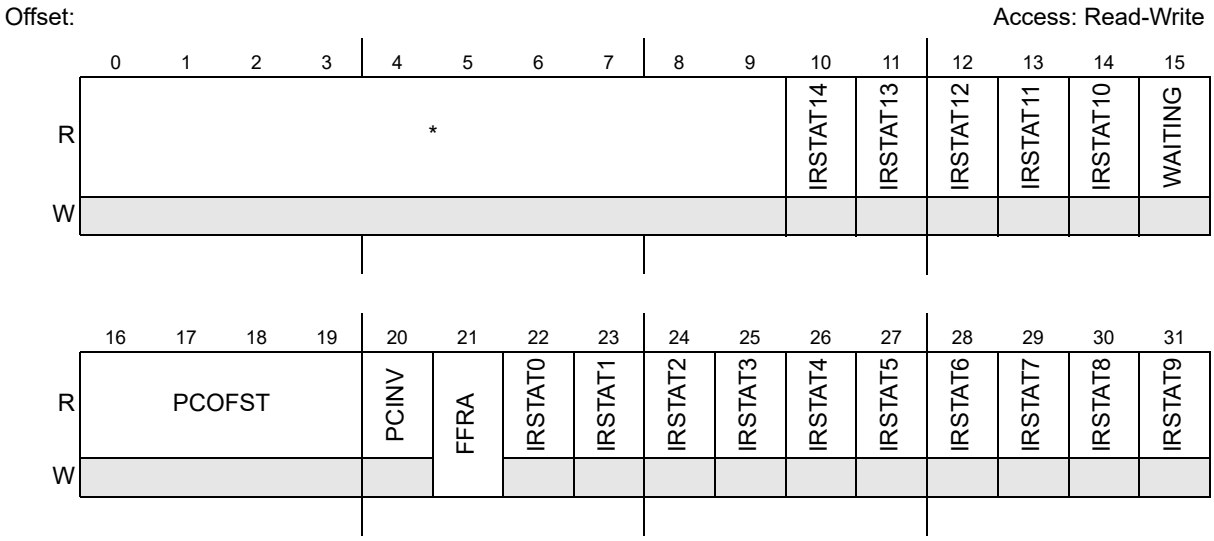


Figure 1124. Control State Register (CTL)

53.5.9.2.1 WAITING: WAITING State Status

This bit indicates whether the CPU was in the waiting state prior to entering debug mode. If set, the CPU was in the waiting state. Upon exiting a debug session, the value of this bit in the restored CPUSCR will determine whether the CPU re-enters the waiting state on a go+exit.

0 CPU was not in the waiting state when debug mode was entered

1 CPU was in the waiting state when debug mode was entered

53.5.9.2.2 PCOFST: PC Offset Field

This field indicates whether the value in the PC portion of the CPUSCR must be adjusted prior to exiting debug mode. Due to the pipelined nature of the CPU, the PC value must be backed-up by emulation software in certain circumstances. The PCOFST field specifies the value to be subtracted from the original value of the PC. This adjusted PC value should be restored into the PC portion of the CPUSCR just prior to exiting debug mode with a go+exit. In the event the PCOFST is non-zero, the IR should be loaded with a nop instruction instead

of the original IR value, other wise the original value of IR should be restored. (But see PCINV which overrides this field)

0000 No correction required.

0001 Subtract 0x04 from PC.

0010 Subtract 0x08 from PC.

0011 Subtract 0x0C from PC.

0100 Subtract 0x10 from PC.

0101 Subtract 0x14 from PC.

all other encodings are reserved

53.5.9.2.3 *: Internal State Bits

These control bits represent internal processor state and should be restored to their original value after a debug session is completed, that is when a OnCE command is issued with the GO and EX bits set and not ignored. When performing instruction execution during a debug session. See *Section 53.5.5.3*, which is not part of the normal program execution flow, these bits should be set to a 0.

53.5.9.2.4 PCINV: PC and IR Invalid Status Bit

This status bit indicates that the values in the IR and PC portions of the CPUSCR are invalid. Exiting debug mode with the saved values in the PC and IR will have unpredictable results. Debug firmware should initialize the PC and IR values in the CPUSCR with desired values prior to exiting debug mode if this bit was set when debug mode was initially entered.

0 No error condition exists.

1 Error condition exists. PC and IR are corrupted.

53.5.9.2.5 FFRA: Feed Forward RA Operand Bit

This control bit causes the content of the $WBBR_{low}$ to be used as the RA operand value (RS for logical, **mtspr**, **mtdcr**, **cntlzw**, and shift operations, RX for VLE **se_** instructions, RT for **e_{logical_op}2i** type instructions, RB for **evaddiw**, **evsubifw**, and the value to use as the PC for calculating the LR update value for branch with link type instructions) of the first instruction to be executed following an update of the CPUSCR.

This allows the debug firmware to update processor registers — initialize the $WBBR_{low}$ with the desired value, set the FFRA bit, and execute an **ori** Rx,Rx,0 instruction to the desired register.

*Note: Not all instructions support using the FFRA control. FFRA is mainly intended for use with the **ori** instruction to allow the debugger to write to a GPR. Support for other instructions is implementation-dependent.*

0 No action.

1 Content of $WBBR_{low}$ used as operand value

53.5.9.2.6 IRStat0: IR Status Bit 0

This control bit indicates a TEA status for the IR.

0 No TEA occurred on the fetch of this instruction.

1 TEA occurred on the fetch of this instruction.

53.5.9.2.7 IRStat1: IR Status Bit 1

This control bit is reserved.

53.5.9.2.8 IRStat2: IR Status Bit 2

This control bit indicates an Instruction Address Compare 1 event status for the IR.

0 No Instruction Address Compare event 1 occurred on the fetch of this instruction.

1 An Instruction Address Compare event 1 occurred on the fetch of this instruction.

53.5.9.2.9 IRStat3: IR Status Bit 3

This control bit indicates an Instruction Address Compare 2 event status for the IR.

0 No Instruction Address Compare 2 event occurred on the fetch of this instruction.

1 An Instruction Address Compare 2 event occurred on the fetch of this instruction.

53.5.9.2.10 IRStat4: IR Status Bit 4

This control bit indicates an Instruction Address Compare 3 event status for the IR.

0 No Instruction Address Compare event 3 occurred on the fetch of this instruction.

1 An Instruction Address Compare event 3 occurred on the fetch of this instruction.

53.5.9.2.11 IRStat5: IR Status Bit 5

This control bit indicates an Instruction Address Compare 4 event status for the IR.

0 No Instruction Address Compare 4 event occurred on the fetch of this instruction.

1 An Instruction Address Compare 4 event occurred on the fetch of this instruction.

53.5.9.2.12 IRStat6: IR Status Bit 6

This control bit indicates a Parity Error status for the IR.

0 No Parity Error occurred on the fetch of this instruction.

1 Parity Error occurred on the fetch of this instruction from the ICache or the IMEM.

53.5.9.2.13 IRStat7: IR Status Bit 7

This control bit indicates a Precise External Termination Error status for the IR.

0 No Precise External Termination Error occurred on the fetch of this instruction.

1 A Precise External Termination Error occurred on the fetch of this instruction.

53.5.9.2.14 IRStat8: IR Status Bit 8

This control bit indicates the PowerISA VLE status for the IR.

0 IR contains a BookE instruction. (unused encoding)

1 IR contains a PowerISA VLE instruction, aligned in the Most Significant Portion of IR if 16-bit.

– This bit should not be modified by the debugger, otherwise the resulting operation is boundedly undefined. It should always indicate VLE (= 1).

53.5.9.2.15 IRStat9: IR Status Bit 9

This control bit is reserved.

53.5.9.2.16 IRStat10: IR Status Bit 10

This control bit indicates an Instruction Address Compare 5 event status for the IR.

0 No Instruction Address Compare 5 event occurred on the fetch of this instruction.

1 An Instruction Address Compare 5 event occurred on the fetch of this instruction.

53.5.9.2.17 IRStat11: IR Status Bit 11

This control bit indicates an Instruction Address Compare 6 event status for the IR.

0 No Instruction Address Compare 6 event occurred on the fetch of this instruction.

1 An Instruction Address Compare 6 event occurred on the fetch of this instruction.

53.5.9.2.18 IRStat12: IR Status Bit 12

This control bit indicates an Instruction Address Compare 7 event status for the IR.

0 No Instruction Address Compare 7 event occurred on the fetch of this instruction.

1 An Instruction Address Compare 7 event occurred on the fetch of this instruction.

53.5.9.2.19 IRStat13: IR Status Bit 13

This control bit indicates an Instruction Address Compare 8 event status for the IR.

0 No Instruction Address Compare 8 event occurred on the fetch of this instruction.

1 An Instruction Address Compare 8 event occurred on the fetch of this instruction.

53.5.9.2.20 IRStat14: IR Status Bit 14

This control bit indicates an MPU Instruction Address Compare event status for the IR.

0 No MPU Instruction Address Compare event occurred on the fetch of this instruction.

1 An MPU Instruction Address Compare event occurred on the fetch of this instruction.

Emulation firmware should modify the content of the CTL, PC, and IR values in the CPUSCR during execution of debug related instructions as well as just prior to exiting debug with a go+exit command. During the debug session, the CTL register should be written with the FFRA bit set as appropriate, and all other bit set to ‘0’, and the IR set to the value of the desired instruction to be executed. IRStat8 will be used to determine the type of instruction present in the IR.

Just prior to exiting debug mode with a go+exit, the PCINV status bit which was originally present when debug mode was first entered should be tested, and if set, the PC and IR initialized for performing whatever recovery sequence is appropriate for a faulted exception vector fetch. If the PCINV bit is cleared, then the PCOFST bits should be examined to determine whether the PC value must be adjusted. Due to the pipelined nature of the CPU, the PC value must be backed-up by emulation software in certain circumstances. The PCOFST field specifies the value to be subtracted from the original value of the PC. This adjusted PC value should be restored in to the PC portion of the CPUSCR just prior to exiting debug mode with a go+exit. In the event the PCOFST is non-zero, the IR should be loaded with a nop instruction (such as **ori r0,r0,0**) instead of the original IR value, otherwise the original value of IR should be restored. Note that when a correction is made to the PC value, it will generally point to the last completed instruction, although that instruction will not be re-executed. The nop instruction is executed instead, and instruction fetch and execution will resume at location PC+4. IRStat8 will be used to determine the type of instruction

present in the IR, thus should be cleared in this case. Note that debug events which may occur on the nop (ICMP) will be generated if enabled.

For the CTL register, the internal state bits should be restored to their original value. The IRStatus bits should be set to '0's if the PC was adjusted. If no PC adjustment was performed, emulation firmware should determine whether other IRStat flags should be set to '0' to avoid re-entry into debug mode for an instruction breakpoint request. Upon exiting debug mode with go+exit, if one of these bits is set, debug mode will be re-entered prior to any further instruction execution.

53.5.9.3 Program Counter Register (PC)

The PC is a 32-bit register that stores the value of the program counter which was present when the chip entered the debug mode. It is affected by the operations performed during the debug mode and must be restored by the external command controller when the CPU returns to normal mode. PC normally points to the instruction contained in the IR portion of CPUSCR. If debug firmware wishes to redirect program flow to an arbitrary location, the PC and IR should be initialized to correspond to the first instruction to be executed upon resumption of normal processing. Alternatively, the IR may be set to a nop and the PC set to point to the location prior to the location at which it is desired to redirect flow to. On exiting debug mode, the nop will be executed, and instruction fetch and execution will resume at PC+4.

53.5.9.4 Write-Back Bus Register ($WBBR_{low}$, $WBBR_{high}$)

WBBR is used as a means of passing operand information between the CPU and the external command controller. Whenever the external command controller needs to read the contents of a register or memory location, it will force the chip to execute an instruction that brings that information to WBBR. $WBBR_{low}$ holds the 32-bit result of most instructions including load data returned for a load or load with update instruction. $WBBR_{high}$ holds the updated effective address calculated by a load with update instruction. It is undefined for other instructions.

As an example, to read the 32 bits of processor register **r1**, an **ori r1,r1,0** instruction is executed, and the result value of the instruction will be latched into $WBBR_{low}$. The contents of $WBBR_{low}$ can then be delivered serially to the external command controller. To update a processor resource, this register is initialized with a data value to be written, and an **ori** instruction is executed which uses this value as a substitute data value. The Control State register FFRA bit forces the value of the $WBBR_{low}$ to be substituted for the normal RS source value of the **ori** instruction, thus allowing updates to processor registers to be performed. (Refer to *Section 53.5.9.2* for more detail on the CTL_{FFRA} bit.)

$WBBR_{low}$ and $WBBR_{high}$ are generally undefined on instructions which do not writeback a result, and due to control issues are not defined on **lmw** or branch instructions as well.

53.5.9.5 Machine State Register (MSR)

The MSR is a 32-bit register used to read/write the Machine State Register. Whenever the external command controller needs to save or modify the contents of the Machine State Register, this register is used.This register is affected by the operations performed during the debug mode and must be restored by the external command controller when returning to normal mode.

53.5.9.6 Exiting Debug Mode and Interrupt Blocking

When exiting debug mode with a Go+Exit, “asynchronous” interrupts are blocked until the first instruction to be executed begins execution. This includes External and Critical input, NMI, and machine check interrupts. Asynchronous debug interrupts are not blocked however, and the CPU will re-enter debug mode without executing an instruction following Go+Exit, although it may fetch an instruction and discard it. Exceptions due to an illegal instruction or error flags set within the CPUSCR CTL register are not blocked, since they apply to the instruction in the CPUSCR IR.

53.5.10 Reserved Registers (Reserved)

The Reserved Registers are used to control various test control logic. These registers are not intended for customer use. To preclude device and/or system damage, these registers should not be accessed.

53.6 Watchpoint Support

e200z420n3 supports the generation and signalling of watchpoints when operating in internal debug mode ($DBCR0_{IDM}$ = 1) or in external debug mode ($EDBCR0_{EDM}$ = 1). Watchpoints are indicated with a dedicated set of interface signals. The **jd_watchpt[0:31]** output signals are used to indicate that a watchpoint has occurred. Certain watchpoints however (DEVENT-based, DTC-based, and Performance Monitor watchpoints) are not qualified with $EDBCR0_{EDM}$ or $DBCR0_{IDM}$.

Each debug address compare function (IAC1–8, DAC1–4), as well as other event types are capable of triggering a watchpoint output. The DBCRx control fields are used to configure watchpoints, regardless of whether events are enabled in DBCR0. Watchpoints may occur whenever an associated event would have been posted in the Debug Status Register if enabled. No explicit enable bits are provided for watchpoints; they are always enabled by definition, although the data address compare watchpoints may be controlled for read and write accesses via configuration fields in DBCR4, 7, and 9. During a debug session, debug events (other than PMI, DEVT1 and DEVT2) with a corresponding DBSR bit are blocked from asserting a watchpoint. The Performance Monitor, DEVNT-based and DTC-based watchpoints are not blocked during a debug session. If not desired, for address-based events the base address values for these events may be programmed to an unused system address. MSR_{DE} has no effect on watchpoint generation.

External logic may monitor the assertion of these signals for debugging or triggering purposes. Watchpoints are signaled in the clock cycle following the occurrence of the actual event. The Nexus3 module also monitors assertion of a portion of these output signals (**jd_watchpt[0:31])** for various development control purposes. (See the “Watchpoint Trace Messaging” section in the Core (e200z420n3) Nexus 3 Module chapter.)

Note that these signals are **m_clk** domain signals, not **j_tclk** domain signals.

Table 1084 shows the **jd_watchpt[0:31]** output signal assignments to various debug events.

Table 1084. Watchpoint Output Signal Assignments

Signal Name	Type	Description
jd_watchpt[0]	IAC1	Instruction Address Compare 1 watchpoint Asserted whenever an IAC1 compare occurs regardless of being enabled to set DBSR status
jd_watchpt[1]	IAC2	Instruction Address Compare 2 watchpoint Asserted whenever an IAC2 compare occurs regardless of being enabled to set DBSR status
jd_watchpt[2]	IAC3	Instruction Address Compare 3 watchpoint Asserted whenever an IAC3 compare occurs regardless of being enabled to set DBSR status
jd_watchpt[3]	IAC4	Instruction Address Compare 4 watchpoint Asserted whenever an IAC4 compare occurs regardless of being enabled to set DBSR status
jd_watchpt[4]	DAC1	Data Address Compare 1 watchpoint Asserted whenever a DAC1 compare occurs regardless of being enabled to set DBSR status, based on $DBCR4_{DAC1CFG}$ settings
jd_watchpt[5]	DAC2	Data Address Compare 2 watchpoint Asserted whenever a DAC2 compare occurs regardless of being enabled to set DBSR status, based on $DBCR4_{DAC2CFG}$ settings
jd_watchpt[6]	DAC3	Data Address Compare 3 watchpoint Asserted whenever a DAC3 compare occurs regardless of being enabled to set DBSR status, based on $DBCR7_{DAC3CFG}$ settings
jd_watchpt[7]	DAC4	Data Address Compare 4 watchpoint Asserted whenever a DAC4 compare occurs regardless of being enabled to set DBSR status, based on $DBCR7_{DAC4CFG}$ settings
jd_watchpt[8]	IAC5	Instruction Address Compare 5 watchpoint Asserted whenever an IAC5 compare occurs regardless of being enabled to set DBSR status
jd_watchpt[9]	IAC6	Instruction Address Compare 6 watchpoint Asserted whenever an IAC6 compare occurs regardless of being enabled to set DBSR status
jd_watchpt[10]	DEVT1	Debug Event Input 1 watchpoint Asserted whenever a DEVT1 debug event occurs regardless of being enabled to set DBSR status
jd_watchpt[11]	DEVT2	Debug Event Input 2 watchpoint Asserted whenever a DEVT2 debug event occurs regardless of being enabled to set DBSR status
jd_watchpt[12]	DEVNT0	Debug Event Output 0 watchpoint Asserted whenever a ‘1’ is written to the bit of the DEVNT field of the DEVENT debug register corresponding to jd_watchpt[12]
jd_watchpt[13]	DEVNT1	Debug Event Output 1 watchpoint Asserted whenever a ‘1’ is written to the bit of the DEVNT field of the DEVENT debug register corresponding to jd_watchpt[13]

Table 1084. Watchpoint Output Signal Assignments (continued)

Signal Name	Type	Description
jd_watchpt[14]	IAC7	Instruction Address Compare 7 watchpoint Asserted whenever an IAC7 compare occurs regardless of being enabled to set DBSR status
jd_watchpt[15]	IAC8	Instruction Address Compare 8 watchpoint Asserted whenever an IAC8 compare occurs regardless of being enabled to set DBSR status
jd_watchpt[16]	IRPT	Interrupt watchpoint Asserted whenever an IRPT debug event occurs regardless of being enabled to set DBSR status
jd_watchpt[17]	RET	Return watchpoint Asserted whenever a RET debug event occurs regardless of being enabled to set DBSR status
jd_watchpt[18]	CIRPT	Critical Interrupt watchpoint Asserted whenever a CIRPT debug event occurs regardless of being enabled to set DBSR status
jd_watchpt[19]	CRET	Critical Return watchpoint Asserted whenever a CRET debug event occurs regardless of being enabled to set DBSR status
jd_watchpt[20]	DEVNT2	Debug Event Output 2 watchpoint Asserted whenever a ‘1’ is written to the bit of the DEVNT field of the DEVENT debug register corresponding to jd_watchpt[20]
jd_watchpt[21]	DEVNT3	Debug Event Output 3 watchpoint Asserted whenever a ‘1’ is written to the bit of the DEVNT field of the DEVENT debug register corresponding to jd_watchpt[21]
jd_watchpt[22]	PMEVENT	Performance Monitor Event input watchpoint Asserted whenever **p_pm_event** transitions from a ‘0’ to a ‘1’ while **m_clk** is running
jd_watchpt[23]	PMC0	Performance Monitor Counter 0 watchpoint Asserted whenever PMC0 triggers an event based on $PMLCa0_{PMP}$
jd_watchpt[24]	PMC1	Performance Monitor Counter 1 watchpoint Asserted whenever PMC1 triggers an event based on $PMLCa1_{PMP}$
jd_watchpt[25]	PMC2	Performance Monitor Counter 2 watchpoint Asserted whenever PMC2 triggers an event based on $PMLCa2_{PMP}$
jd_watchpt[26]	PMC3	Performance Monitor Counter 3 watchpoint Asserted whenever PMC3 triggers an event based on $PMLCa3_{PMP}$
jd_watchpt[27]	MPU	Memory Protection Unit watchpoint Asserted whenever a the MPU generates a debug event based on region descriptor matches with DEBUG control enabled.
jd_watchpt[28]	TRAP	TRAP watchpoint Asserted whenever an TRAP debug event occurs regardless of being enabled to set DBSR status

Table 1084. Watchpoint Output Signal Assignments (continued)

Signal Name	Type	Description
jd_watchpt[29]	DTC1	Data Trace Control Range 1 watchpoint Asserted whenever an access meets the conditions for DTC Range 1
jd_watchpt[30]	DTC2	Data Trace Control Range 2 watchpoint Asserted whenever an access meets the conditions for DTC Range 2
jd_watchpt[31]	DTC3	Data Trace Control Range 3 watchpoint Asserted whenever an access meets the conditions for DTC Range 3

53.7 MPU Operation During Debug

A debug session begins when the CPU initially enters debug mode, and ends when a OnCE command with GO+EXIT is executed, releasing the CPU for normal operation. If desired during a debug session, the debug firmware may substitute default values obtained from the OnCE Control Register for Access Attribute (I, G) bits. Refer to the bit definitions in the OCR (*Section 53.5.6.3* for more detail.

Normal operation of the MPU may be modified during a 'debug session' via the OnCE Control Register (OCR). A debug session begins when the CPU initially enters debug mode, and ends when a OnCE command with GO+EXIT is executed, releasing the CPU for normal operation. If desired during a debug session, the debug firmware may disable the MPU protection mechanism and may substitute default values for the Access Protection (UX, UR, UW, SX, SR, SW) bits, and values obtained from the OnCE Control Register for Memory Attributes (I, G) bits normally provided by a matching MPU entry.

When disabled during a debug session, no MPU-related storage interrupt conditions will occur. If the debugger desires to use the normal protection mechanism, the MPU may be left enabled in the OnCE OCR, and normal protections provided by the MPU (including the possibility of a storage interrupt) will remain in effect.

The OCR control bits are used when debug mode is entered. Refer to the bit definitions in the OCR (*Section 53.5.6.3* for more detail. When the MPU is disabled for instruction accesses (OCR_{I_DMDIS}) or for data accesses (OCR_{D_DMDIS}), substituted access attribute bits will control operation on respective accesses initiated during debug.

53.8 Cache Array Access During Debug

The cache arrays may be read and written during debug mode via the CDACNTL and CDADATA debug registers.

53.9 Basic Steps for Enabling, Using, and Exiting External Debug Mode

The following steps show one possible scenario for a debugger wishing to use the external debug facilities. *This simplified flow is intended to illustrate basic operations, but does not cover all potential methods in depth.*

Enabling External Debug Mode and initializing Debug registers

- The debugger should ensure that the **jd_en_once** control signal is asserted in order to enable OnCE operation
- Select the OCR and write a value to it in which OCR_{DR}, OCR_{WKUP}, are set to '1'. The tap controller must step through the proper states as outlined earlier. This step will place the CPU in a debug state in which it is halted and awaiting single-step commands or a release to normal mode
- Scan out the value of the OSR to determine that the CPU clock is running and the CPU has entered the Debug state. This can be done in conjunction with a Read of the CPUSCR. The OSR is shifted out during the Shift_IR state. The CPUSCR will be shifted out during the Shift_DR state. The debugger should save the scanned-out value of CPUSCR for later restoration.
- Select the DBCR0 register and update it with the $EDBCR0_{EDM}$ bit set
- Clear the DBSR status bits
- Write appropriate values to the DBCR0–8, IAC, and DAC registers. Note that the initial write to DBCR0 will only affect the EDM bit, so the remaining portion of the register must now be initialized, keeping the EDM bit set

At this point the system is ready to commence debug operations. Depending on the desired operation, different steps must occur.

- Optionally, ensure that the values entered into the MSR portion of the CPUSCR during the following steps cause interrupt to be disabled (clearing MSR_{EE} and MSR_{CE}). This will ensure that external interrupt sources do not cause single-step errors.

To single-step the CPU:

- debugger scans in either a new or a previously saved value of the CPUSCR (with appropriate modification of the PC and IR as described in *Section 53.5.9.2*), with a Go+Noexit OnCE Command value.
- The debugger scans out the OSR with "no-register selected", Go cleared, and determines that the PCU has re-entered the Debug state and that no ERR condition occurred

To return the CPU to normal operation (without disabling external debug mode):

- The OCR_{DR}, control bit should be cleared, leaving the OCR_{WKUP} bit set
- The debugger restores the CPUSCR with a previously saved value of the CPUSCR (with appropriate modification of the PC and IR as described in *Section 53.5.9.2*), with a Go+Exit OnCE Command value.
- The OCR_{WKUP} bit may then be cleared

To exit External Debug Mode:

- The debugger should place the CPU in the debug state via the OCR_{DR} with OCR_{WKUP} asserted, scanning out and saving the CPUSCR
- The debugger should write the DBCR0–8 registers as needed, likely clearing every enable except the $EDBCR0_{EDM}$ bit
- The debugger should write the DBSR to a cleared state
- The debugger should re-write the DBCR0 with all bits including EDM cleared
- The debugger should clear the OCR_{DR} bit
- The debugger restores the CPUSCR with the previously saved value of the CPUSCR (with appropriate modification of the PC and IR as described in *Section 53.5.9.2*), with a Go+Exit OnCE Command value.
- The OCR_{WKUP} bit may then be cleared

Note: These steps are meant by way of examples, and are not meant to be an exact template for debugger operation.

54 Debug and Calibration Interface (DCI)

54.1 Introduction

The Debug and Calibration Interface (DCI) module provides debug and calibration features for the MCU. It includes a standard 1149.1/1149.7 compatible JTAG interface which is used to connect with an external JTAG tool using a low frequency clock interface.

It also features a software based debug mode which can be controlled by the MCU without using an external tool. Additionally, the DCI provides features for debug control using break-points and synchronous restart of the cores. This chapter describes the DCI features for the Production device (PD) only.

54.1.1 Features

The DCI features are the following:

- Debug mode enable control for connected tool or software
- 1149.1 and 1149.7 controllers
- Debug break and cross-triggering control
- Synchronous restart control for all CPUs when exiting debug mode
- Tool hot plug capability
- Security access control
- Debug reset control

54.1.2 Overview

The DCI can take the following different inputs for its JTAG signals:

- P.JTAG—JTAG signals coming from the Production device JTAG pads. This is the standard 5-pin JTAG interface and is the default operating mode.
- M.JTAG—JTAG signals generated by JTAGM module using software debug mode.
  - In software debug mode, there is no tool, and debug software running on the MCU generates the JTAG packets and writes them to JTAGM using special debug registers in the JTAGM module.

By default the DCI operates in standard 5-pin JTAG mode (IEEE 1149.1). It can be configured in 3-pin reduced pin JTAG mode (IEEE 1149.7) after starting from standard 5-pin JTAG mode.

When reduced pin JTAG mode (IEEE 1149.7) is configured TMSC pin is configured in bidirectional mode during operation.

The DCI also provides a centralized break control for system IPs and cores, which can be controlled by an external tool as well as the internal debug peripherals such as SPU. The DCI also provides the Event Out ($\overline{\text{EVTO}}$) signal to the debug tool in case of the occurrence of any internal debug event.

Figure 1125 shows the DCI block diagram.

Figure 1125. DCI block diagram

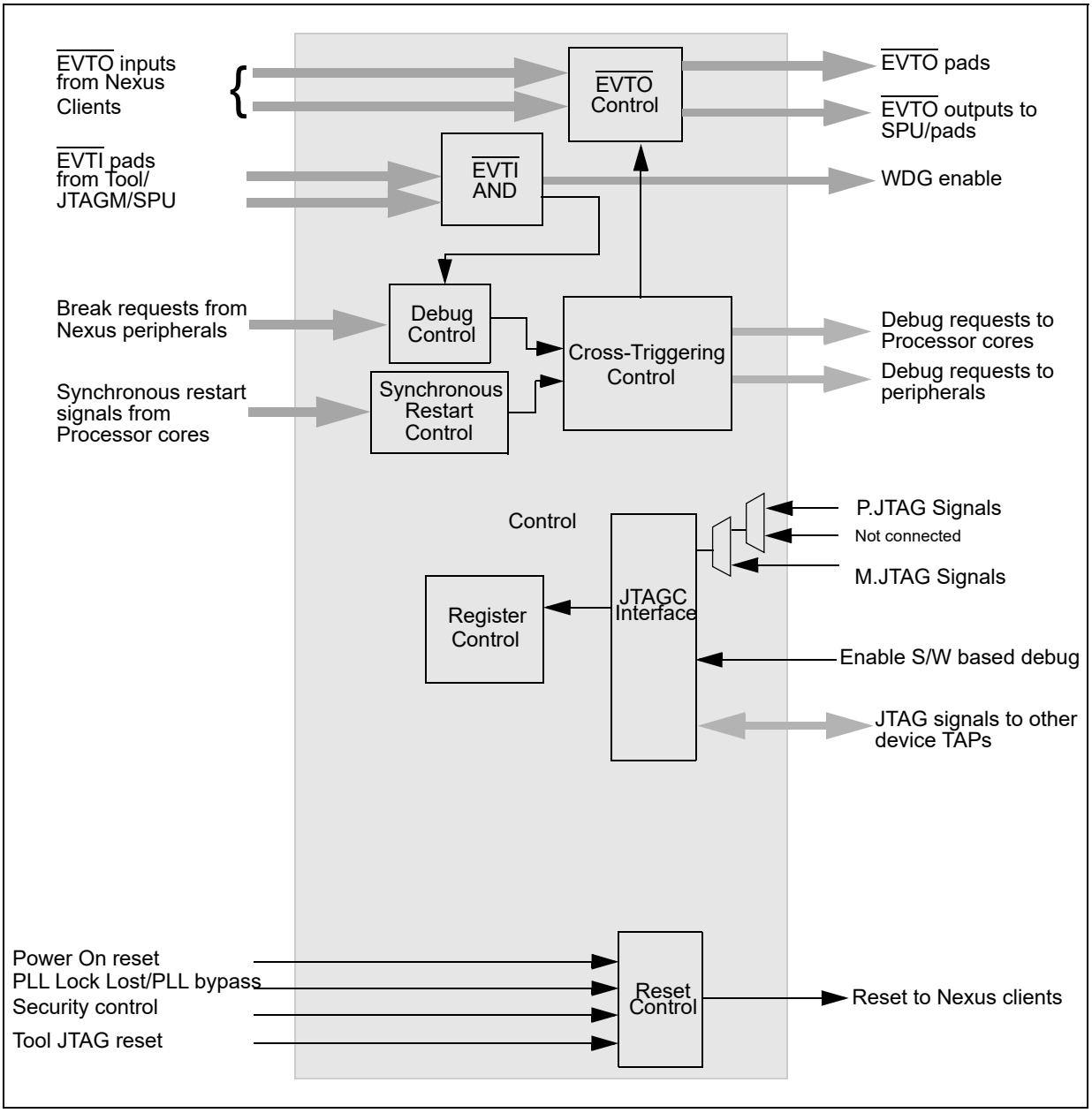
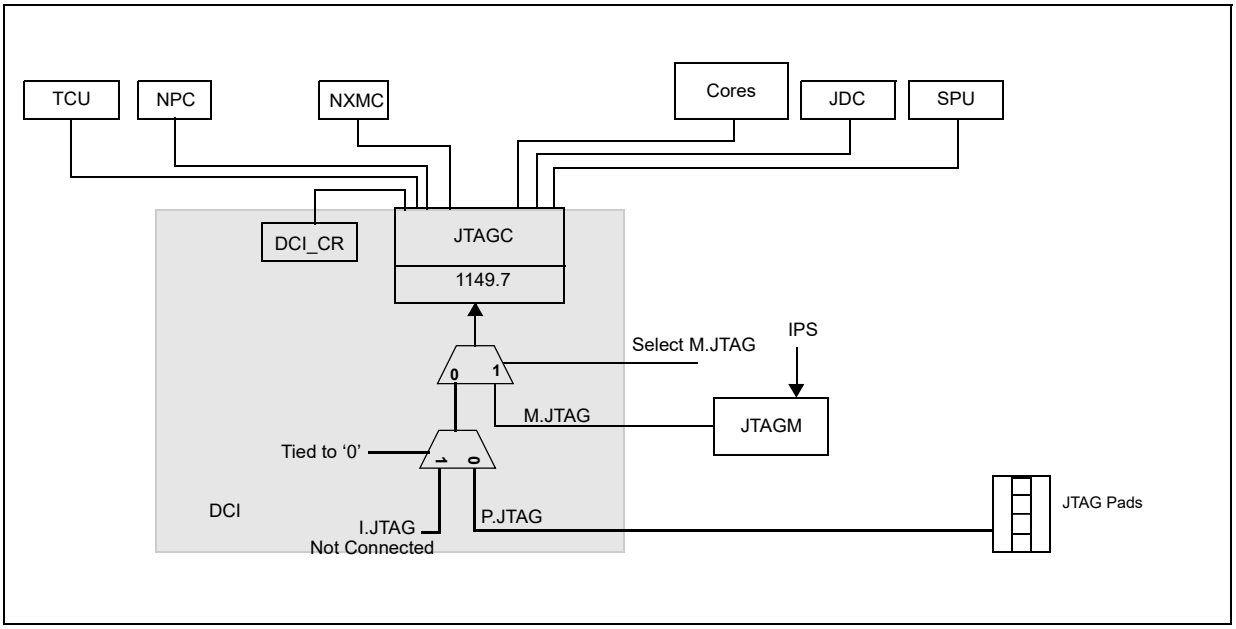


Figure 1126 shows the DCI configuration for the standard production package.

Figure 1126. DCI in a standard configuration



54.1.3 Operating modes

Selection between various JTAG sources is done using a JTAGM register bit, JTAGM_MCR[DTM]. The JTAGM bit, JTAGM_MCR[DTM], is programmable via memory-mapped registers. JTAGM is selected as a JTAG source when JTAGM_MCR[DTM] is set.

Table 1085 shows a summary of the configuration options.

Table 1085. DCI JTAG source selection

JTAGM_MCR[DTM]	DCI Source	JCOMP
0	P.JTAG (Standard JTAG)	Asserted
1	M.JTAG (Software)	Negated

The reset signal to the DCI block consists of the following:

- Power-on reset, Low/High voltage detection on IO/Core supplies
- JCOMP reset

The DCI puts the JTAGC TAP in reset when either power-on/low-voltage-detect reset is asserted or JCOMP is negated.

The M.JTAG provides an alternate interface to the debug logic.

When no debug tool is connected, device power-on reset puts the DCI in standard 5-pin JTAG mode. This configures the TDO pin in output mode with the reset value (1'b0). There is no change in port values if no debug tool is connected.

Table 1086 contains the operating mode descriptions.

Table 1086. DCI operating modes

Mode	Description
DCI_JTAG	JTAG mode. This is the default state after power-on reset. In this mode the DCI is in JTAG mode (controlled with P.JTAG) and is controlled by an external JTAG tool.
DCI_SW_DBG	Software Debug State. This state is achieved when no tool is present and software sets the JTAGM_MCR[DTM] bit. The DCI is then controlled using M.JTAG signals.

54.1.3.1 Software enabled debug and calibration mode

Apart from the JTAG sources, such as a JTAG-based tool , it is possible to generate JTAG sequences through software programming using JTAGM IP.

To support a software debug session via a CAN or SPI link, the software can enable debug and calibration mode via an memory-mapped register. It is done first by setting a control bit in the JTAGM (JTAGM_MCR[DTM]). For more details, refer to the JTAGM documentation. When enabled, the DCI is configured to receive M.JTAG signals which are driven by JTAGM registers accessible through software. Software can control all DCI features like any other JTAG source.

After a system reset, the JTAGM_MCR[DTM] bit is reset and the JCOMP pad is pulled-low, thus keeping the DCI in the reset state. Therefore, software is required to first change to a non-secured mode. Then it can program the JTAGM_MCR[DTM] bit, to select M.JTAG as the source inside the DCI, and then set the JTAGM_MCR[jtagm_JCOMP] bit present in the JTAGM, which takes all debug modules out of reset.

When MCR[jtagm_JCOMP] is programmed to 0, the JTAGC block reset is asserted, thus putting the debug and calibration logic in reset state. This bit is used during software based debug to initialize the debug and calibration logic.

Software based debug is not possible when a tool is connected. The presence of the tool is determined using the rising edge of the JCOMP pad to set an internal flag which, when set, disables the selection of JTAGM as a JTAG source.

Note: It is recommended to utilize either hardware based debug (JTAG) or software based debug. If only one of the debug method is used, all three connection methods (connect and reset target, connect and break application, connect and keep application running) are supported.

If both debug methods are used simultaneously, then an external tool may arbitrarily connect when the SoC has already been halted using the software debug mode. In this case, two consecutive assertions of external JCOMP are necessary to allow the external tool to take control of debug session.

54.1.3.2 Nexus reset control

The JCOMP input that is used as the primary reset signal for the DCI is also used by the DCI to generate a single-bit reset signal for other Nexus blocks. It is used to reset all debug

functions without affecting any system level functions. The DCI provides the reset signal for all Nexus blocks. This reset is controlled using the following signals:

- Power-on reset/Low voltage detect
- JCOMP
- SoC Debug_Enable signal
- System PLL is out of lock state and not bypassed

Asserting power-on reset, negating JCOMP, or negating the Debug_Enable signal, results in asynchronous entry into reset state. Following negation of power-on reset, the Nexus reset signal remains asserted until the system clock achieves lock or in bypass mode, as indicated by the system clock lock status signal, and Debug_Enable is asserted and JCOMP is asserted. Any subsequent loss of PLL lock does not generate a Nexus reset. This reset signal is unaffected by other sources of reset.If Debug_Enable is de-asserted by the SoC security modules, then the DCI holds the debug and trace logic in reset. The Debug_Enable signal is generated by security logic after functional or destructive reset based on the device censored mode, the PASS_LOCK3[DBL] control, and the device life cycle. If the Debug_Enable signal is asserted by the security logic, it is possible to trace through functional resets

The following table shows how the Nexus reset is asserted/negated based on various signals.

Table 1087. Nexus reset truth table

POR	JCOMP	Debug_Enable	PLL lock	Nexus reset
Assert	X	X	X	Assert
X	Negate	X	X	Assert
X	X	Disable	X	Assert
X	X	X	Unlocked	Assert
Negate	Assert	Enable	Locked	Negate

54.1.3.3 System watchdog control

The DCI is also used to enable or disable the Software Watchdog Timer (SWT). The DCI provides a single bit SWT enable signal which is used to control all SWTs in the MCU. For this, the EVTI[0] pin is used. If the EVTI[0] pin remains high when the tool asserts JCOMP, then the SWT is enabled. But if the tool drives the EVTI[0] pin low before asserting JCOMP pin, then the SWT is disabled. By default the SWT is enabled and a tool has an option to enable/disable the SWT using the EVTI[0] pin.

54.1.3.4 Break control

The DCI provides central break control among the CPUs and non-CPU debug modules such as DMA, Watchdog and other peripherals.

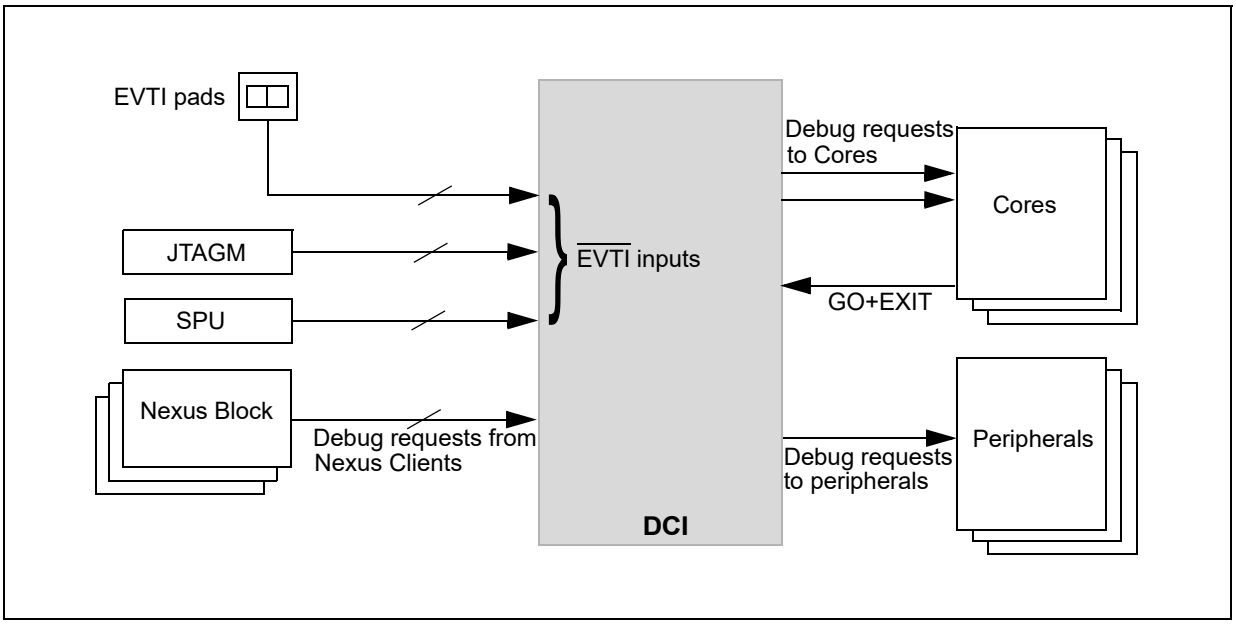
There are various different input sources that provide debug requests:

- $\overline{\text{EVTI}}$[0] signal from EVTI pad
- $\overline{\text{EVTI}}$[1:0] signals from JTAGM
- $\overline{\text{EVTI}}$[1:0] signals from the Sequence Processing Unit (SPU)

The DCI generates following debug request signals:

- Debug Request signal to be used for all non-CPU peripherals
- External Debug Request to be used to request external debug event to processor cores
- OnCE Debug Request to be used to request OnCE debug event to processor cores
- All break control requests are issued simultaneously to CPUs and peripherals, then synchronized to each clock domain

Figure 1127. DCI debug control mechanism



The various sources of $\overline{\text{EVTI}}$ inputs are synchronized and logically ANDed in the DCI to generate the following:

- Debug requests for each processor core
- External debug requests for each processor core
- Debug request for all non-CPU peripherals

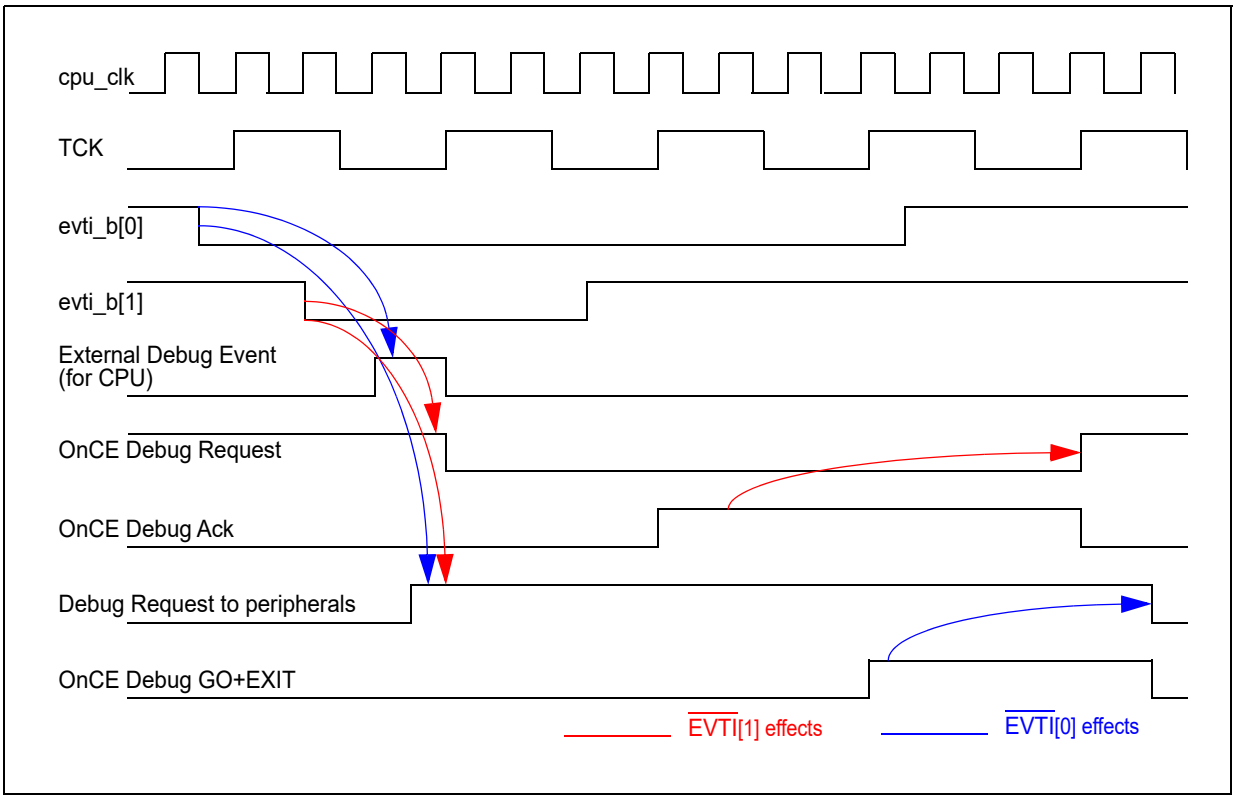
The $\overline{\text{EVTI}}$[0] from the EVTI pads, SPU and JTAGM are used to generate debug requests for non-CPU peripherals and for processor cores through the external debug request signal. The $\overline{\text{EVTI}}$[1] from the SPU and JTAGM, is also used to generate debug requests for non-CPU peripherals and it generates debug requests to processor cores through the OnCE debug request signal.

The external debug request to processor core is asserted as a single cycle pulse and is synchronized on the respective core clock. The OnCE debug request is negated when the acknowledge signal from the Cores is asserted.The CPU(s) remain in the debug state until GO+EX command is executed for that CPU. Exiting a CPU out of debug state by executing GO+EX command does not take other CPUs out of debug state. The peripheral debug request is negated when the processor core(s) execute GO+EXIT command.The $\overline{\text{EVTI}}$[0] signal coming from the pad is enabled for debug control using DCI_CR[4] bit.

In order to selectively enable/disable debug requests to system peripherals, the respective debug enable bit inside the peripheral should be used.

Figure 1128 shows the timing sequence.

Figure 1128. DCI debug signal generation



54.1.3.5 Cross-triggering control

Many of the blocks in the device, including the Nexus trace blocks and more sophisticated peripherals, have debug features that allow them to request device-wide breaks to enter debug mode. Many blocks can be programmed to halt operation upon debug mode entry or ignore it and continue normal operation. All debug mode entry requests are input to the DCI, ORed together and the result is used to produce the debug requests to the CPUs and non-CPU peripherals.

The cross-triggering feature is implemented through selectively enabling debug control bits in the respective modules. Processor cores (enabled using DBCR0[DEVT1] bit) and non-core peripherals can be programmed to halt operation through their respective control bits. Cross-triggering provides a selective feature to put modules in debug mode upon receiving halt request signals from Nexus modules.

54.1.3.6 Synchronous restart control

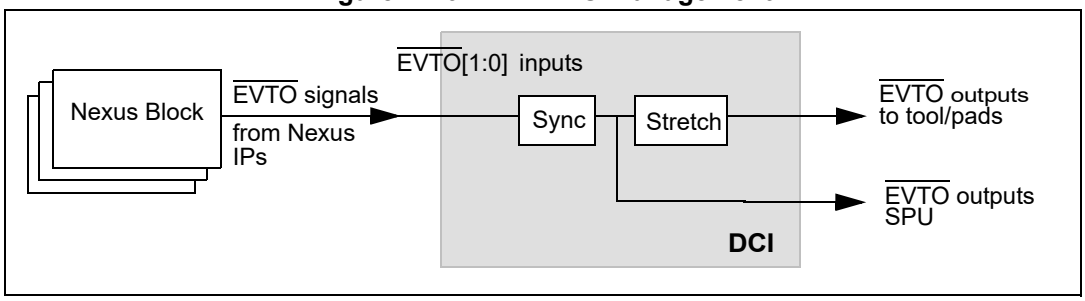
The DCI implements a synchronous restart control feature to synchronously restart all debug peripherals (CPU and non-CPU peripherals which are put in debug state). This is accomplished with the following steps:

1. The tool sequentially scans into CPUSCR for each processor core to restore the proper processor state.
2. The tool scans into the Production device tap controller the AUX_PARALLEL instruction, which then selects all processor core tap controllers in parallel.
3. To synchronously exit any combination of processor cores from debug mode, the tool simultaneously scans into all processor core tap instruction registers (OCMD) the 10-bit sequence 0x180 (that is set GO and EX, clear RS[0:6]).
4. If a processor core is out of debug mode, it sets a flag OnCE GO+EXIT.
5. When the OR of the processor core OnCE GO+EXIT signals is asserted, then all other selected non-processor cores and peripherals in debug would also be synchronously exited from debug mode. It is done by negating peripheral debug request signal.

54.1.3.7 EVTO management

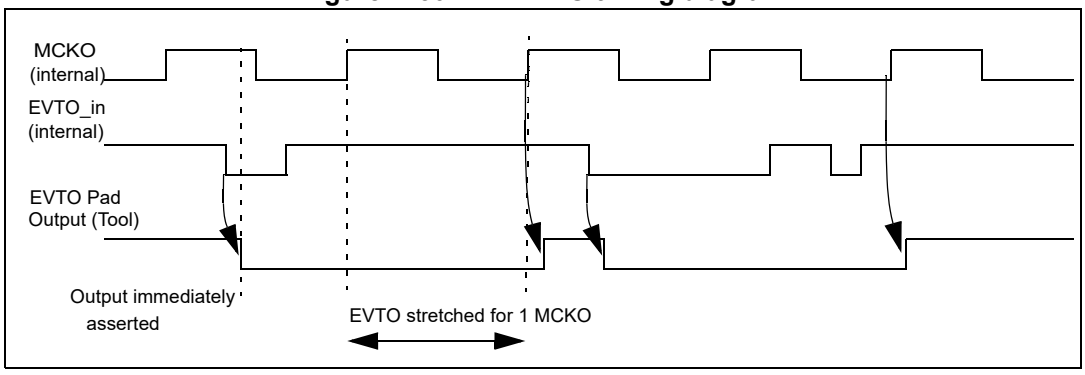
The EVTO management block diagram is presented in *Figure 1129*.

Figure 1129. DCI EVTO management



The DCI module controls distribution of the EVTO output from all Nexus Clients including the cores that produce EVTO[0] signals. The DCI uses a high frequency clock to sample EVTO signal coming from all modules which might generate EVTO outputs at their own clock frequency. After receiving a single clock period of asserted EVTO[0] signals from any Nexus client, the DCI immediately asserts EVTO[0] outputs and stretches it for a minimum of one output clock period for the output to the tool(*Figure 1130*). The DCI also generates EVTO output in other device clock domains so that other Nexus clients can receive EVTO input without the need of any synchronization. EVTO function is active as long as the DCI is not in reset.

Figure 1130. DCI EVTO timing diagram



54.1.3.8 Reset control

The DCI includes control bits to allow a connected tool to force a reset of the connected device. The description is as follows:

- Force functional reset—This reset is generated by programming the DCI_CR[FPD_FUNC] bit. When set, it forces functional reset. This bit has to be reset by the tool to negate the reset signal.
- Force POR or destructive reset—This reset is generated by programming DCI_CR[FPD_DEST] bit. When set it forces destructive reset. This bit has to be reset by the tool to negate the reset signal.

These reset signals are directly connected to the respective DCI_CR bits. The DCI does not reset the DCI_CR bits after generation of the respective reset signals. The tool must specifically reset the appropriate DCI_CR bits to remove the reset signals. These reset

control bits can be set/reset through JTAG programming, and only reset through JCOMP assertion or Power-on-reset.

54.2 External signal description

The JTAGC module used inside DCI module defines the interface with external signals. It consists of five signals (TMS/TDI/TDO/TCK/JCOMP) that connect to off chip development tools and allow access to test support functions. For more details for JTAG signals, refer to JTAGC documentation.

54.3 Register description

The PD DCI module contains two registers, DCI_CR and DCI_PINCR, which are present inside the JTAGC module. These registers are only controlled using the JTAGC TAP and no memory-mapped registers are present.

54.3.1 DCI control register (DCI_CR)

The DCI_CR is a 32-bit data register. This register is programmed using JTAG shift with the corresponding enable instruction described in the JTAGC chapter. *Figure 1131* shows the format of the DCI_CR.

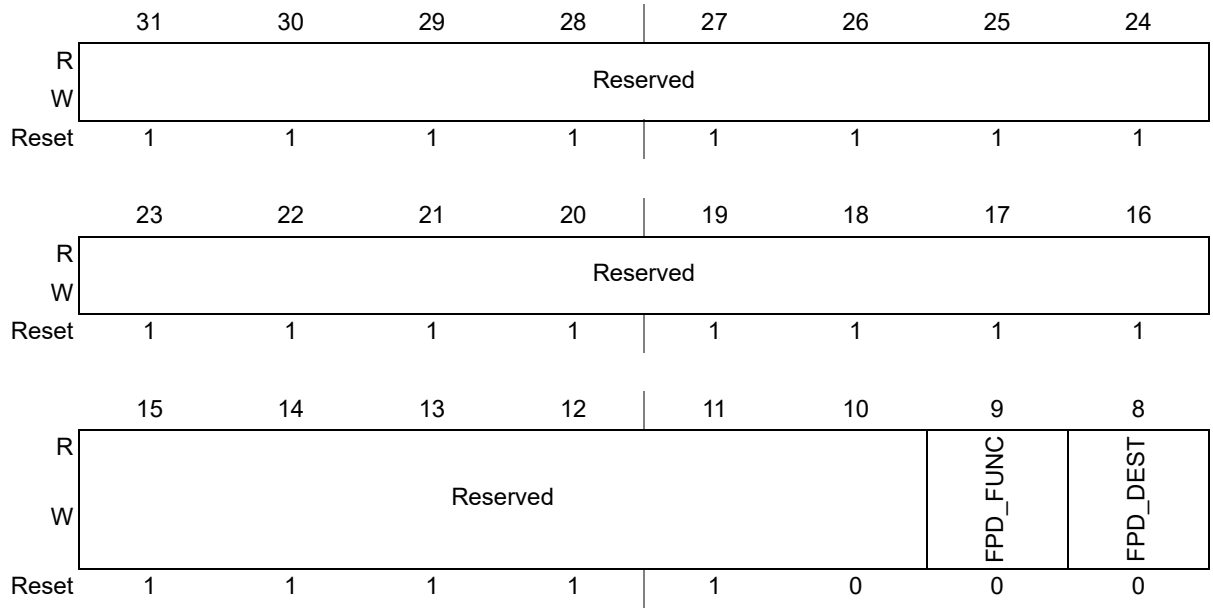


Figure 1131. DCI control register (DCI_CR)

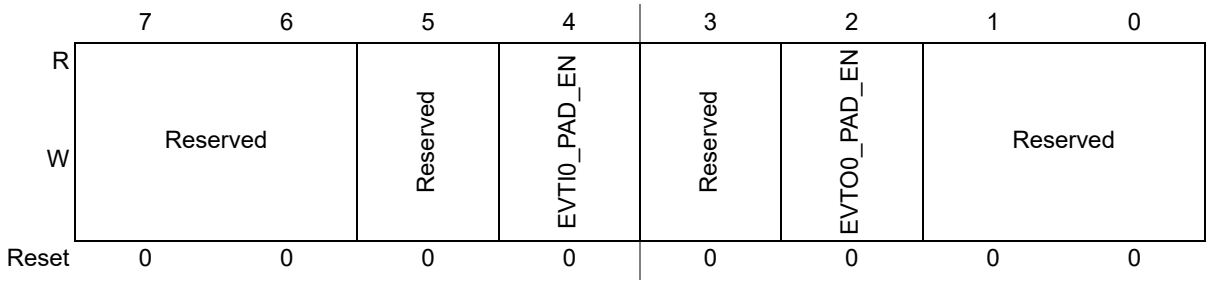


Figure 1131. DCI control register (DCI_CR) (continued)

The DCI_CR is described in *Table 1088*.

Table 1088. DCI_CR field descriptions

Field	Description
9 FPD_FUNC	Force Functional reset 0 Do not force functional reset 1 Force functional reset
8 FPD_DEST	Force Destructive reset 0 Do not force destructive reset 1 Force destructive reset
4 EVTI0_PAD_EN	$\overline{\text{EVTI}}$[0] feature enable 0 $\overline{\text{EVTI}}$[0] disabled (default) 1 $\overline{\text{EVTI}}$[0] enabled
2 EVTO0_PAD_EN	Output enable for $\overline{\text{EVTO}}$[0] pad 0 $\overline{\text{EVTO}}$[0] pad output not enabled (default) 1 $\overline{\text{EVTO}}$[0] pad output enabled

54.3.2 DCI EVTx pin multiplexing control register (DCI_PINCR)

The DCI_PINCR is a 64-bit data register. This register is programmed using JTAG shift with the corresponding enable instruction described in the JTAGC chapter, ENABLE_DATA2. For a description of the DCI_PINCR functionality, see the Calibration and Debug configuration chapter (or the Debug and Trace chapter) which describes how the modules are configured.

55 Debug and Trace

55.1 Introduction

55.1.1 Features

SPC574Sx includes comprehensive features to assist in the software development process:

- Supports Nexus class 3 debug and trace
- Supports memory reconstruction[r]
- Supports data tracing without loss of data
- Supports CPU controlled dump via the trace port and NXMC clients
- All bus masters are traceable
- Device CPU instruction/data trace captured using the CPU's own Nexus 3 module
- Device Flexray trace captured using a dedicated NXMC Nexus trace client
- Device DMA trace captured using a dedicated NXMC Nexus trace client
- Debug mode enable control for connected tool or software
- IEEE 1149.1 and 1149.7 controllers
- Debug break and cross-triggering control
- Synchronous restart control for CPU when exiting debug mode
- Tool hot plug capability
- Security access control
- Debug reset control

Internal debug support in the e200z420n3 and e200z419n3 cores allows for software and hardware debug by providing debug functions, such as instruction and data breakpoints and program trace modes.

The e200z420n3 and e200z419n3 Nexus 3 modules provide real-time development capabilities for e200z4d processors in compliance with the IEEE-ISTO 5001 standard. This module provides development support capabilities without requiring the use of address and data pins for internal visibility.

The development features supported are Program Trace, Data Trace, Watchpoint Messaging, Ownership Trace, Data Acquisition Messaging, and Read/Write Access via the JTAG interface. The Nexus 3 module also supports two Class 4 features: Watchpoint Triggering and Processor Overrun Control.

r. For the instrumentation use case, external reconstruction of the internal RAM content is required.

55.1.2 Architecture

The top level debug and trace architecture is shown in *Figure 1132*. It comprises the following modules:

- Debug Control Interface (DCI)
- JTAG Master (JTAGM)
- 2 Nexus Crossbar Multimaster Trace Clients (NXMC)
- Nexus Port Controller (NPC)
- Sequence Processing Unit (SPU)
- JTAG Data Communication Module (JDC)

The CPUs themselves provide the Nexus 3 module and the OnCE module. *Figure 1132* shows the interconnections between the debug and trace modules. The JTAG port pins are shared between different modules. The DCI block acts as a port sharing controller among the Nexus class 1 (JTAG) features, the JDC (JTAG to SW interface), and the JTAGM block (JTAG control via SW). The Multi-Nexus architecture ends up to the NPC (Nexus Port Controller) module, which arbitrates the various Nexus tracing requests coming from different Nexus clients. The Nexus scheme includes the possibility to trace the CPU data and instructions and the DMA accesses. Two additional Nexus clients (NXMC0 and NXMC1) have been added to trace respectively the DMA and FlexRay accesses.

The DCI sends and receives the $\overline{\text{EVTI}}$[1:0] and $\overline{\text{EVTO}}$[1:0] signals to/from the cores and other modules. *Table 1089* summarizes the sources and destinations of $\overline{\text{EVTI}}$, $\overline{\text{EVTO}}$, and related signals for the PD. For example, for the PD, $\overline{\text{EVTI}}$[0] are outputs of the JTAGM and SPU, and inputs to the Cores and NXMC.

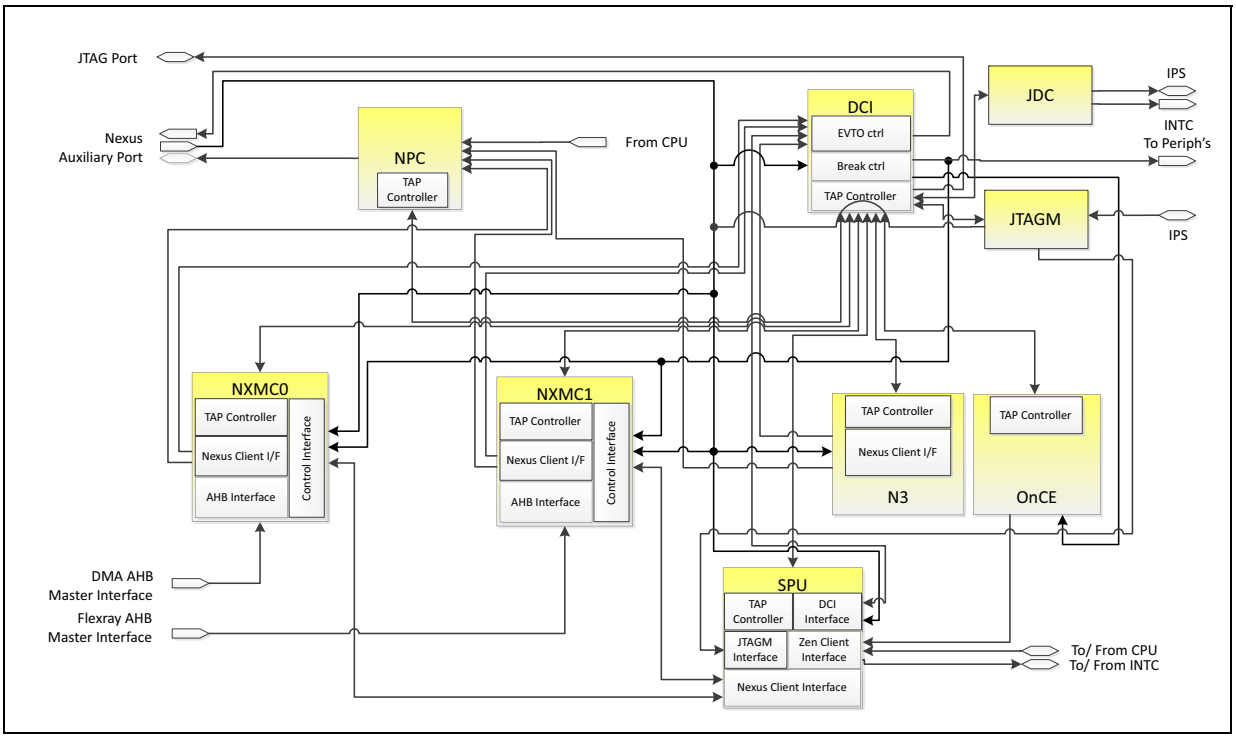
Table 1089. PD DCI signals

Signal	Cores	NXMC	JTAGM	SPU	DTS
$\overline{\text{EVTI}}$[0]	I	I	O	O	—
$\overline{\text{EVTI}}$[1]	—	—	O	O	—
$\overline{\text{EVTO}}$[0]	O	O	I	I	—
$\overline{\text{EVTO}}$[1]	—	—	I	I	O
Core debug event	I	—	—	—	—
DCI cross triggering	O	—	—	—	—

The DCI sends the system debug signal, ipg_debug, to the following modules. For details regarding module functionality during system debug, refer to the individual module chapter.

- CAN
- eTimer
- DSPI
- eDMA
- FCCU
- PWM
- NXMC
- PIT
- SARADC
- SRX
- SSCM
- STM
- SWT

Figure 1132. Debug architecture



55.2 Register description

55.2.1 Nexus Crossbar Multimaster Trace Clients (NXMC)

The source of each Nexus trace message is identified in the source field (SRC) of the Nexus trace message. This 4-bit field is used by tools, along with the Nexus Message type (TCODE), to determine the decoding of the trace message.

The Nexus trace sources are shown in *Table 1090*.

Table 1090. Nexus trace source identifiers

Nexus Trace Source	Source Trace ID	
e200_CORE_0	0b0000	0x0
AUX_BUS_MON_0	0b1100	0xC
AUX_BUS_MON_1	0b1101	0xD

55.2.2 Sequence Processing Unit (SPU)

The Sequence Processing Unit (SPU) provides on-device logic analyzer trigger functions, such as performance monitoring, by using internal device signals as trigger sources. Performance monitor functions are available at both the system and CPU level. Complex trigger and system performance monitor functions are implemented in the SPU module. System-level performance monitor functions are integrated into the SPU complex trigger logic, thus allowing the counting and timing of any debug trigger combinations supported by the SPU.

Various clients generate watchpoints and other triggers when operating in Debug mode. The SPU collects these triggers and uses them as conditions for programmable sequences of states and resultant actions. The SPU provides the capability to create complex debug triggers. By configuring each of the events to trigger specific actions, it achieves complex logic-analyzer-like behavior, providing vital real-time visibility and the ability to debug system activities.

55.2.2.1 SPU Level1 Input Mux Configuration registers

The SPU Level1 Mux inputs for the L1SELn registers (where n = 0 to 7) are described in the following tables.
Table 1091 contains the L1SEL0 register field descriptions.

Table 1091. L1SEL0 register field descriptions

Field	Description
MUX 0	000 CPU0 instruction address compare 1 (watchpoint 0) 001 Reserved 010 CPU2 instruction address compare 1 (watchpoint 0) 011 Reserved 100 Reserved 101 Reserved 110 CPU0 debug event input 1 (watchpoint 10) 111 CPU0 debug event output 2 (watchpoint 20)
MUX 1	000 CPU0 instruction address compare 2 (watchpoint 1) 001 Reserved 010 CPU2 instruction address compare 2 (watchpoint 1) 011 Reserved 100 Reserved 101 Reserved 110 Reserved 111 Reserved
MUX 2	000 CPU0 instruction address compare 3 (watchpoint 2) 001 Reserved 010 CPU2 instruction address compare 3 (watchpoint 2) 011 Reserved 100 Reserved 101 Reserved 110 CPU2 debug event input 1 (watchpoint 10) 111 CPU2 debug event output 2 (watchpoint 20)
MUX 3	000 CPU0 instruction address compare 4 (watchpoint 3) 001 Reserved 010 CPU2 instruction address compare 4 (watchpoint 3) 011 Reserved 100 Reserved 101 Reserved 110 Reserved 111 Reserved
MUX 4	000 CPU0 instruction address compare 5 (watchpoint 8) 001 Reserved 010 CPU2 instruction address compare 5 (watchpoint 8) 011 Reserved 100 Reserved 101 Reserved 110 CPU0 debug event input 2 (watchpoint 11) 111 CPU0 debug event output 3 (watchpoint 21)

Table 1091. L1SEL0 register field descriptions (continued)

Field	Description
MUX 5	000 CPU0 instruction address compare 6 (watchpoint 9) 001 Reserved 010 CPU2 instruction address compare 6 (watchpoint 9) 011 Reserved 100 Reserved 101 Reserved 110 Reserved 111 Reserved
MUX 6	000 CPU0 instruction address compare 7 (watchpoint 14) 001 Reserved 010 CPU2 instruction address compare 7 (watchpoint 14) 011 Reserved 100 Reserved 101 Reserved 110 CPU2 debug event input 2 (watchpoint 11) 111 CPU2 debug event output 3 (watchpoint 21)
MUX 7	000 CPU0 instruction address compare 8 (watchpoint 15) 001 Reserved 010 CPU2 instruction address compare 8 (watchpoint 15) 011 Reserved 100 Reserved 101 Reserved 110 Reserved 111 Reserved

Table 1092 contains the L1SEL1 register field descriptions.

Table 1092. L1SEL1 register field descriptions

Field	Description
MUX 8	000 CPU0 instruction address compare 1 (watchpoint 0) 001 Reserved 010 CPU2 instruction address compare 1 (watchpoint 0) 011 Reserved 100 Reserved 101 Reserved 110 CPU0 debug event output 0 (watchpoint 12) 111 Reserved
MUX 9	000 CPU0 instruction address compare 2 (watchpoint 1) 001 Reserved 010 CPU2 instruction address compare 2 (watchpoint 1) 011 Reserved 100 Reserved 101 Reserved 110 Reserved 111 Reserved
MUX 10	000 CPU0 instruction address compare 3 (watchpoint 2) 001 Reserved 010 CPU2 instruction address compare 3 (watchpoint 2) 011 Reserved 100 Reserved 101 Reserved 110 CPU2 debug event output 0 (watchpoint 12) 111 Reserved
MUX 11	000 CPU0 instruction address compare 4 (watchpoint 3) 001 Reserved 010 CPU2 instruction address compare 4 (watchpoint 3) 011 Reserved 100 Reserved 101 Reserved 110 Reserved 111 Reserved
MUX 12	000 CPU0 instruction address compare 5 (watchpoint 8) 001 Reserved 010 CPU2 instruction address compare 5 (watchpoint 8) 011 Reserved 100 Reserved 101 Reserved 110 CPU0 debug event output 1 (watchpoint 13) 111 Reserved

Table 1092. L1SEL1 register field descriptions (continued)

Field	Description
MUX 13	000 CPU0 instruction address compare 6 (watchpoint 9) 001 Reserved 010 CPU2 instruction address compare 6 (watchpoint 9) 011 Reserved 100 Reserved 101 Reserved 110 Reserved 111 Reserved
MUX 14	000 CPU0 instruction address compare 7 (watchpoint 14) 001 Reserved 010 CPU2 instruction address compare 7 (watchpoint 14) 011 Reserved 100 Reserved 101 Reserved 110 CPU2 debug event output 1 (watchpoint 13) 111 Reserved
MUX 15	000 CPU0 instruction address compare 8 (watchpoint 15) 001 Reserved 010 CPU2 instruction address compare 8 (watchpoint 15) 011 Reserved 100 Reserved 101 Reserved 110 Reserved 111 Reserved

Table 1093 contains the L1SEL2 register field descriptions.

Table 1093. L1SEL2 register field descriptions

Field	Description
MUX16	000 CPU0 data address compare 1 (watchpoint 4) 001 Reserved 010 CPU2 data address compare 1 (watchpoint 4) 011 Reserved 100 CPU0 write to L1CSR0 101 Reserved 110 Reserved 111 Reserved
MUX 17	000 CPU0 data address compare 2 (watchpoint 5) 001 Reserved 010 CPU2 data address compare 2 (watchpoint 5) 011 Reserved 100 CPU0 write to L1CSR1 101 Reserved 110 Reserved 111 Reserved
MUX 18	000 CPU0 data address compare 3 (watchpoint 6) 001 Reserved 010 CPU2 data address compare 3 (watchpoint 6) 011 Reserved 100 CPU0 write to E2ECTL0 101 Reserved 110 Reserved 111 Reserved
MUX 19	000 CPU0 data address compare 4 (watchpoint 7) 001 Reserved 010 CPU2 data address compare 4 (watchpoint 7) 011 Reserved 100 CPU0 write to DMEMCTL0 101 Reserved 110 Reserved 111 Reserved
MUX 20	000 CPU0 data address compare 1 (watchpoint 4) 001 Reserved 010 CPU2 data address compare 1 (watchpoint 4) 011 Reserved 100 CPU0 write to IMEMCTL0 101 Reserved 110 CPU2 write to L1CSR0 111 Reserved

Table 1093. L1SEL2 register field descriptions (continued)

Field	Description
MUX 21	000 CPU0 data address compare 2 (watchpoint 5) 001 Reserved 010 CPU2 data address compare 2 (watchpoint 5) 011 Reserved 100 CPU0 write to HID0 101 Reserved 110 CPU2 write to L1CSR1 111 Reserved
MUX 22	000 CPU0 data address compare 3 (watchpoint 6) 001 Reserved 010 CPU2 data address compare 3 (watchpoint 6) 011 Reserved 100 CPU0 write to HID1 101 Reserved 110 CPU2 write to E2ECTL0 111 Reserved
MUX 23	000 CPU0 data address compare 4 (watchpoint 7) 001 Reserved 010 CPU2 data address compare 4 (watchpoint 7) 011 Reserved 100 CPU0 write to BUCSR 101 Reserved 110 CPU2 write to DMEMCTL0 111 Reserved

Table 1094 contains the L1SEL3 register field descriptions.

Table 1094. L1SEL3 register field descriptions

Field	Description
MUX 24	000 CPU0 data address compare 1 (watchpoint 4) 001 Reserved 010 CPU2 data address compare 1 (watchpoint 4) 011 Reserved 100 CPU0 write to SPEFSCR 101 Reserved 110 CPU2 write to IMEMCTL0 111 Reserved
MUX 25	000 CPU0 data address compare 2 (watchpoint 5) 001 Reserved 010 CPU2 data address compare 2 (watchpoint 5) 011 Reserved 100 CPU0 write to IVPR 101 Reserved 110 CPU2 write to HID0 111 Reserved
MUX 26	000 CPU0 data address compare 3 (watchpoint 6) 001 Reserved 010 CPU2 data address compare 3 (watchpoint 6) 011 Reserved 100 CPU0 write to MSR 101 Reserved 110 CPU2 write to HID1 111 Reserved
MUX 27	000 CPU0 data address compare 4 (watchpoint 7) 001 Reserved 010 CPU2 data address compare 4 (watchpoint 7) 011 Reserved 100 Reserved 101 Reserved 110 CPU2 write to BUCSR 111 Reserved
MUX 28	000 CPU0 data address compare 1 (watchpoint 4) 001 Reserved 010 CPU2 data address compare 1 (watchpoint 4) 011 Reserved 100 Reserved 101 Reserved 110 CPU2 write to SPEFSCR 111 Reserved

Table 1094. L1SEL3 register field descriptions (continued)

Field	Description
MUX 29	000 CPU0 data address compare 2 (watchpoint 5) 001 Reserved 010 CPU2 data address compare 2 (watchpoint 5) 011 Reserved 100 Reserved 101 Reserved 110 CPU2 write to IVPR 111 Reserved
MUX 30	000 CPU0 data address compare 3 (watchpoint 6) 001 Reserved 010 CPU2 data address compare 3 (watchpoint 6) 011 Reserved 100 Reserved 101 Reserved 110 CPU2 write to MSR 111 Reserved
MUX 31	000 CPU0 data address compare 4 (watchpoint 7) 001 Reserved 010 CPU2 data address compare 4 (watchpoint 7) 011 Reserved 100 Reserved 101 Reserved 110 Reserved 111 Reserved

Table 1095 contains the L1SEL4 register field descriptions.

Table 1095. L1SEL4 register field descriptions

Field	Description
MUX 32	000 CPU0 performance monitor counter 0 (watchpoint 23) 001 Reserved 010 CPU0 memory protection unit (watchpoint 27) 011 Reserved 100 Reserved 101 Reserved 110 Production device NAR full 111 GTM watchpoint 0
MUX 33	000 CPU0 performance monitor counter 1 (watchpoint 24) 001 Reserved 010 Reserved 011 Reserved 100 Reserved 101 Reserved 110 Production device NAR partial full 111 GTM watchpoint 1
MUX 34	000 CPU0 performance monitor counter 2 (watchpoint 25) 001 Reserved 010 CPU2 memory protection unit (watchpoint 27) 011 Reserved 100 Reserved 101 Reserved 110 Buddy device NAR partial full 111 GTM watchpoint 2
MUX 35	000 CPU0 performance monitor counter 3 (watchpoint 26) 001 Reserved 010 Reserved 011 Reserved 100 Reserved 101 Reserved 110 Buddy device NAR full 111 GTM watchpoint 3
MUX 36	000 CPU2 performance monitor counter 0 (watchpoint 23) 001 Reserved 010 CPU0 performance monitor event input (watchpoint 22) 011 Reserved 100 Reserved 101 Reserved 110 Reserved 111 GTM watchpoint 4

Table 1095. L1SEL4 register field descriptions (continued)

Field	Description
MUX 37	000 CPU2 performance monitor counter 1 (watchpoint 24) 001 Reserved 010 Reserved 011 Reserved 100 NXMC1 watchpoint 7 101 Reserved 110 Reserved 111 GTM watchpoint 5
MUX 38	000 CPU2 performance monitor counter 2 (watchpoint 25) 001 Reserved 010 CPU2 performance monitor event input (watchpoint 22) 011 Reserved 100 Reserved 101 Reserved 110 Reserved 111 GTM watchpoint 6
MUX 39	000 CPU2 performance monitor counter 3 (watchpoint 26) 001 Reserved 010 Reserved 011 Reserved 100 Reserved 101 Reserved 110 Reserved 111 GTM watchpoint 7

Table 1096 contains the L1SEL5 register field descriptions.

Table 1096. L1SEL5 register field descriptions

Field	Description
MUX 40	000 CPU0 interrupt (watchpoint 16) 001 CPU0 critical interrupt (watchpoint 18) 010 CPU0 data trace control range 1 (watchpoint 29) 011 Reserved 100 Reserved 101 Reserved 110 SPU counter event 0 111 GTM watchpoint 8
MUX 41	000 Reserved 001 Reserved 010 Reserved 011 Reserved 100 Reserved 101 Reserved 110 SPU counter event 1 111 GTM watchpoint 9
MUX 42	000 CPU2 interrupt (watchpoint 16) 001 CPU2 critical interrupt (watchpoint 18) 010 CPU2 data trace control range 1 (watchpoint 29) 011 Reserved 100 Reserved 101 Reserved 110 SPU counter event 2 111 GTM watchpoint 10
MUX43	000 Reserved 001 Reserved 010 Reserved 011 Reserved 100 Reserved 101 Reserved 110 SPU counter event 3 111 GTM watchpoint 11
MUX 44	000 CPU0 return (watchpoint 17) 001 CPU0 critical return (watchpoint 19) 010 CPU0 data trace control range 2 (watchpoint 30) 011 Reserved 100 Reserved 101 Reserved 110 SPU counter event 4 111 Reserved

Table 1096. L1SEL5 register field descriptions (continued)

Field	Description
MUX 45	000 Reserved 001 Reserved 010 Reserved 011 Reserved 100 Reserved 101 Reserved 110 SPU counter event 5 111 Reserved
MUX 46	000 CPU2 return (watchpoint 17) 001 CPU2 critical return (watchpoint 19) 010 CPU2 data trace control range 2 (watchpoint 30) 011 Reserved 100 Reserved 101 Reserved 110 SPU counter event 6 111 Reserved
MUX 47	000 Reserved 001 Reserved 010 Reserved 011 Reserved 100 Reserved 101 Reserved 110 SPU counter event 7 111 Reserved

Table 1097 contains the L1SEL6 register field descriptions.

Table 1097. L1SEL6 register field descriptions

Field	Description
MUX 48	000 CPU0 context load (using e_lmvsprw instruction) 001 CPU0 context restore (using e_stmvsprw instruction) 010 CPU0 data trace control range 3 (watchpoint 31) 011 NXMC0 watchpoint 0 100 Reserved 101 Reserved 110 SPU counter event 8 111 Reserved
MUX 49	000 Reserved 001 Reserved 010 Reserved 011 NXMC0 watchpoint 1 100 Reserved 101 Reserved 110 SPU counter event 9 111 Reserved
MUX 50	000 CPU2 context load (using e_lmvsprw instruction) 001 CPU2 context restore (using e_stmvsprw instruction) 010 CPU2 data trace control range 3 (watchpoint 31) 011 NXMC0 watchpoint 2 100 Reserved 101 Reserved 110 SPU counter event 10 111 Reserved
MUX 51	000 Reserved 001 Reserved 010 Reserved 011 NXMC0 watchpoint 3 100 Reserved 101 Reserved 110 SPU counter event 11 111 Reserved
MUX 52	000 CPU0 interrupt current priority match to C0PIS 001 CPU0 write to ME bits of MSR 010 Reserved 011 NXMC0 watchpoint 4 100 Reserved 101 Reserved 110 SPU counter event 12 111 Reserved

Table 1097. L1SEL6 register field descriptions (continued)

Field	Description
MUX 53	000 Reserved 001 Reserved 010 Reserved 011 NXMC0 watchpoint 5 100 Reserved 101 Reserved 110 SPU counter event 13 111 Reserved
MUX 54	000 CPU2 interrupt current priority match to C2PIS 001 CPU2 write to ME bits of MSR 010 Reserved 011 NXMC0 watchpoint 6 100 Reserved 101 Reserved 110 SPU counter event 14 111 Reserved
MUX 55	000 Reserved 001 Reserved 010 Reserved 011 NXMC0 watchpoint 7 100 Reserved 101 Reserved 110 SPU counter event 15 111 Reserved

Table 1098 contains the L1SEL7 register field descriptions

Table 1098. L1SEL7 register field descriptions

Field	Description
MUX 56	000 CPU0 write to EE bits of MSR 001 CPU0 write to CE bits of MSR 010 NXMC1 watchpoint 0 011 Reserved 100 Reserved 101 Reserved 110 Reserved 111 DCI EVTI0
MUX 57	000 Reserved 001 Reserved 010 NXMC1 watchpoint 1 011 Reserved 100 Reserved 101 Reserved 110 Reserved 111 DCI EVTI1
MUX 58	000 CPU2 write to EE bits of MSR 001 CPU2 write to CE bits of MSR 010 NXMC1 watchpoint2 011 Reserved 100 Reserved 101 Reserved 110 Reserved 111 DCI EVTO0
MUX 59	000 Reserved 001 Reserved 010 NXMC1 watchpoint 3 011 Reserved 100 Reserved 101 Reserved 110 Reserved 111 DCI EVTO1
MUX 60	000 CPU0 processor exception vector match to C0PEVP 001 CPU0 write to DE bits of MSR 010 NXMC1 watchpoint 4 011 Reserved 100 Reserved 101 Reserved 110 Reserved 111 DCI system halt

Table 1098. L1SEL7 register field descriptions (continued)

Field	Description
MUX 61	000 Reserved 001 Reserved 010 NXMC1 watchpoint 5 011 Reserved 100 Reserved 101 Reserved 110 Reserved 111 Reserved
MUX 62	000 CPU2 processor exception vector match to C2PEVP 001 CPU2 write to DE bits of MSR 010 NXMC1 watchpoint 6 011 Reserved 100 Reserved 101 Reserved 110 Reserved 111 Reserved
MUX 63	000 Reserved 001 Reserved 010 Reserved 011 Reserved 100 Reserved 101 Reserved 110 Reserved 111 1 Reserved

55.3 Lock step behavior

This feature will be provided for the external tool control only, while it will not be available for the JTAGM and SW-debug control. The lockstep debugging shall be intended as the possibility to inspect the registers of the lock-step core during a debug session[s].

In order to guarantee this, the TDI signal shall be sent to both cores, so that the two cores get the same debug configuration and evolve, even during the debug mode, along the same states.

Figure 1133 shows the proposed architecture for the lock-step debugging. Three AUX_PARALLEL modes exist:

- AUX_PARALLEL_0: this TAP chain is used to broadcast the same JTAG commands to both cores and to read the output from the Main core. TDI is connected to both the Main and Lockstep cores; TDO is connected to the Main core.
- AUX_PARALLEL_DELAY_0: this TAP chain is used to broadcast the same JTAG commands to both cores, but with a delay of 2 x **m_clk**, and to read the output from the Main core. TDI is connected to both the Main and Lockstep cores; TDO is connected to

s. No system data access via the XBAR is possible through the lock-step core only

the Main core; the JTAG input signals are synchronized to the **m_clk** (see *Figure 1134* and *Figure 1135*) a delay of 2 x **m_clk** is then applied to the JTAG input signals connected to the Lock-Step core.

- AUX_PARALLEL_1: this TAP chain is used to broadcast the same JTAG commands to both cores and to read the output from the Lockstep core. TDI is connected to both the Main and Lockstep cores; TDO is connected to the Main core.

Figure 1133. Lock-step debugging architecture

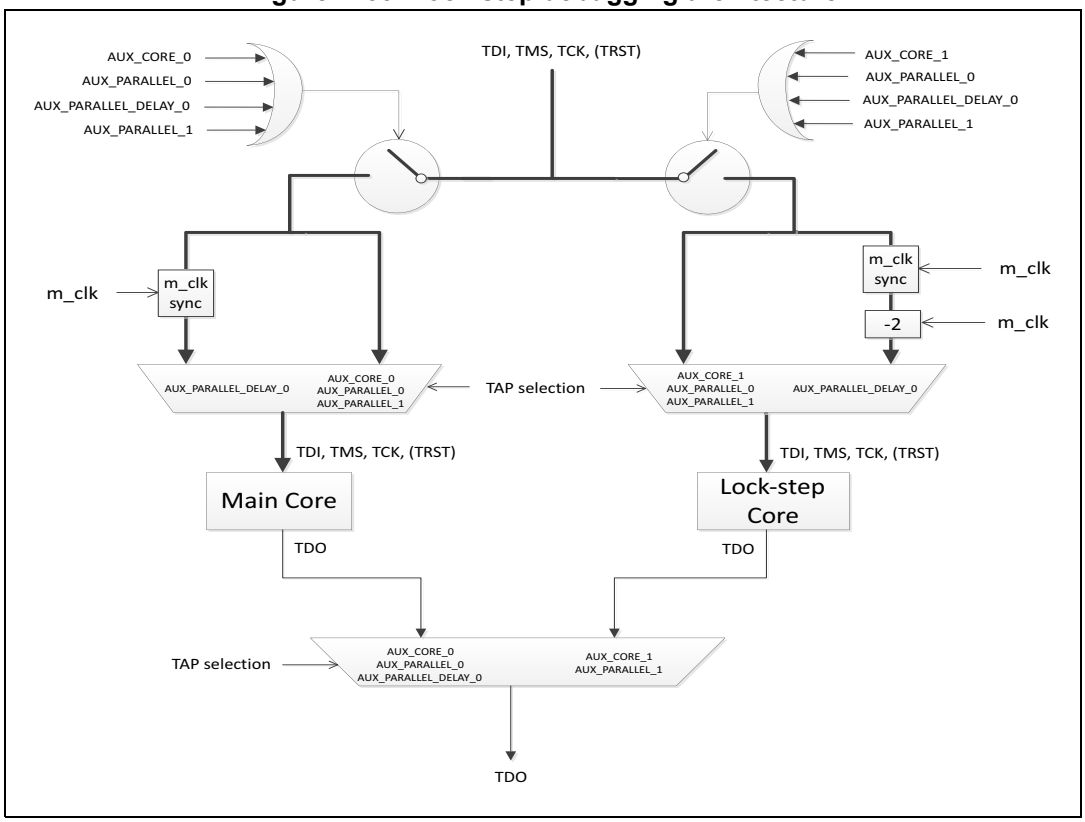


Figure 1134 and *Figure 1135* show how the synchronization is done. The TCK clock is synchronized to the system clock (**m_clk**) and a TCK_EN signal is also generated. The TCK_EN and the system clock are then used to sample the TMS and TDI signals.

Figure 1134. Architecture of the JTAG signal synchronization to the system clock

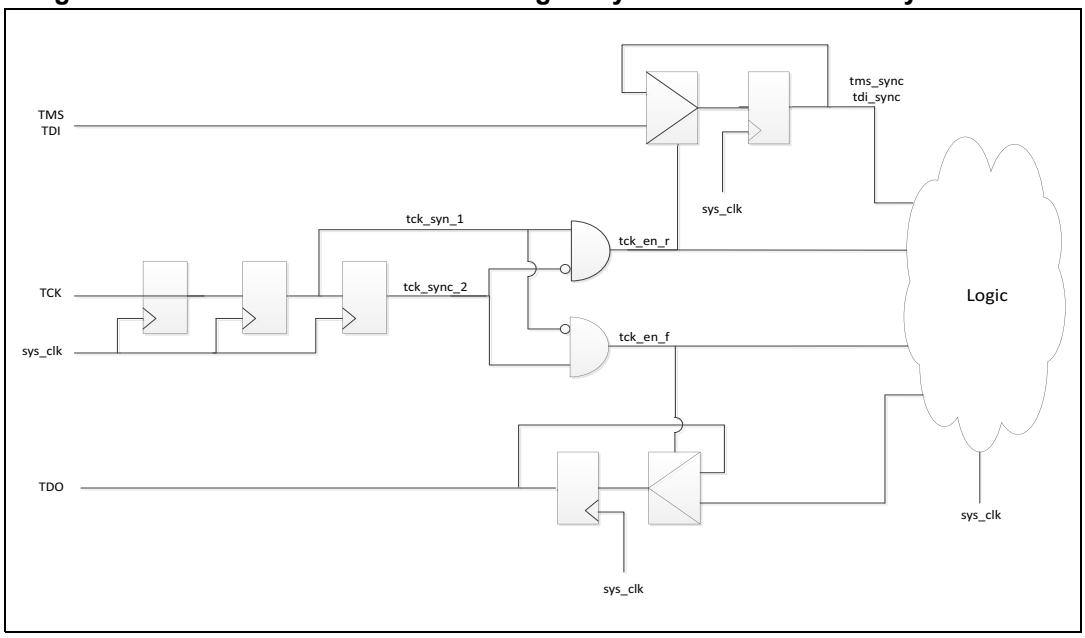
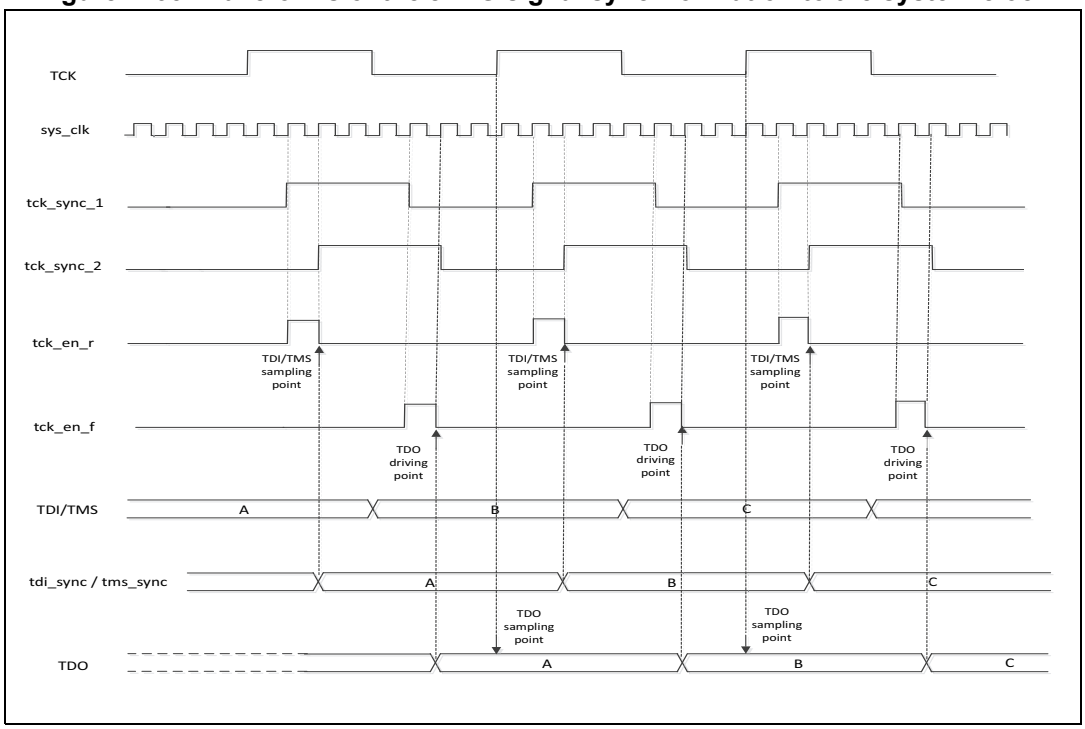


Figure 1135. Waveforms of the JTAG signal synchronization to the system clock



55.3.1 Debugging phases

The debug session can be divided into 4 phases:

1. Entry into the core Debug Mode
2. External tool acknowledge of the core Debug Mode
3. JTAG operations while the core is in Debug Mode
4. Exit from the core Debug Mode (step-by-step and free-run)

55.3.1.1 Entry into the core Debug Mode

The events causing the core to enter the debug mode can be internal or external. The internal debug events will be implicitly delayed by the fact that the cores are progressing with 2 x **m_clk** delay phase. Because the two cores are executing the same instructions, but delayed with 2 x **m_clk**, we can assume that the lock-step core will enter the debug state after the same delay. In case of external event, the **p_devt1** signal shall also be delayed between the Main core and the Lock-step core, in order to guarantee that after 2 x **m_clk**, the lock-step core is in debug state too.

55.3.1.2 External tool acknowledge of the core Debug Mode

Once a core enters into Debug Mode, the external tool will have to acknowledge this status and take control. The external tool will acknowledge the status by reading the OnCE status register. In order to make sure that both cores are stalled, the external tool shall read the OnCE status register of both cores.

55.3.1.3 JTAG operations while the core is in Debug Mode

Once the debug mode state has been acknowledged by the external tool for both the Main and Lock-step cores, then both cores are stalled in the same state.

Now the external tool can take control and send JTAG commands in order to set-up breakpoints and watchpoints.

By selecting the AUX_PARALLEL_0 or AUX_PARALLEL_1 TAPs, the external tool can send the same JTAG input signals (TCK, TMS, TDI) synchronously to both cores and select which TDO shall be used to read the data out.

This operation is suitable to set up the debug configurations (such as breakpoints and watchpoints) and also to read the core registers.

55.3.1.4 Exit from the core Debug Mode (step-by-step and free-run)

To let the core execute the code again (step-by-step or free-run), the external tool shall send a GO instruction. Due to the fact that the input bus signals of the lock-step core are hard-wired with a delay of 2 x **m_clk**, it is mandatory that the two cores restart with 2 x **m_clk** delay, otherwise the Lock-Step core will not be able to run correctly[t]. In order to achieve this, the external tool shall connect the AUX_PARALLEL_DELAY_0 TAP before the GO instruction is sent. From a timing point of view, this means that the core restart operation is possible only if the Main and Lock-step cores evolve synchronously, keeping a delay of 2 x **m_clk** clock cycles, otherwise the state of the two cores will never be synchronized.

t. The situation is similar to what happens after a system reset. In that case, the system reset shall be delayed for the lock-step core, otherwise the lock-step core will start fetching wrong data instructions

Therefore, the JTAG signals shall be synchronized to the **m_clk** when the AUX_PARALLEL_DELAY_0 TAP is selected.

55.3.2 Internal register reading

Access to processor registers and the contents of memory locations are performed by enabling external debug mode (setting DBCR0EDM to '1'), placing the processor into debug mode, followed by scanning instructions and data into and out of the CPU Scan Chain (CPUSCR).

Execution of scanned instructions by the CPU is used as the method to access required data. Memory locations may be read by scanning a load instruction into the Zen core which will reference the desired memory location, executing the load instruction, and then scanning out the result of the load. Other resources are accessed in a similar manner.

The WBBR is used as a means of passing operand information between the CPU and the external command controller. Whenever the external command controller needs to read the contents of a register or memory location, it will force the chip to execute an instruction (by using the GO command) that brings that information to the WBBR. $WBBR_{low}$ holds the 32-bit result of most instructions including load data returned for a load or load with update instruction. $WBBR_{high}$ holds the updated effective address calculated by a load with update instruction. It is undefined for other instructions. As an example, to read the lower 32 bits of processor register r1, an **e_ori r1,r1,0** instruction is executed, and the result value of the instruction will be latched into $WBBR_{low}$.

For resources outside of the CPU block and contained in the OnCE block, the processor is not disturbed, and may continue execution. If a processor resource is required, an internal debug request (dbg_dbgrq) may be asserted to the CPU by the OnCE controller, causing the CPU to finish the current instruction being executed, save the instruction pipeline information, enter Debug Mode, and wait for further commands.

Before executing any instruction, the external tool shall save the CPU context (CPUSCR chain). This operation is done by using the dedicated TAP chains, which send the JTAG command to one core only (AUX_CORE_0 or AUX_CORE_1).

By selecting the AUX_CORE_0 or AUX_CORE_1 TAPs, the external tool can send the JTAG input signals (TCK, TMS, TDI) to one core only, and connect it to the TDO output. This operation is suitable to read out each core register.

55.3.3 Step-by-step execution

Single stepping of instructions is performed by placing the CPU in debug mode, scanning in appropriate information into the CPUSCR register, and setting the GO bit (with the EX bit cleared) with the RS field indicating either the CPUSCR or No Register Selected. The instruction opcode is inserted into the IR register. The Instruction Register (IR) provides a mechanism for controlling the debug session by serving as a means for forcing in selected instructions, and then causing them to be executed in a controlled manner by the debug control block. The opcode of the next instruction to be executed when entering debug mode is contained in this register when the scan-out of this chain begins. This value should be saved for later restoration if continuation of the normal instruction stream is desired. The CPU will then temporarily exit the debug state (but not the debug session) to execute the instruction, and will then return to the debug state (indicated via the OnCE Status Register).

55.3.4 Exiting a debug session

The debug session remains in force until the final OnCE “go+exit” command is executed, at which time the CPU will return to the previous state it was in (unless a new debug request is pending). A scan into the CPUSCR is required prior to executing each “go+exit” or “go+noexit” OnCE command.

55.3.5 External tool operation

The external tool should implement the following flows (for the lock-step debugging case):

1. The external tool activates the AUX_CORE_0 and polls the debug status register of the Main core in order to check whether it is stalled.
2. The external tool activates the AUX_CORE_1 and polls the debug status register of the Lockstep core in order to confirm that the lockstep core is also stalled.
3. Now the 2 cores are stalled in the same state and the external tool, using the AUX_PARALLEL_0 and AUX_PARALLEL_1 can read their registers and set new breakpoints.
4. When the GO instruction is requested, the external tool selects the AUX_PARALLEL_DELAY_0 and afterwards sends the GO instruction

55.4 Lock step tracing

The current implementation does not foresee any data or instruction tracing capability for the lock-step core. The following reasons explain which are the issues with the lock-step core Nexus tracing:

- the core has no data trace capability
- the lockstep core is not able to drive any signal on the XBAR

During the lock-step debug session, the TDI commands go in parallel to both cores. If the instruction tracing is enabled and if the lock-step core Nexus client is connected to the NPC, then the same duplicated instructions will appear at the tool level.

55.5 External signal description

The JTAGC module used inside DCI module defines the interface with external signals. It consists of four signals (TDI/TDO/TCK/TMS) that connect to off chip development tools and allow access to test support functions. For more details for JTAG signals, refer to JTAGC documentation.

55.5.1 Interfaces

Two interfaces are used to support debug and trace functionality. These are the JTAG port and the Nexus Auxiliary Port (NAP). The JTAG is port is used for debug and test purposes while the NAP is used to transfer trace data streams.

55.5.1.1 JTAG port

The SPC574Sx features a JTAG port with a 4-pin interface. The supported signals are TDI, TDO, TCK and TMS. TRST is not used.

Table 1099. JTAG signals

Signal	Description	Direction	Width
TDI	Test Data Input	Input	1
TDO	Test Data Output	Output	1
TCK	Test Clock	Input	1
TMS	Test Mode Select	Input	1

55.5.1.2 Nexus auxiliary port

Table 1100. Nexus signals

Channel	Functionality	Direction	Width
EVTI	Event Trigger Input	Input	1
EVTO	Event Trigger Output	Output	1
MSEO	MDO Start/End Output	Output	2
MCKO	MDO and MSEO signal clock	Output	1
MDO(0…N[(1)])	Trace	Output	N[(1)]

1. 4 MDO pin configuration for QFP100. 12 MDO pin configuration for QFP144.

55.5.2 Nexus trace enabling during reset

During the reset phase it is possible to enable the Nexus tracing, so that the application can be monitored immediately after the reset phase. The steps needed to accomplish this feature are described hereafter:

1. Nexus program trace enabling

These steps are performed to enable the Nexus program tracing.

a) Reset the system

b) Select NPC TAP: Access to the NPC registers is enabled when the TAP controller instruction register is loaded with the NEXUS-ENABLE instruction.

c) Program NPC.PCR register for FPM/RPM, MCKO enable, …: When the NEXUS-ENABLE instruction is decoded by the TAP controller, the input port allows development tool access to all Nexus registers. Each register has a 7-bit address index. All register access is performed via the SELECT-DR-SCAN path. The Nexus Controller defaults to the REG_SELECT state when enabled. Accessing a

register requires two passes through the SELECT-DR-SCAN path: one pass to select the register and the second pass to read/write the register.

d) Select CPU TAP: CPU OnCE module is enabled by default. The OCMD JTAG register is used to shift in the OnCE instructions.

e) Enable the Nexus module: The Nexus module is enabled by loading a single instruction (NEXUS3-ACCESS) into the JTAG Instruction Register (IR) (OnCE OCMD register). Once enabled, the module will be ready to accept control input via the JTAG/OnCE pins. Each of the individual Nexus blocks on the device implements a TAP controller for accessing its registers. When Nexus has ownership of the TAP, only the block whose NEXUS-ENABLE instruction is loaded has control of the TAP. This allows the interface to all of these individual TAP controllers to appear to be a single port from outside the device. If no register is selected as the shift path for a Nexus block, that block acts like a single-bit shift register, or bypass register. Note that the OnCE is always enabled (jd_en_once is fixed to '1').

f) Program DC1 and DC2 for required trace enabling (registers are present in core)

g) Select NXMC0 and NXMC1 TAP

h) Program Nexus Development Control registers DC1 and DC2 for required trace enabling (the registers are in the CPU)

i) Program DTC for data trace enabling (in NXMC module)

j) Program DTSA/DTSE with start and end address of data to be traced (in NXMC module)

2. Nexus AF (alternate function) enabling

These steps are performed to select alternate function mapping for Nexus pads.

a) Select DCI TAP

b) Program DCI_PINCR to select which pads are to be used as Nexus pads

The DCI $\overline{\text{EVTx}}$ pin multiplexing control register (DCI_PINCR) is described below. The $\overline{\text{EVTI}}$ and $\overline{\text{EVTO}}$ functions share the same set of possible pins. If this register is written to enable both $\overline{\text{EVTI}}$ and $\overline{\text{EVTO}}$ on the same pin, the $\overline{\text{EVTI}}$ function is selected, although this type of programming should be avoided. Programming the $\overline{\text{EVTI}}$ fields to select $\overline{\text{EVTI}}$[0] input to come from more than one pin is considered a programming error and no pin is enabled.

	0	1	2	3	4	5	6	7
R W	MSEOB1E	MSEOB1D	MSEOB1C	MSEOB1B	MSEOB1A	RDYBC	RDYBB	RDYBA
Reset	0	0	0	0	0	0	0	0

	8	9	10	11	12	13	14	15
R W	MSEOB0E	MSEOB0D	MSEOB0C	MSEOB0B	MSEOB0A	MCKOE	MCKOD	MCKOC
Reset	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23
R W	MCKOB	MCKOA	MDO11C	MDO11B	MDO11A	MDO10C	MDO10B	MDO10A
Reset	0	0	0	0	0	0	0	0

	24	25	26	27	28	29	30	31
R W	MDO9B	MDO9A	MDO7D	MDO7C	MDO7B	MDO7A	MDO6B	MDO6A
Reset	0	0	0	0	0	0	0	0

	32	33	34	35	36	37	38	39
R W	MDO5B	MDO5A	MDO3E	MDO3D	MDO3C	MDO3B	MDO3A	MDO2E
Reset	0	0	0	0	0	0	0	0

	40	41	42	43	44	45	46	47
R W	MDO2D	MDO2C	MDO2B	MDO2A	MDO1E	MDO1D	MDO1C	MDO1B
Reset	0	0	0	0	0	0	0	0

	48	49	50	51	52	53	54	55
R W	MDO1A	MDO0G	MDO0F	MDO0E	MDO0D	MDO0C	MDO0B	MDO0A
Reset	0	0	0	0	0	0	0	0

	56	57	58	59	60	61	62	63
R W	EVTOBD	EVTOBC	EVTOBB	EVTOBA	EVTIB03	EVTIB02	EVTIB01	EVTIB00
Reset	0	0	0	0	0	0	0	0

Figure 1136. DCI EVTx pin multiplexing control register (DCI_PINCR)

Table 1101. DCI_PINCR register field descriptions

Field	Description
0 MSEOB1E	MSEOB1 function on port pin PH[11] (pad 123) 0 Disables MSEOB1 function on port pin PH[11] (pad 123) to Nexus 1 Enables MSEOB1 function on port pin PH[11] (pad 123) to Nexus
1 MSEOB1D	MSEOB1 function on port pin PD[4] (pad 52) 0 Disables MSEOB1 function on port pin PD[4] (pad 52) to Nexus 1 Enables MSEOB1 function on port pin PD[4] (pad 52) to Nexus
2 MSEOB1C	MSEOB1 function on port pin PE[1] (pad 65) 0 Disables MSEOB1 function on port pin PE[1] (pad 65) to Nexus 1 Enables MSEOB1 function on port pin PE[1] (pad 65) to Nexus
3 MSEOB1B	MSEOB1 function on port pin PA[14] (pad 14) 0 Disables MSEOB1 function on port pin PA[14] (pad 14) to Nexus 1 Enables MSEOB1 function on port pin PA[14] (pad 14) to Nexus
4 MSEOB1A	MSEOB1 function on port pin PH[15] (pad 127) 0 Disables MSEOB1 function on port pin PH[15] (pad 127) to Nexus 1 Enables MSEOB1 function on port pin PH[15] (pad 127) to Nexus
5 RDYBC	RDY function on port pin PH[15] (pad 127) 0 Disables RDY function on port pin PH[15] (pad 127) to Nexus 1 Enables RDY function on port pin PH[15] (pad 127) to Nexus
6 RDYBB	RDY function on port pin PI[0] (pad 128) 0 Disables RDY function on port pin PI[0] (pad 128) to Nexus 1 Enables RDY function on port pin PI[0] (pad 128) to Nexus
7 RDYBA	RDY function on port pin PC[5] (pad 37) 0 Disables RDY function on port pin PC[5] (pad 37) to Nexus 1 Enables RDY function on port pin PC[5] (pad 37) to Nexus
8 MSEOB0E	MSEOB0 function on port pin PH[3] (pad 115) 0 Disables MSEOB0 function on port pin PH[3] (pad 115) to Nexus 1 Enables MSEOB0 function on port pin PH[3] (pad 115) to Nexus
9 MSEOB0D	MSEOB0 function on port pin PD[3] (pad 51) 0 Disables MSEOB0 function on port pin PD[3] (pad 51) to Nexus 1 Enables MSEOB0 function on port pin PD[3] (pad 51) to Nexus
10 MSEOB0C	MSEOB0 function on port pin PE[13] (pad 77) 0 Disables MSEOB0 function on port pin PE[13] (pad 77) to Nexus 1 Enables MSEOB0 function on port pin PE[13] (pad 77) to Nexus
11 MSEOB0B	MSEOB0 function on port pin PE[3] (pad 67) 0 Disables MSEOB0 function on port pin PE[3] (pad 67) to Nexus 1 Enables MSEOB0 function on port pin PE[3] (pad 67) to Nexus
12 MSEOB0A	MSEOB0 function on port pin PB[1] (pad 17) 0 Disables MSEOB0 function on port pin PB[1] (pad 17) to Nexus 1 Enables MSEOB0 function on port pin PB[1] (pad 17) to Nexus

Table 1101. DCI_PINCR register field descriptions (continued)

Field	Description
13 MCKOE	MCKO function on port pin PF[6] (pad 86) 0 Disables MCKO function on port pin PF[6] (pad 86) to Nexus 1 Enables MCKO function on port pin PF[6] (pad 86) to Nexus
14 MCKOD	MCKO function on port pin PD[2] (pad 50) 0 Disables MCKO function on port pin PD[2] (pad 50) to Nexus 1 Enables MCKO function on port pin PD[2] (pad 50) to Nexus
15 MCKOC	MCKO function on port pin PE[12] (pad 76) 0 Disables MCKO function on port pin PE[12] (pad 76) to Nexus 1 Enables MCKO function on port pin PE[12] (pad 76) to Nexus
16 MCKOB	MCKO function on port pin PE[11] (pad 75) 0 Disables MCKO function on port pin PE[11] (pad 75) to Nexus 1 Enables MCKO function on port pin PE[11] (pad 75) to Nexus
17 MCKOA	MCKO function on port pin PA[5] (pad 5) 0 Disables MCKO function on port pin PA[5] (pad 5) to Nexus 1 Enables MCKO function on port pin PA[5] (pad 5) to Nexus
18 MDO11C	MDO11 function on port pin PH[15] (pad 127) 0 Disables MDO11 function on port pin PH[15] (pad 127) to Nexus 1 Enables MDO11 function on port pin PH[15] (pad 127) to Nexus
19 MDO11B	MDO11 function on port pin PH[4] (pad 116) 0 Disables MDO11 function on port pin PH[4] (pad 116) to Nexus 1 Enables MDO11 function on port pin PH[4] (pad 116) to Nexus
20 MDO11A	MDO11 function on port pin PE[2] (pad 66) 0 Disables MDO11 function on port pin PE[2] (pad 66) to Nexus 1 Enables MDO11 function on port pin PE[2] (pad 66) to Nexus
21 MDO10C	MDO10 function on port pin PH[5] (pad 117) 0 Disables MDO10 function on port pin PH[5] (pad 117) to Nexus 1 Enables MDO10 function on port pin PH[5] (pad 117) to Nexus
22 MDO10B	MDO10 function on port pin PF[2] (pad 82) 0 Disables MDO10 function on port pin PF[2] (pad 82) to Nexus 1 Enables MDO10 function on port pin PF[2] (pad 82) to Nexus
23 MDO10A	MDO10 function on port pin PE[3] (pad 67) 0 Disables MDO10 function on port pin PE[3] (pad 67) to Nexus 1 Enables MDO10 function on port pin PE[3] (pad 67) to Nexus
24 MDO9B	MDO9 function on port pin PH[6] (pad 118) 0 Disables MDO9 function on port pin PH[6] (pad 118) to Nexus 1 Enables MDO9 function on port pin PH[6] (pad 118) to Nexus
25 MDO9A	MDO9 function on port pin PE[4] (pad 68) 0 Disables MDO9 function on port pin PE[4] (pad 68) to Nexus 1 Enables MDO9 function on port pin PE[4] (pad 68) to Nexus

Table 1101. DCI_PINCR register field descriptions (continued)

Field	Description
26 MDO7D	MDO7 function on port pin PF[9] (pad 89) 0 Disables MDO7 function on port pin PF[9] (pad 89) to Nexus 1 Enables MDO7 function on port pin PF[9] (pad 89) to Nexus
27 MDO7C	MDO7 function on port pin PF[1] (pad 81) 0 Disables MDO7 function on port pin PF[1] (pad 81) to Nexus 1 Enables MDO7 function on port pin PF[1] (pad 81) to Nexus
28 MDO7B	MDO7 function on port pin PF[0] (pad 80) 0 Disables MDO7 function on port pin PF[0] (pad 80) to Nexus 1 Enables MDO7 function on port pin PF[0] (pad 80) to Nexus
29 MDO7A	MDO7 function on port pin PE[5] (pad 69) 0 Disables MDO7 function on port pin PE[5] (pad 69) to Nexus 1 Enables MDO7 function on port pin PE[5] (pad 69) to Nexus
30 MDO6B	MDO6 function on port pin PH[7] (pad 119) 0 Disables MDO6 function on port pin PH[7] (pad 119) to Nexus 1 Enables MDO6 function on port pin PH[7] (pad 119) to Nexus
31 MDO6A	MDO6 function on port pin PE[6] (pad 70) 0 Disables MDO6 function on port pin PE[6] (pad 70) to Nexus 1 Enables MDO6 function on port pin PE[6] (pad 70) to Nexus
32 MDO5B	MDO5 function on port pin PH[8] (pad 120) 0 Disables MDO5 function on port pin PH[8] (pad 120) to Nexus 1 Enables MDO5 function on port pin PH[8] (pad 120) to Nexus
33 MDO5A	MDO5 function on port pin PE[7] (pad 71) 0 Disables MDO5 function on port pin PE[7] (pad 71) to Nexus 1 Enables MDO5 function on port pin PE[7] (pad 71) to Nexus
34 MDO3E	MDO3 function on port pin PH[2] (pad 114) 0 Disables MDO3 function on port pin PH[2] (pad 114) to Nexus 1 Enables MDO3 function on port pin PH[2] (pad 114) to Nexus
35 MDO3D	MDO3 function on port pin PC[11] (pad 43) 0 Disables MDO3 function on port pin PC[11] (pad 43) to Nexus 1 Enables MDO3 function on port pin PC[11] (pad 43) to Nexus
36 MDO3C	MDO3 function on port pin PE[8] (pad 72) 0 Disables MDO3 function on port pin PE[8] (pad 72) to Nexus 1 Enables MDO3 function on port pin PE[8] (pad 72) to Nexus
37 MDO3B	MDO3 function on port pin PE[5] (pad 69) 0 Disables MDO3 function on port pin PE[5] (pad 69) to Nexus 1 Enables MDO3 function on port pin PE[5] (pad 69) to Nexus
38 MDO3A	MDO3 function on port pin PB[0] (pad 16) 0 Disables MDO3 function on port pin PB[0] (pad 16) to Nexus 1 Enables MDO3 function on port pin PB[0] (pad 16) to Nexus

Table 1101. DCI_PINCR register field descriptions (continued)

Field	Description
39 MDO2E	MDO2 function on port pin PF[13] (pad 93) 0 Disables MDO2 function on port pin PF[13] (pad 93) to Nexus 1 Enables MDO2 function on port pin PF[13] (pad 93) to Nexus
40 MDO2D	MDO2 function on port pin PC[9] (pad 41) 0 Disables MDO2 function on port pin PC[9] (pad 41) to Nexus 1 Enables MDO2 function on port pin PC[9] (pad 41) to Nexus
41 MDO2C	MDO2 function on port pin PE[9] (pad 73) 0 Disables MDO2 function on port pin PE[9] (pad 73) to Nexus 1 Enables MDO2 function on port pin PE[9] (pad 73) to Nexus
42 MDO2B	MDO2 function on port pin PE[6] (pad 70) 0 Disables MDO2 function on port pin PE[6] (pad 70) to Nexus 1 Enables MDO2 function on port pin PE[6] (pad 70) to Nexus
43 MDO2A	MDO2 function on port pin PA[15] (pad 15) 0 Disables MDO2 function on port pin PA[15] (pad 15) to Nexus 1 Enables MDO2 function on port pin PA[15] (pad 15) to Nexus
44 MDO1E	MDO1 function on port pin PF[12] (pad 92) 0 Disables MDO1 function on port pin PF[12] (pad 92) to Nexus 1 Enables MDO1 function on port pin PF[12] (pad 92) to Nexus
45 MDO1D	MDO1 function on port pin PC[7] (pad 39) 0 Disables MDO1 function on port pin PC[7] (pad 39) to Nexus 1 Enables MDO1 function on port pin PC[7] (pad 39) to Nexus
46 MDO1C	MDO1 function on port pin PE[10] (pad 74) 0 Disables MDO1 function on port pin PE[10] (pad 74) to Nexus 1 Enables MDO1 function on port pin PE[10] (pad 74) to Nexus
47 MDO1B	MDO1 function on port pin PE[7] (pad 71) 0 Disables MDO1 function on port pin PE[7] (pad 71) to Nexus 1 Enables MDO1 function on port pin PE[7] (pad 71) to Nexus
48 MDO1A	MDO1 function on port pin PA[10] (pad 10) 0 Disables MDO1 function on port pin PA[10] (pad 10) to Nexus 1 Enables MDO1 function on port pin PA[10] (pad 10) to Nexus
49 MDO0G	MDO0 function on port pin PF[10] (pad 90) 0 Disables MDO0 function on port pin PF[10] (pad 90) to Nexus 1 Enables MDO0 function on port pin PF[10] (pad 90) to Nexus
50 MDO0F	MDO0 function on port pin PD[6] (pad 54) 0 Disables MDO0 function on port pin PD[6] (pad 54) to Nexus 1 Enables MDO0 function on port pin PD[6] (pad 54) to Nexus
51 MDO0E	MDO0 function on port pin PC[10] (pad 42) 0 Disables MDO0 function on port pin PC[10] (pad 42) to Nexus 1 Enables MDO0 function on port pin PC[10] (pad 42) to Nexus

Table 1101. DCI_PINCR register field descriptions (continued)

Field	Description
52 MDO0D	MDO0 function on port pin PC[5] (pad 37) 0 Disables MDO0 function on port pin PC[5] (pad 37) to Nexus 1 Enables MDO0 function on port pin PC[5]] (pad 37) to Nexus
53 MDO0C	MDO0 function on port pin PE[11] (pad 75) 0 Disables MDO0 function on port pin PE[11] (pad 75) to Nexus 1 Enables MDO0 function on port pin PE[11] (pad 75) to Nexus
54 MDO0B	MDO0 function on port pin PE[8] (pad 72) 0 Disables MDO0 function on port pin PE[8] (pad 72) to Nexus 1 Enables MDO0 function on port pin PE[8] (pad 72) to Nexus
55 MDO0A	MDO0 function on port pin PA[6] (pad 6) 0 Disables MDO0 function on port pin PA[6] (pad 6) to Nexus 1 Enables MDO0 function on port pin PA[6] (pad 6) to Nexus
56 EVTOBD	EVTO function on port pin PH[9] (pad 121) 0 Disables EVTO function on port pin PH[9] (pad 121) to Nexus 1 Enables EVTO function on port pin PH[9] (pad 121) to Nexus
57 EVTOBC	EVTO function on port pin PF[0] (pad 80) 0 Disables EVTO function on port pin PF[0] (pad 80) to Nexus 1 Enables EVTO function on port pin PF[0] (pad 80) to Nexus
58 EVTOBB	EVTO function on port pin PE[14] (pad 78) 0 Disables EVTO function on port pin PE[14] (pad 78) to Nexus 1 Enables EVTO function on port pin PE[14] (pad 78) to Nexus
59 EVTOBA	EVTO function on port pin PA[2] (pad 2) 0 Disables EVTO function on port pin PA[2] (pad 2) to Nexus 1 Enables EVTO function on port pin PA[2] (pad 2) to Nexus
60 EVTIB03	EVTI function on port pin PH[1] (pad 113) 0 Disables EVTI function on port pin PH[1] (pad 113) to Nexus 1 Enables EVTI function on port pin PH[1] (pad 113) to Nexus
61 EVTIB02	EVTI function on port pin PH[0] (pad 112) 0 Disables EVTI function on port pin PH[0] (pad 112) to Nexus 1 Enables EVTI function on port pin PH[0] (pad 112) to Nexus
62 EVTIB01	EVTI function on port pin PE[15] (pad 79) 0 Disables EVTI function on port pin PE[15] (pad 79) to Nexus 1 Enables EVTI function on port pin PE[15] (pad 79) to Nexus
63 EVTIB00	EVTI function on port pin PA[1] (pad 1) 0 Disables EVTI function on port pin PA[1] (pad 1) to Nexus 1 Enables EVTI function on port pin PA[1] (pad 1) to Nexus

55.6 Functionality

55.6.1 JTAG Master (JTAGM)

The SPC574Sx Nexus infrastructure is configurable by software from within the device.

The JTAGM module drives the JTAG signals from within the device and provides a parallel interface that can exchange data via customer software or another serial communication module.

The JTAGM is connected to the IPS interface. The data transferred to this module drives the TDI and TMS outputs, and accepts TDO as input to and from the DCI.

The JTAGM module also allows control of the EVTO and EVTI signals. Asserting EVTI in the JTAGM triggers the debug mode in the DCI.

55.6.2 Nexus Crossbar Multimaster Trace Client (NXMC)

The NXMC module provides real time trace capabilities for peripherals and works in conjunction with the Nexus Auxiliary Port. This module is present twice in the device to cover all master accesses to memories and allow external memory reconstruction by external measurement and online data acquisition tools.

- FlexRay NXMC: it is used to trace FlexRay accesses.
- DMA Master Port NXMC: A second NXMC module is used to trace the DMA accesses.

The NXMC clients are connected to and interact with the following modules:

- DCI: TAP controller interface for configuration via JTAG and EVTO management
- SPU: trace control
- NPC: arbitration for access to NAP and MDO pads
- AHB bus: trace source, connected to the data snoop block

The NXMC peripheral interface block is unused, since the module is used for data snooping on the AHB bus only.

55.6.3 JTAG Data Communication module (JDC)

The JDC provides the functionality to move register data between the JTAG and the IPS domain. It consists of JTAG accessible and IPS accessible registers and the associated logic to coordinate the transfers. The module connects to the peripheral bridge and the DCI.

55.6.4 Nexus Port Controller (NPC)

The Nexus Port Controller controls the usage and sharing of the Nexus Auxiliary Port between Nexus clients. The Nexus Auxiliary Port is accessed by:

- The N3 block of the CPU
- The DMA master port NXMC client
- The FlexRay master port NXMC client

The NPC is placed in disabled mode upon exit of reset. Access to the NPC registers is enabled when the TAP controller instruction register is loaded with the NEXUS-ENABLE instruction. Once the NPC is enabled, if message transmission via the auxiliary port is

desired, a write to the PCR is then required to enable the NPC and select the mode of operation. Asserting MCKO_EN places the NPC in enabled mode and enables MCKO. The frequency of MCKO is selected by writing the MCKO_DIV field. Asserting or negating the FPM bit selects full-port or reduced-port mode, respectively.

55.6.5 Sequence Processing Unit (SPU)

The SPU is able to generate complex debug events, based upon input events from sources throughout the SOC. It generates trace control events for data, program, ownership, and watchdog trace (DTM, PTM, OTM and WTM), as well as EVTO, HALT and debug events for the clients.

On SPC574Sx the required client interfaces are:

- One e200z420 client
- Two NXMC clients

Furthermore, the SPU interfaces with the following modules:

- DCI for debug control and EVTO handling
- JTAGM for generating CPU interrupt via the INTC

56 JTAG Controller (JTAGC)

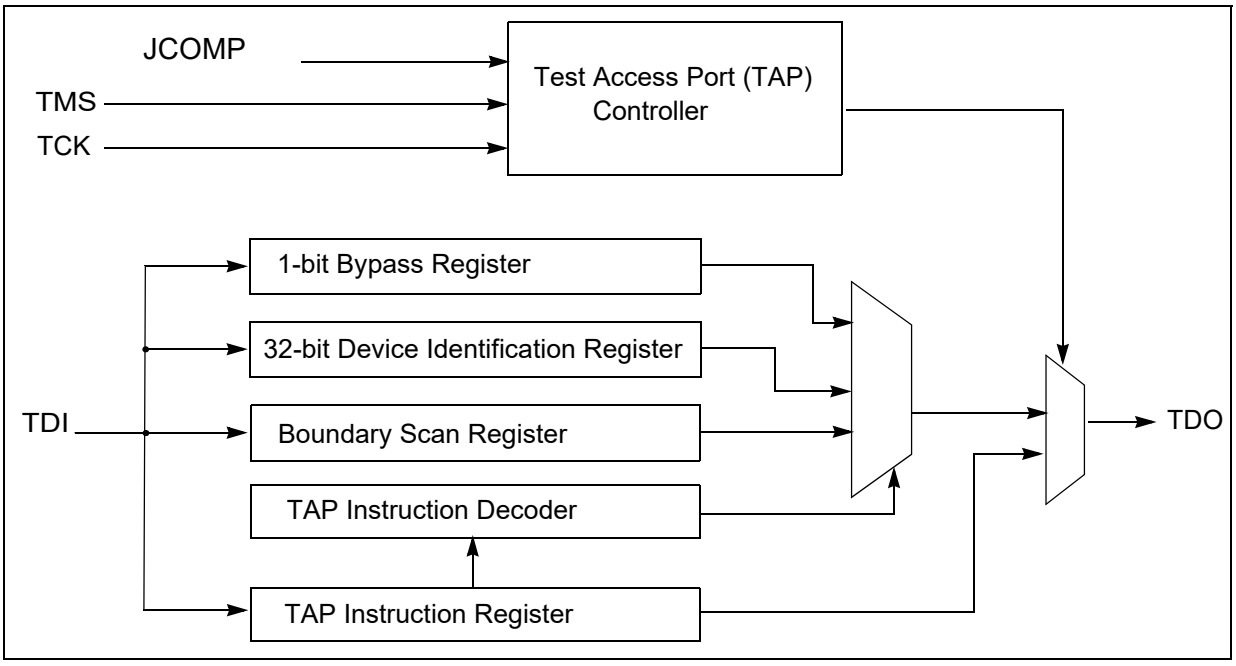
56.1 Introduction

The JTAGC block provides the means to test chip functionality and connectivity while remaining transparent to system logic when not in test mode. Testing is performed via a boundary scan technique, as defined in the IEEE 1149.1-2001 standard. All data input to and output from the JTAGC block is communicated in serial format.

56.1.1 Block Diagram

The following is a block diagram of the JTAG Controller (JTAGC) block.

Figure 1137. JTAG (IEEE 1149.1) block diagram



56.1.2 Features

The JTAGC block is compliant with the IEEE 1149.1-2001 standard, and supports the following features:

- IEEE 1149.1-2001 Test Access Port (TAP) interface
  - 4 pins (TDI, TMS, TCK, and TDO)
- JCOMP input that provides reset control and the ability to share the TAP.
- Instruction register that supports several IEEE 1149.1-2001 defined instructions as well as several public and private device-specific instructions. Refer to *Table 1108* for a list of supported instructions.
- Sharing of the TAP with other TAP controllers via ACCESS_AUX_x instructions
- TEST_CTRL register
- JTAG_PASSWORD register
- DATA1 register
- DATA2 register
- Data registers, bypass register, boundary scan register, and device identification register.
- TAP controller state machine that controls the operation of the data registers, instruction register and associated circuitry.

56.1.3 Modes of operation

The JTAGC block uses JCOMP and a power-on reset indication as its primary reset signals. Several IEEE 1149.1-2001 defined test modes are supported, as well as a bypass mode.

56.1.3.1 Reset

The JTAGC block is placed in reset when either power-on reset is asserted, JCOMP is negated, or the TMS input is held high for enough consecutive rising edges of TCK to sequence the TAP controller state machine into the Test-Logic-Reset state. Holding TMS high for five consecutive rising edges of TCK guarantees entry into the Test-Logic-Reset state regardless of the current TAP controller state. Asserting power-on reset or setting JCOMP to a value other than the value required to enable the JTAGC block results in asynchronous entry into the reset state. While in reset, the following actions occur:

- The TAP controller is forced into the Test-Logic-Reset state, thereby disabling the test logic and allowing normal operation of the on-chip system logic to continue unhindered.
- The instruction register is loaded with the IDCODE instruction.

56.1.3.2 IEEE 1149.1-2001 defined test modes

The JTAGC block supports several IEEE 1149.1-2001 defined test modes. A test mode is selected by loading the appropriate instruction into the instruction register while the JTAGC is enabled. Supported test instructions include EXTEST, HIGHZ, CLAMP, SAMPLE and SAMPLE/PRELOAD. Each instruction defines the set of data register(s) that may operate and interact with the on-chip system logic while the instruction is current. Only one test data register path is enabled to shift data between TDI and TDO for each instruction.

The boundary scan register is enabled for serial access between TDI and TDO when the EXTEST, SAMPLE or SAMPLE/PRELOAD instructions are active. The single-bit bypass register shift stage is enabled for serial access between TDI and TDO when the BYPASS,

HIGHZ, CLAMP or reserved instructions are active. The functionality of each test mode is explained in more detail in *Section 56.4.4*.

56.1.3.3 Bypass mode

When no test operation is required, the BYPASS instruction can be loaded to place the JTAGC block into bypass mode. While in bypass mode, the single-bit bypass shift register is used to provide a minimum-length serial path to shift data between TDI and TDO.

56.2 External signal description

The JTAGC consists of five signals that connect to off chip development tools and allow access to test support functions. The JTAGC signals are outlined in the following table and described in the following sections.

Table 1102. JTAG signal properties

Name	I/O	Function	Reset State	Pull
TCK	Input	Test Clock	—	Down
TDI	Input	Test Data In	—	Up
TDO	Output	Test Data Out	High Z(1)	—
TMS	Input	Test Mode Select	—	Up
JCOMP	Input	JTAG Compliancy	—	Down

1. TDO output buffer enable is negated when the JTAGC is not in the Shift-IR or Shift-DR states. A weak pull may be implemented at the TDO pad for use when JTAGC is inactive.

56.2.1 TCK—Test clock input

Test Clock Input (TCK) is an input pin used to synchronize the test logic and control register access through the TAP.

56.2.2 TDI—Test data input

Test Data Input (TDI) is an input pin that receives serial test instructions and data. TDI is sampled on the rising edge of TCK.

56.2.3 TDO—Test data output

Test Data Output (TDO) is an output pin that transmits serial output for test instructions and data. TDO is three-stateable and is actively driven only in the Shift-IR and Shift-DR states of the TAP controller state machine, which is described in *Section 56.4.3*.

56.2.4 TMS—Test mode select

Test Mode Select (TMS) is an input pin used to sequence the IEEE 1149.1-2001 test control state machine. TMS is sampled on the rising edge of TCK.

56.2.5 JCOMP—JTAG compliancy

The JCOMP signal provides IEEE 1149.1-2001 compatibility and provides the ability to share the TAP. The JTAGC TAP controller is enabled when JCOMP is set to the JTAGC enable encoding, otherwise the JTAGC TAP controller remains in reset.

56.3 Register description

This section provides a detailed description of the JTAGC block registers accessible through the TAP interface, including data registers and the instruction register. Individual bit-level descriptions and reset states of each register are included. These registers are not memory-mapped and can only be accessed through the TAP.

56.3.1 Instruction register

The JTAGC block uses a 6-bit instruction register as shown in the following figure. The instruction register allows instructions to be loaded into the block to select the test to be performed or the test data register to be accessed or both. Instructions are shifted in through TDI while the TAP controller is in the Shift-IR state, and latched on the falling edge of TCK in the Update-IR state. The latched instruction value can only be changed in the Update-IR and Test-Logic-Reset TAP controller states. Synchronous entry into the Test-Logic-Reset state results in the IDCODE instruction being loaded on the falling edge of TCK. Asynchronous entry into the Test-Logic-Reset state results in asynchronous loading of the IDCODE instruction. During the Capture-IR TAP controller state, the instruction shift register is loaded with the value 000001b, making this value the register's read value when the TAP controller is sequenced into the Shift-IR state.

	0	1	2	3	4	5
R	0	0	0	0	0	1
W	Instruction Code					
Reset:	0	0	0	0	0	1

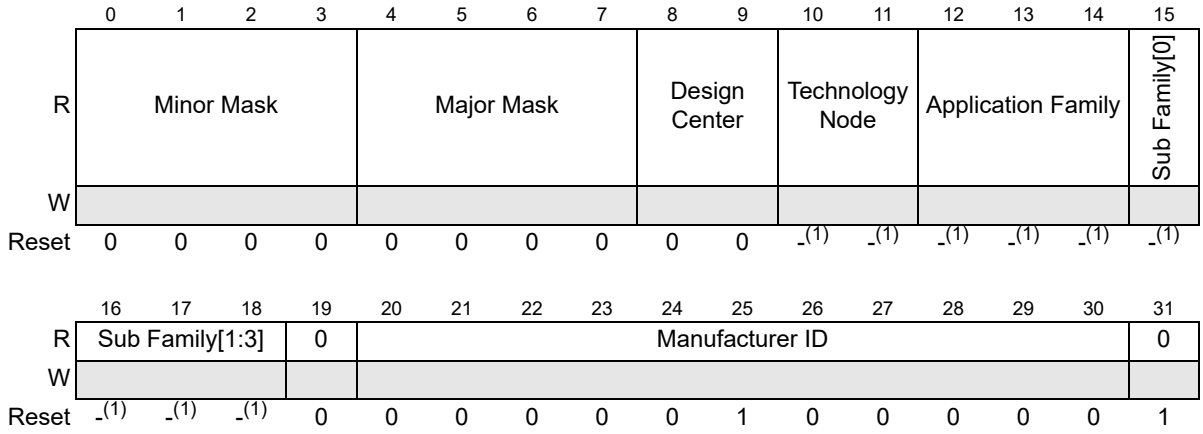
Figure 1138. 6-bit instruction register

56.3.2 Bypass register

The bypass register is a single-bit shift register path selected for serial data transfer between TDI and TDO when the BYPASS, CLAMP, HIGHZ or reserve instructions are active. After entry into the Capture-DR state, the single-bit shift register is set to a logic 0. Therefore, the first bit shifted out after selecting the bypass register is always a logic 0.

56.3.3 Device identification register

The device identification (JTAG ID) register, shown in the following figure, allows the revision number, part number, manufacturer, and design center responsible for the design of the part to be determined through the TAP. The device identification register is selected for serial data transfer between TDI and TDO when the IDCODE instruction is active. Entry into the Capture-DR state while the device identification register is selected loads the IDCODE into the shift register to be shifted out on TDO in the Shift-DR state. No action occurs in the Update-DR state.



1. The value is shown in *Table 1103*

Figure 1139. Device identification register

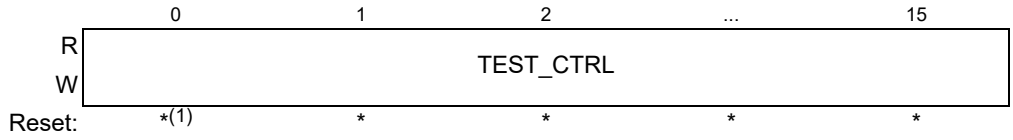
The following table describes the device identification register functions.

Table 1103. Device identification register field descriptions

Field	Description
0:3 Minor Mask	Minor Mask These bits contain the minor mask of the part
4:7 Major Mask	Major Mask These bits contain the major mask of the part
8:9 Design Center	Design Center These bits indicate the device design center
10:11 Technology Node	Technology Node These bits indicate the device technology node 00 55nm
12:14 Application Family	Application Family These bits indicate the device application family 101 SPC57S - Chassis & Safety 55nm
15:18 Sub Family	Sub Family These bits indicate the device sub family 0100 SPC574S60
20:30 Manufacturer ID	Manufacturer Identity Code These bits contain the reduced Joint Electron Device Engineering Council (JEDEC) ID 00000100000: STMicroelectronics

56.3.4 TEST_CTRL Register

The TEST_CTRL register is a 15-bit shift register path from TDI to TDO selected when the ENABLE_TEST_CTRL instruction is active. The TEST_CTRL register transfers its value to a parallel hold register on the rising edge of TCK when the TAP controller state machine is in the Update-DR state.



1. The reset value of TEST_CTRL is 0x1B

Figure 1140. TEST_CTRL Register

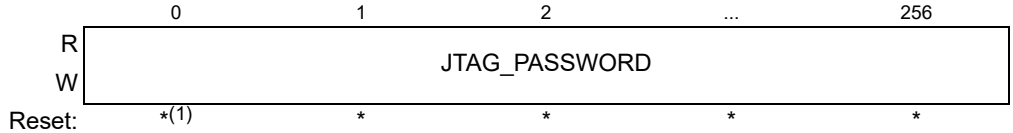
The following table describes the TEST_CTRL register functions.

Table 1104. TEST_CTRL register field descriptions

Field	Description
TEST_CTRL	Test Control. Used to control chiptop test functions.

56.3.5 JTAG_PASSWORD register

The JTAG_PASSWORD register is a 256-bit shift register path from TDI to TDO selected when the ENABLE_JTAG_PASSWORD instruction is active. The default reset value of the JTAG_PASSWORD register is 256'b0. The JTAG_PASSWORD register transfers its value to a parallel hold register on the rising edge of TCK when the TAP controller state machine is in the Update-DR state. Once the ENABLE_JTAG_PASSWORD instruction is executed, the register value remains valid until a JTAG reset occurs. The operation of this register is described in the security documentation.



1. The reset value of JTAG_PASSWORD is 256'b0.

Figure 1141. JTAG_PASSWORD register

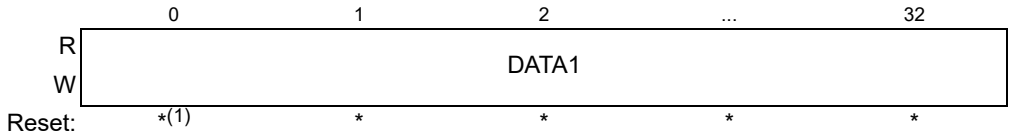
The following table describes the JTAG_PASSWORD register functions.

Table 1105. JTAG_PASSWORD register field descriptions

Field	Description
JTAG_PASSWORD	JTAG Password. The JTAG_PASSWORD bits are used to provide the JTAG password for security.

56.3.6 DATA1 register

The DATA1 register is a 32-bit shift register path from TDI to TDO selected when the ENABLE_DATA1 instruction is active. The default reset value of the DATA1 register is 32'b0. The DATA1 register transfers its value to a parallel hold register on the rising edge of TCK when the TAP controller state machine is in the Update-DR state. Once the ENABLE_DATA1 instruction is executed, the register value remains valid until a JTAG reset occurs.



1. The reset value of DATA1 is 32’b0.

Figure 1142. DATA1 register

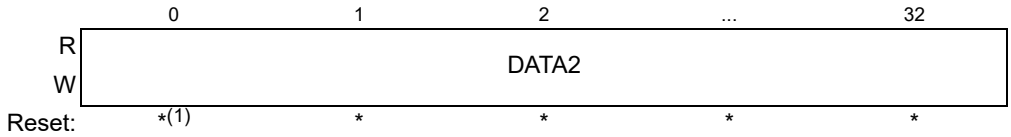
The following table describes the DATA1 register functions.

Table 1106. DATA1 register field descriptions

Field	Description
DATA1	JTAG data register 1.

56.3.7 DATA2 register

The DATA2 register is a 32-bit shift register path from TDI to TDO selected when the ENABLE_DATA2 instruction is active. The default reset value of the DATA2 register is 32’b0. The DATA2 register transfers its value to a parallel hold register on the rising edge of TCK when the TAP controller state machine is in the Update-DR state. Once the ENABLE_DATA2 instruction is executed, the register value remains valid until a JTAG reset occurs.



1. The reset value of DATA2 is 32’b0.

Figure 1143. DATA2 register

The following table describes the DATA2 register functions.

Table 1107. DATA2 register field descriptions

Field	Description
DATA2	JTAG data register 2.

56.3.8 Boundary scan register

The boundary scan register is connected between TDI and TDO when the EXTEST, SAMPLE or SAMPLE/PRELOAD instructions are active. It is used to capture input pin data, force fixed values on output pins, and select a logic value and direction for bidirectional pins. Each bit of the boundary scan register represents a separate boundary scan register cell, as described in the IEEE 1149.1-2001 standard and discussed in *Section 56.4.5*. The size of the boundary scan register and bit ordering is device-dependent and can be found in the device BSDL file.

56.4 Functional description

This section explains the JTAGC functional description.

56.4.1 JTAGC reset configuration

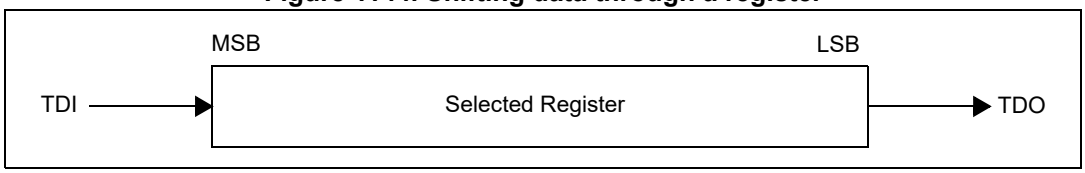
While in reset, the TAP controller is forced into the Test-Logic-Reset state, thus disabling the test logic and allowing normal operation of the on-chip system logic. In addition, the instruction register is loaded with the IDCODE instruction.

56.4.2 IEEE 1149.1-2001 (JTAG) Test Access Port

The JTAGC block uses the IEEE 1149.1-2001 TAP for accessing registers. This port can be shared with other TAP controllers on the MCU. Ownership of the port is determined by the value of the currently loaded instruction. For more detail on TAP sharing via JTAGC instructions refer to *Section 56.4.4.12*.

Data is shifted between TDI and TDO through the selected register starting with the least significant bit, as illustrated in the following figure. This applies for the instruction register, test data registers, and the bypass register.

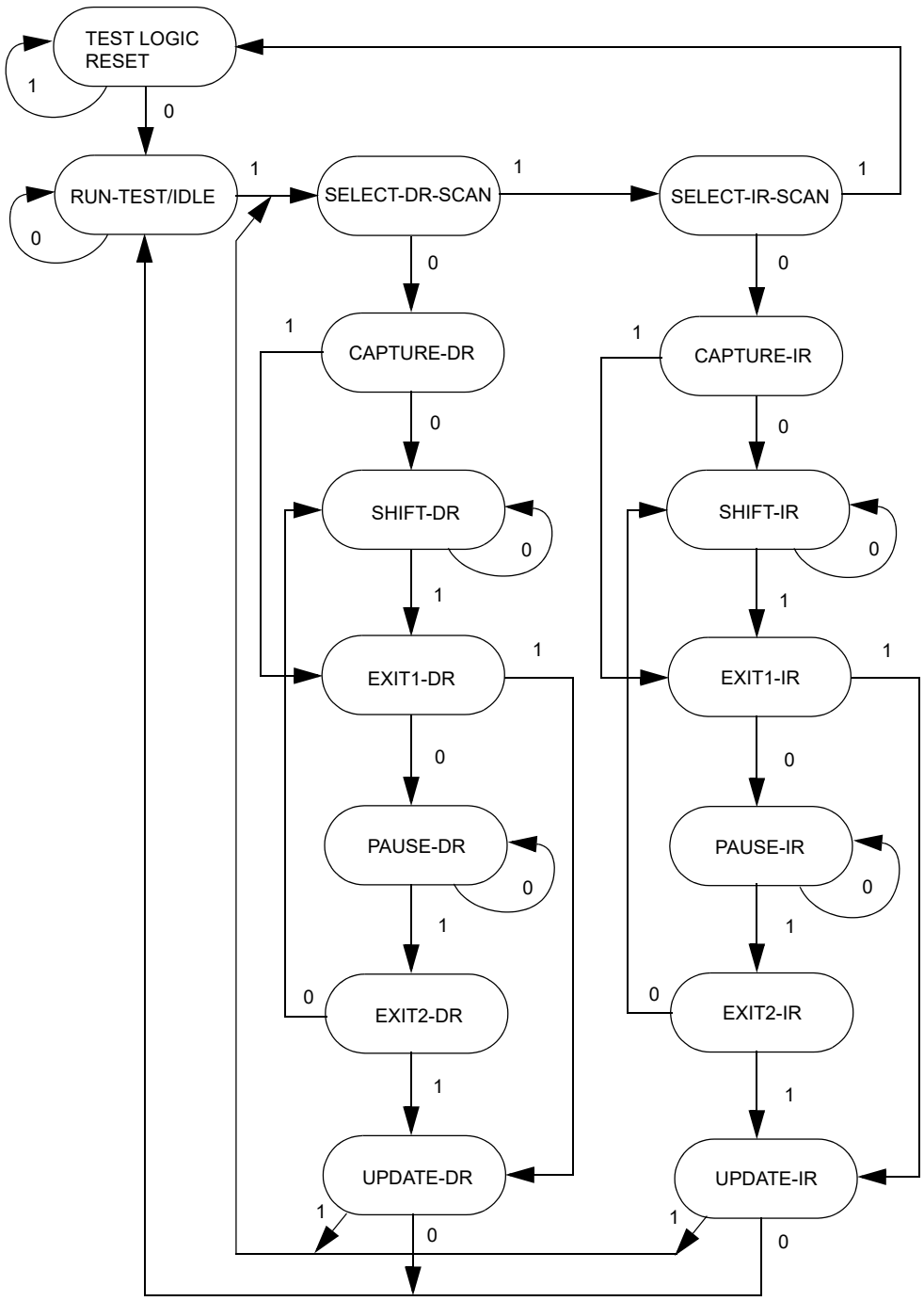
Figure 1144. Shifting data through a register



56.4.3 TAP controller state machine

The TAP controller is a synchronous state machine that interprets the sequence of logical values on the TMS pin. The following figure shows the machine's states. The value shown next to each state is the value of the TMS signal sampled on the rising edge of the TCK signal. As the following figure shows, holding TMS at logic 1 while clocking TCK through a sufficient number of rising edges also causes the state machine to enter the Test-Logic-Reset state.

Figure 1145. IEEE 1149.1-2001 TAP controller finite state machine



Note:

1. The value shown adjacent to each state transition represents the value of TMS at the time of a rising edge of TCK.

56.4.3.1 Enabling the TAP controller

The JTAGC TAP controller is enabled by setting JCOMP to a logic 1 value.

56.4.3.2 Selecting an IEEE 1149.1-2001 register

Access to the JTAGC data registers is achieved by loading the instruction register with any of the JTAGC block instructions while the JTAGC is enabled. Instructions are shifted in via the Select-IR-Scan path and loaded in the Update-IR state. At this point, all data register access is performed via the Select-DR-Scan path.

The Select-DR-Scan path is used to read or write the register data by shifting in the data (LSB first) during the Shift-DR state. When reading a register, the register value is loaded into the IEEE 1149.1-2001 shifter during the Capture-DR state. When writing a register, the value is loaded from the IEEE 1149.1-2001 shifter to the register during the Update-DR state. When reading a register, there is no requirement to shift out the entire register contents. Shifting may be terminated once the required number of bits have been acquired.

56.4.4 JTAGC block instructions

The JTAGC block implements the IEEE 1149.1-2001 defined instructions listed in the following table. This section gives an overview of each instruction; refer to the IEEE 1149.1-2001 standard for more details. All undefined opcodes are reserved.

Table 1108. 6-bit JTAG instructions

Instruction	Code[5:0]	Instruction Summary
IDCODE	000001	Selects device identification register for shift
SAMPLE/PRELOAD	000010	Selects boundary scan register for shifting, sampling, and preloading without disturbing functional operation
SAMPLE	000011	Selects boundary scan register for shifting and sampling without disturbing functional operation
EXTEST	000100	Selects boundary scan register while applying preloaded values to output pins and asserting functional reset
TEST_LEAKAGE(1),(2)	000101	Selects bypass register while tristating all output pins and asserts the jtag_leakage signal
ENABLE_TEST_CTRL(1)	000110	Selects TEST_CTRL register
ENABLE_JTAG_PASSWORD	000111	Selects JTAG_PASSWORD register
HIGHZ	001001	Selects bypass register while three-stating all output pins and asserting functional reset
Reserved	001010	Reserved
CLAMP	001100	Selects bypass register while applying preloaded values to output pins and asserting functional reset
ENABLE_DCI_CR	001110	Enables access to DCI CR register
ENABLE_DATA1	001110	Enables access to DATA1 register
ENABLE_DCI_PINCR	001111	Enables access to DCI PINCR register
ENABLE_DATA2	001111	Enables access to DATA2 register
ACCESS_AUX_x	100000-111110	Grants one of the auxiliary TAP controllers ownership of the TAP as shown in the cells below.

Table 1108. 6-bit JTAG instructions (continued)

Instruction	Code[5:0]	Instruction Summary
Reserved	100000	Reserved
ACCESS_AUX_NPC_PD	100001	Enables access to the NPC TAP controller on production device
Reserved	100001	Reserved
Reserved	100010	Reserved
Reserved	100011	Reserved
Reserved	100100	Reserved
ACCESS_AUX_TCU	100101	Enables access to the TCU TAP controller
ACCESS_AUX_JDC	100110	Enables access to the JDC TAP controller
Reserved	100111	Reserved
ACCESS_AUX_CORE_0	101000	Enabled access to E200 Core 0
Reserved	101001	Reserved
Reserved	101010	Reserved
Reserved	101011	Reserved
Reserved	101100	Reserved
Reserved	101101	Reserved
Reserved	101110	Reserved
Reserved	101111	Reserved
Reserved	110000	Reserved
Reserved	110001	Reserved
Reserved	110010	Reserved
Reserved	110011	Reserved
ACCESS_AUX_BUS_MON_0	110100	Enables access to NXMC Client 0 TAP controller
ACCESS_AUX_BUS_MON_1	110101	Enables access to NXMC Client 1 TAP controller
Reserved	110110	Reserved
Reserved	110111	Reserved
Reserved	111000	Reserved
Reserved	111001	Reserved
ACCESS_AUX_SPU_A	111010	Enables access to SPU Client TAP controller
Reserved	111011	Reserved
ACCESS_AUX_PARALLEL(3)	111100	Enables access to all cores in parallel
Reserved	111101	Reserved
ACCESS_AUX_DCI_PD	111110	Enables access to pass control from BD to PD

Table 1108. 6-bit JTAG instructions (continued)

Instruction	Code[5:0]	Instruction Summary
BYPASS	111111	Selects bypass register for data operations
Reserved(4)	All other opcodes	Decoded to select bypass register

1. Instruction for manufacturing purposes only
2. Assertion of power-on reset or setting JCOMP for non-JTAGC operation required to exit this instruction
3. The use of this instruction is limited to the synchronous exit of debug mode via the execution of the GO+EXIT OnCE command. The execution of other instructions or commands is not supported
4. The manufacturer reserves the right to change the decoding of reserved instruction codes in the future

56.4.4.1 IDCODE instruction

IDCODE selects the 32-bit device identification register as the shift path between TDI and TDO. This instruction allows interrogation of the MCU to determine its version number and other part identification data. IDCODE is the instruction placed into the instruction register when the JTAGC block is reset.

56.4.4.2 SAMPLE/PRELOAD instruction

The SAMPLE/PRELOAD instruction has two functions:

- First, the SAMPLE portion of the instruction obtains a sample of the system data and control signals present at the MCU input pins and just before the boundary scan register cells at the output pins. This sampling occurs on the rising edge of TCK in the Capture-DR state when the SAMPLE/PRELOAD instruction is active. The sampled data is viewed by shifting it through the boundary scan register to the TDO output during the Shift-DR state. Both the data capture and the shift operation are transparent to system operation.
- Secondly, the PRELOAD portion of the instruction initializes the boundary scan register cells before selecting the EXTEST or CLAMP instructions to perform boundary scan tests. This is achieved by shifting in initialization data to the boundary scan register during the Shift-DR state. The initialization data is transferred to the parallel outputs of the boundary scan register cells on the falling edge of TCK in the Update-DR state. The data is applied to the external output pins by the EXTEST or CLAMP instruction. System operation is not affected.

56.4.4.3 SAMPLE instruction

The SAMPLE instruction obtains a sample of the system data and control signals present at the MCU input pins and just before the boundary scan register cells at the output pins. This sampling occurs on the rising edge of TCK in the Capture-DR state when the SAMPLE instruction is active. The sampled data is viewed by shifting it through the boundary scan register to the TDO output during the Shift-DR state. There is no defined action in the Update-DR state. Both the data capture and the shift operation are transparent to system operation.

56.4.4.4 EXTEST—External test instruction

EXTEST selects the boundary scan register as the shift path between TDI and TDO. It allows testing of off-chip circuitry and board-level interconnections by driving preloaded data contained in the boundary scan register onto the system output pins. Typically, the

preloaded data is loaded into the boundary scan register using the SAMPLE/PRELOAD instruction before the selection of EXTEST. EXTEST asserts the internal system reset for the MCU to force a predictable internal state while performing external boundary scan operations.

56.4.4.5 TEST_LEAKAGE instruction

The TEST_LEAKAGE instruction forces the jtag_leakage output signal to high. It is intended to tristate all output pad buffers and disable all of the part's pad input buffers except JCOMP and TEST. The jtag_leakage signal is asserted at the falling edge of TCK following the TAP controller state machine transition from the Update-IR to the Run-Test-Idle state. Once asserted, the part disables TCK, TMS, and TDI inputs and forces them to a logic 1. The TAP controller state machine remains in the Run-Test-Idle state until the JCOMP input is set to a value other than the JTAGC enable encoding. TEST_LEAKAGE also asserts the internal system reset for the MCU to force a predictable internal state.

56.4.4.6 ENABLE_TEST_CTRL instruction

The ENABLE_TEST_CTRL instruction selects the TEST_CTRL register for connection as the shift path between TDI and TDO.

56.4.4.7 ENABLE_JTAG_PASSWORD instruction

The ENABLE_JTAG_PASSWORD instruction selects the JTAG_PASSWORD register for connection as the shift path between TDI and TDO.

56.4.4.8 ENABLE_DATA1 instruction

The ENABLE_DATA1 instruction selects the DATA1 register for connection as the shift path between TDI and TDO.

56.4.4.9 ENABLE_DATA2 instruction

The ENABLE_DATA2 instruction selects the DATA2 register for connection as the shift path between TDI and TDO.

56.4.4.10 HIGHZ instruction

HIGHZ selects the bypass register as the shift path between TDI and TDO. While HIGHZ is active all output drivers are placed in an inactive drive state (for example high impedance). HIGHZ also asserts the internal system reset for the MCU to force a predictable internal state.

56.4.4.11 CLAMP instruction

CLAMP allows the state of signals driven from MCU pins to be determined from the boundary scan register while the bypass register is selected as the serial path between TDI and TDO. CLAMP enhances test efficiency by reducing the overall shift path to a single bit (the bypass register) while conducting an EXTEST type of instruction through the boundary scan register. CLAMP also asserts the internal system reset for the MCU to force a predictable internal state.

56.4.4.12 ACCESS_AUX_x instructions

The JTAGC is configurable to allow other TAP controllers on the device to share the port with it. This is done by providing ACCESS_AUX_x instructions for each of these TAP controllers. When this instruction is loaded, control of the JTAG pins are transferred to the selected TAP controller. Any data input via TDI and TMS is passed to the selected TAP controller, and any TDO output from the selected TAP controller is sent back to the JTAGC to be output on the pins. The JTAGC regains control of the JTAG port during the UPDATE-DR state if the PAUSE-DR state was entered. Auxiliary TAP controllers are held in RUN-TEST/IDLE while they are inactive. Instructions not used to access an auxiliary TAP controller on a device are treated like the BYPASS instruction.

56.4.4.13 BYPASS instruction

BYPASS selects the bypass register, creating a single-bit shift register path between TDI and TDO. BYPASS enhances test efficiency by reducing the overall shift path when no test operation of the MCU is required. This allows more rapid movement of test data to and from other components on a board that are required to perform test functions. While the BYPASS instruction is active the system logic operates normally.

56.4.5 Boundary scan

The boundary scan technique allows signals at component boundaries to be controlled and observed through the shift-register stage associated with each pad. Each stage is part of a larger boundary scan register cell, and cells for each pad are interconnected serially to form a shift-register chain around the border of the design. The boundary scan register consists of this shift-register chain, and is connected between TDI and TDO when the EXTEST, SAMPLE, or SAMPLE/PRELOAD instructions are loaded. The shift-register chain contains a serial input and serial output, as well as clock and control signals.

56.5 Initialization/Application information

The test logic is a static logic design, and TCK can be stopped in either a high or low state without loss of data. However, the system clock is not synchronized to TCK internally. Any mixed operation using both the test logic and the system functional logic requires external synchronization.

To initialize the JTAGC block and enable access to registers, must be loaded the appropriate instruction for the test or action to be performed.

57 JTAG Data Communication (JDC)

57.1 Introduction

The JTAG Data Communication (JDC) module provides the capability to move register data between the IPS and JTAG domains. This facilitates communication between internal resources that access memory mapped register space and an external tool that accesses the JTAG port.

57.2 Overview

The JDC module consists of IPS accessible registers, JTAG accessible registers, and associated logic to coordinate movement of data from one register domain to another.

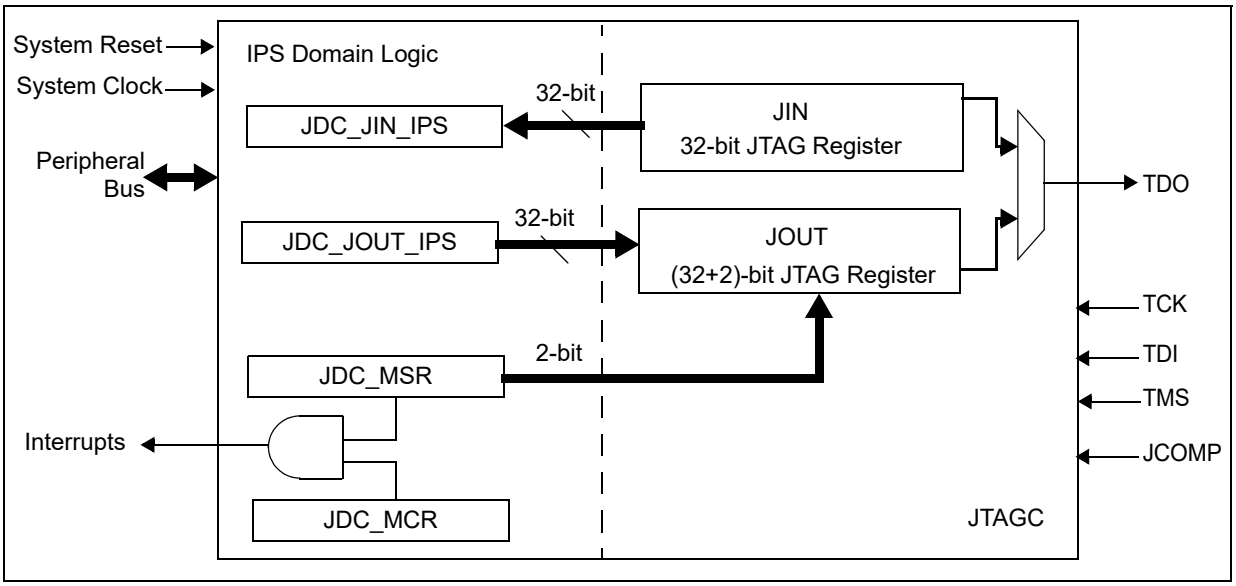
The JDC implements the following IPS data registers, occupying separate memory space:

- 32-bit memory mapped register that can be read or written via IPS (JDC_JOUT_IPS), and whose contents are ported out for capture into a JTAG register (JOUT) to be read via the JTAG port.
- 32-bit memory mapped register that can only be read via IPS (JDC_JIN_IPS), and whose contents are loaded from a JTAG register (JIN).

In addition to the data registers themselves, logic is implemented to indicate when new data has been shifted in via the JTAG port and is ready to be read from the JDC_JIN_IPS register, and when new data has been written to the JDC_JOUT_IPS register and is ready to be read via the JTAG port. The state of these flags is captured into a status register (JDC_MSR), and also provided to a JTAG register (JOUT) for tool visibility.

There is a single bus interface to port register data out to the JTAGC, and a single bus interface to port data in from the JTAGC. The architecture is shown in *Figure 1146*.

Figure 1146. JDC block diagram



57.3 Register definition

Table 1109 shows the JDC module registers. Four 32-bit registers are implemented. Only 32-bit accesses are valid. The effects of access that are not 32 bits are not defined.

Table 1109. JDC memory map

Address offset	Register	Access	Reset value	Location
0x0	Module Configuration Register (JDC_MCR)	R/W	0x0000_0000	*Section 57.3.1.1*
0x4	Module Status Register (JDC_MSR)	W1C	0x0000_0000	*Section 57.3.1.2*
0x8	JTAG Output IPS Data Register (JDC_JOUT_IPS)	R/W	0x0000_0000	*Section 57.3.1.3*
0xC	JTAG Input IPS Data Register (JDC_JIN_IPS)	Read Only	0x0000_0000	*Section 57.3.1.4*

57.3.1 Register descriptions

57.3.1.1 Module configuration register (JDC_MCR)

Figure 1147 shows the JDC_MCR. The JDC_MCR is used to enable the interrupt outputs of the JDC. This register is reset by system destructive reset.

Offset: 0x00 Access: User read/write

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

Figure 1147. Module Configuration Register (JDC_MCR)

The JDC_MCR is described in *Table 1110*.

Table 1110. JDC_MCR register field descriptions

Field	Description
15 JIN_IEN	JIN Interrupt Enable. 1 Setting of JDC_MSR[JIN_INT] bit asserts the JIN interrupt 0 Setting of JDC_MSR[JIN_INT] bit does not assert the JIN interrupt
31 JOUT_IEN	JOUT Interrupt Enable. 1 Setting of JDC_MSR[JOUT_INT] bit asserts the JOUT interrupt 0 Setting of JDC_MSR[JOUT_INT] bit does not assert the JOUT interrupt

57.3.1.2 Module status register (JDC_MSR)

Figure 1148 shows the format of the JDC_MSR. The JDC_MSR holds the JTAG register status and interrupt bits. This register is reset by system destructive reset.

Offset: 0x04 Access: User w1c

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	JIN_RDY	0	JIN_INT	0	0	0	0	0	0	0	0	0	0	0	0	0	JOUT_RDY	0	JOUT_INT
W																w1c																w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1148. Module Status Register (JDC_MSR)

The JDC_MSR is described in *Table 1111*.

Table 1111. JDC_MSR register field descriptions

Field	Description
13 JIN_RDY	JIN Ready (read only). 1 Set when new data is written to the JDC_JIN_IPS register 0 Cleared upon software read of JDC_JIN_IPS contents via IPS
15 JIN_INT	JIN Interrupt. 1 Set when new data is written to the JDC_JIN_IPS register 0 Cleared by writing logic 1
29 JOUT_RDY	JOUT Ready (read only). 1 Set when new data is written to the JDC_JOUT_IPS register 0 Cleared upon tool read of JOUT register via JTAG port
31 JOUT_INT	JOUT Interrupt. 1 Set when JOUT_RDY bit is cleared by tool reading JOUT register 0 Cleared by writing logic 1

57.3.1.3 JTAG output IPS data register (JDC_JOUT_IPS)

The JDC_JOUT_IPS register holds data written via IPS. The JDC_JOUT_IPS contents are ported out and captured into the JOUT register to be read via the JTAG port. *Figure 1149* shows the format of the JDC_JOUT_IPS register. This register is reset by system destructive reset.

Offset: 0x08 Access: User read/write

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

Figure 1149. JDC_JOUT_IPS register

The JDC_JOUT_IPS register is described in *Table 1112*.

Table 1112. JDC_JOUT_IPS register field descriptions

Field	Description
0:31 DATA	Contains JOUT_IPS data.

57.3.1.4 JTAG input IPS data register (JDC_JIN_IPS)

Data written to the JTAG input data register (JIN) via the JTAG port is held in the JDC_JIN_IPS register, where it can be read via IPS. *Figure 1150* shows the format of the JDC_JIN_IPS register. This register is reset by system destructive reset.

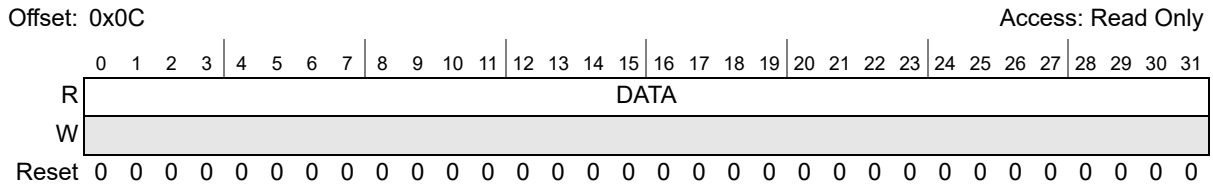


Figure 1150. JDC_JIN_IPS register

The JDC_JIN_IPS register is described in *Table 1113*.

Table 1113. JDC_JIN_IPS register field descriptions

Field	Description
0:31 DATA	Contains JIN_IPS data.

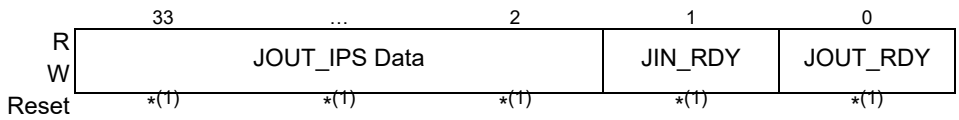
57.3.2 Non-Memory mapped register definition

The JDC also implements two JTAG accessible registers that are not memory mapped. One JTAG register is used to shift in data to be placed in the JDC_JIN_IPS register. The other JTAG register captures data from the JDC_JOUT_IPS register and ready status from the JDC_MSR to be shifted out via the JTAG port.

57.3.2.1 JTAG output data register (JOUT)

The JOUT register captures data from the JDC_JOUT_IPS register upon execution of the JOUT_READ JTAG instruction. It also holds the JDC_JIN_RDY and JDC_JOUT_RDY status bits. *Figure 1151* shows the format of the JOUT register. This register is reset by system destructive reset and JTAG reset.

Figure 1151. JOUT JTAG register



1. The reset value of the JOUT register is 34'b0.

The JOUT register is described in *Table 1114*.

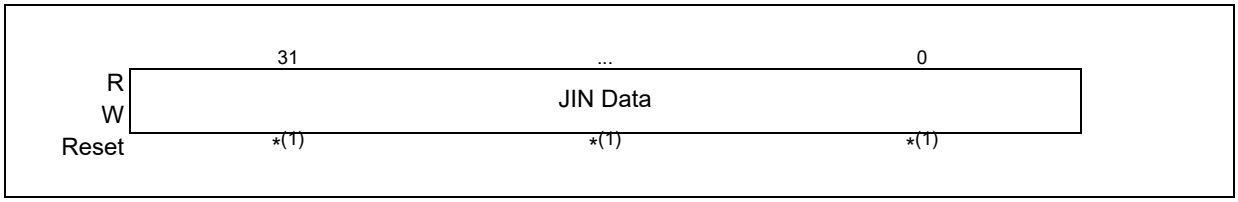
Table 1114. JOUT JTAG register field descriptions

Field	Description
JOUT_IPS Data	Data value from JOUT_IPS register
JIN_RDY	State of JIN_RDY bit from JDC_MSR
JOUT_RDY	State of JOUT_RDY bit from JDC_MSR

57.3.2.2 JTAG input data register (JIN)

Data is written to the JIN register via JTAG during execution of the JIN_WRITE JTAG instruction. The JIN data is later captured in the JDC_JIN_IPS register to be read via IPS. *Figure 1152* shows the format of the JIN register. This register is reset by system destructive reset and JTAG reset.

Figure 1152. JIN JTAG Register



1. The reset value of the JIN register is 32'b0.

The JIN register is described in *Table 1115*.

Table 1115. JIN JTAG register field descriptions

Field	Description
JIN Data	Contains data to be captured in JDC_JIN_IPS register upon exit of Update-DR state when executing WRITE_JIN JTAG instruction.

57.4 Functional description

The JDC module provides the ability to shift in data via the JTAG port and capture that data into a memory mapped register that can be accessed via IPS. It also provides the ability to capture data written to a memory mapped register into a JTAG shift register for output via the JTAG port. An overview of the module functionality is described below.

- A software write to the JOUT_IPS register sets the JOUT_RDY flag bit, indicating new data is available to be read from the JOUT register via the JTAG port. The state of the JOUT_RDY bit is reflected in the MSR and also ported out to the JOUT register. A JTAG read of the JOUT register via execution of the JOUT_READ instruction with a JOUT_RDY bit whose value is logic 1 indicates the register contains new data. The JOUT_RDY flag bit is cleared upon exit of the Capture-DR JTAG state during execution of the JOUT_READ instruction. Clearing the JOUT_RDY bit indicates to software that a new data value can be written to the JOUT_IPS register.
- A JTAG write to the JIN register via execution of the JIN_READ JTAG instruction updates the contents of the JDC_JIN_IPS register upon exit of the Update-DR state. An update of the JDC_JIN_IPS register sets the JIN_RDY flag bit, indicating new data

is available to be read via IPS. The state of the JIN_RDY bit is reflected in the MSR register and also ported out to the JOUT register. The JIN_RDY flag bit is cleared upon software read of the JDC_JIN_IPS register. A JTAG read of the JOUT register with a JIN_RDY value of logic 0 indicates new data can be written to the JIN register.

57.4.1 IPS register access

IPS access is available to the registers described in *Section 57.3.1*.

57.4.2 JTAG register access

Refer to the JTAGC documentation for information on how to access the JTAG registers.

57.4.2.1 JDC block instructions

The JDC block implements the IEEE 1149.1-2001 defined instructions listed in *Table 1116*. This section gives an overview of each instruction; refer to the IEEE 1149.1-2001 standard for more details. All undefined opcodes are reserved.

Table 1116. JTAG instructions

Instruction	Code[4:0]	Instruction Summary
Reserved	00001	Factory debug reserved
JOUT_READ	00010	Selects JOUT data register. Data from JOUT_IPS is captured into JOUT data register upon entry to Capture-DR state while JOUT_READ is active.
JIN_WRITE	01110	Selects JIN data register. Data from JIN is captured into JIN_IPS upon exit of Update-DR state while JIN_WRITE is active.
BYPASS	11111	Selects bypass register for data operations
Reserved	All other opcodes	Decoded to select bypass register

58 JTAG Master (JTAGM)

58.1 Introduction

The JTAG Master (JTAGM) is a module that is able to act as JTAG master inside the device. The module has a parallel interface that can exchange data with another serial communication module or via customer software.

The data transferred to this module is transformed to produce TCK, TMS, TDI and TRST outputs and to accept TDO inputs. The JTAGM is connected in the device to allow these five signals to connect to the JTAGC as if the JTAG data is coming from an outside tool. The JTAGM generates all required JTAG scan chains to allow software and high speed serial communication access to all JTAG mapped resources.

58.1.1 Overview

The JTAGM is the master to drive JTAG signals from within the device. It has options to receive parallel data from software through the IPS interface. It then sends this data on TDI, TMS and TCK to the DCI. The JTAGM also receives serial data through TDO from the DCI and transfers this data through another parallel interface. The JTAGM has the following registers:

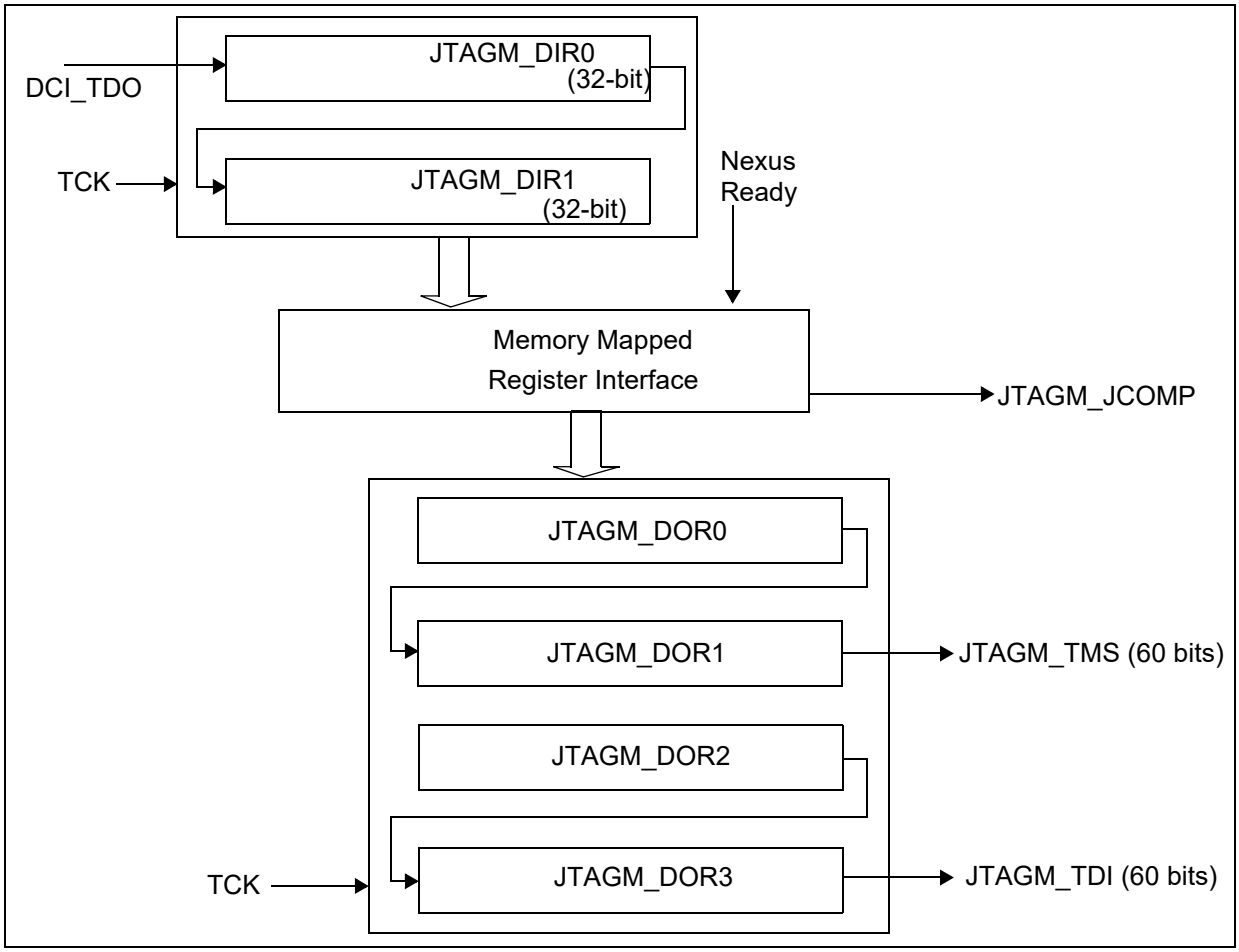
- Configuration register
- Status register
- Four data output registers
- Two data input registers

All these registers, described in detail in *Section 58.4*, are memory mapped and can be accessed by software.

The clocks for the JTAGM are derived from the system clock. The JTAGM also samples the ready signal from the Nexus, to provide a more efficient handshake to the external tool to read and write any memory mapped address location within the device.

Figure 1153 shows the block diagram of the JTAGM.

Figure 1153. JTAGM block diagram



58.1.2 Feature description

The main features of the JTAGM are the following:

- Provides efficient handshake to the external tool to read any memory mapped address location within the device
- Provides software the option to write data for driving JTAG

58.2 Functional description

58.2.1 Module architecture

The JTAGM is simple in concept. TCK is generated from the device system clock. Data is pushed to the modules via a 32-bit wide parallel interface. The data is in the form of 60 bits of TMS data and 60 bits of TDI data. This data is the logical state to be driven onto the TMS and TDI output pins on each of the 60 TCK cycles. During these 60 TCK cycles, the TDO pin is sampled and the 60 bits of sampled data is transferred to the 32-bit wide module interface. This concept allows complete flexible generation on TMS and TDI data and capture of TDO data. The only limitation is that JTAG signals can only be generated in 60 TCK cycles packets. Extra cycles are wasted in idle cycles at the end of a scan chain.

58.2.2 JTAGM Ready signal

The Nexus Read/Write Access (RWA) module allows an external tool to read and write any memory mapped address location within the device via JTAG commands. Read accesses to memory take a finite amount of time and a JTAG read of the data register too soon results in erroneous data being read back. There is a bit in a JTAG register to show the read data is available.

58.2.2.1 Status register ready bit

A bit is provided in the JTAGM status register to indicate the status of the ready signal. To provide this ready signal functionality, the ready signal is monitored. If the ready signal has toggled between the start of the previous 60 TCK shift period and start of the current 60 TCK shift period, the ready bit in the JTAGM status register is set. Otherwise the ready bit in the status register is cleared.

The entire content of the 32-bit status register is appended to the 60 bits of JTAG TDO data captured along with 4 bits of 0-padding and sent as output on the parallel output port as a 96-bit payload on channel A. This provides back to the tool, an indication that the data read in the current transaction is valid.

58.2.2.2 Inter-JTAG frame gap timer

To operate in conjunction with the status register ready bit, a configurable timer is provided to force a programmable gap between the end of one 60-bit JTAG frame and the start of the next. This timer is configured via the JTAGM configuration register from 1 to 64 TCKs in 1 TCK intervals. Furthermore, if the ready signal asserts during this inter-frame gap period, the gap timer is aborted and the next JTAG frame starts immediately.

58.2.3 $\overline{\text{EVTO}}$ and $\overline{\text{EVTI}}$ signals

The standard Nexus EVTI (Event In) and EVTO (Event Out) signals are optionally present on dedicated package pins and are used by an external tool in conjunction with the JTAG interface to pass information, events and triggers between the tool and the MCU.

The JTAGM as well as providing a higher speed alternative to the standard JTAG interface, also allows control and status of the EVTI and EVTO signals to be embedded into the serial protocol. The EVTI0 and EVTI1 signals can be asserted by writing control bits in the Module Configuration Register (JTAGM_MCR). The JTAGM can be configured by bits in the Module Configuration Register (JTAGM_MCR) to respond to a rising/falling/both edge transition of either EVTO0 or EVTO1. The response is a CPU interrupt.

The CPU interrupt is gated by an Interrupt enable bit, in MCR and reported by Interrupt Status flags in the Status Register (JTAGM_SR). The status bits stay set until cleared by writing to the clear bits in the JTAG_SR.

Asserting the evti0_assert bit in the JTAGM_MCR register can trigger debug mode in the DCI. Asserting the evti1_assert bit in the JTAGM_MCR register can trigger debug mode in the DCI.

Note: EVTI generation and EVTO detection bits of JTAGM can be used by debugger to detect EVTI assertion and to generate EVTO. This method is useful when EVTI/O pins are not available on SoC PADs.

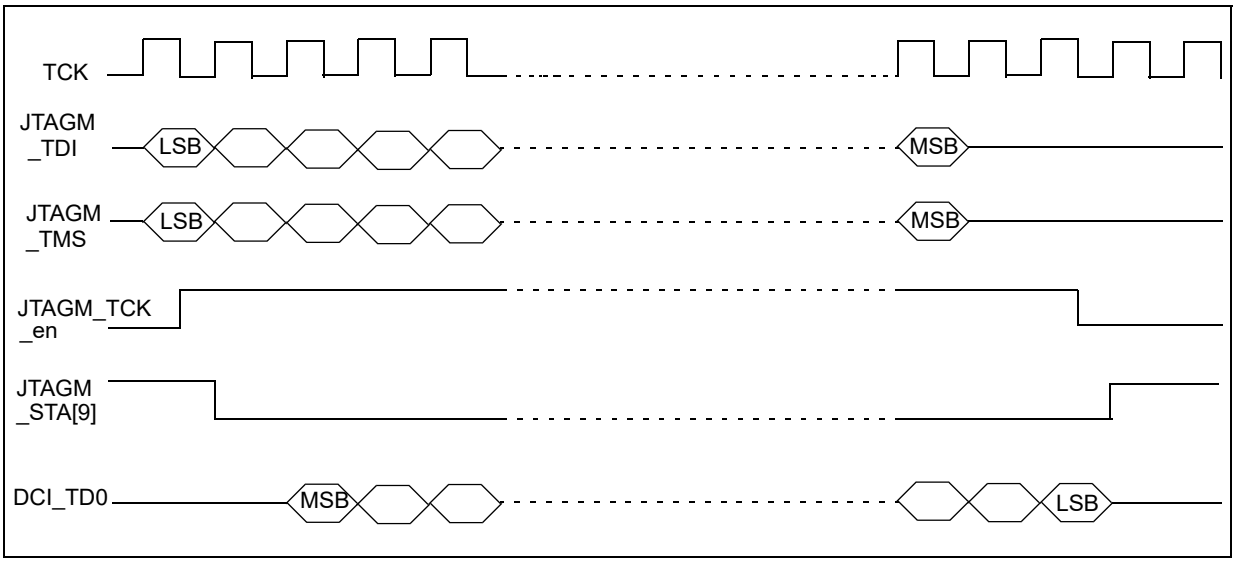
58.2.4 JTAGM configuration and status monitoring

The JTAGM module requires some level of configuration and status monitoring. The JTAGM includes a 32-bit configuration register (JTAGM_MCR) and a 32-bit status register (JTAGM_SR, which includes the ready bit mentioned in *Section 58.2.2.1*). The status register also includes status bits provided by the DCI module.

58.2.5 JTAGM to DCI serial interface

Figure 1154 shows the data transfer timings between JTAGM and DCI.

Figure 1154. JTAGM to DCI serial interface diagram



58.2.6 Software interface

It is possible for software to write the 120 bits of JTAG data to be output and to read the 60 bits of data. The JTAGM module has an IPS interface.

58.3 Modes of operation

There are no special modes of operation. The JTAGM works normally in the debug mode.

58.4 Memory map and register definition

Table 1117 contains the memory map of the JTAGM registers.

Table 1117. Memory map

Offset	Register	Location
0x00	Module Configuration Register (JTAGM_MCR)	*Section 58.4.1.1*
0x04	Status Register (JTAGM_SR)	*Section 58.4.1.2*

Table 1117. Memory map (continued)

Offset	Register	Location
0x08	Data Out Register 0 (JTAGM_DOR0)	*Section 58.4.1.3*
0x0C	Data Out Register 1 (JTAGM_DOR1)	*Section 58.4.1.4*
0x10	Data Out Register 2 (JTAGM_DOR2)	*Section 58.4.1.5*
0x14	Data Out Register 3 (JTAGM_DOR3)	*Section 58.4.1.6*
0x18	Reserved	—
0x1C	Data Input Register 0 (JTAGM_DIR0)	*Section 58.4.1.7*
0x20	Data Input Register 1 (JTAGM_DIR1)	*Section 58.4.1.8*

58.4.1 Register descriptions

58.4.1.1 Module configuration register (JTAGM_MCR)

This register contains the status bits for the current transfer. *Figure 1155* shows the format of the JTAGM_MCR.

Offset 0x00 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	SWRESET	evto0_sense		evto1_sense		evto_IE	evti0_assert	evti1_assert	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	inter_jtag_frame_timer						0	SIE	IIE	TCKSEL			jtagm_JCOMP	DTM
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0

Figure 1155. JTAGM_MCR register

The JTAGM_MCR register is described in *Table 1118*.

Table 1118. JTAGM_MCR register field descriptions

Field	Description
0 SWRESET	Software Reset. Writing a 1 resets the state machine and counters inside JTAGM. It is a self clearing bit. Writing a 0 has no effect. **Note:** After being written to “1” , this bit is auto cleared within 3 TCK cycles.
1:2 evto0_sense	Determines JTAGM response to activity on $\overline{\text{EVTO}}$0 00 No action is taken by the JTAGM in response to any $\overline{\text{EVTO}}$0 edges (default setting) 01 evto0_edge status bit is set whenever a falling edge is detected on $\overline{\text{EVTO}}$0 10 evto0_edge status bit is set whenever a rising edge is detected on $\overline{\text{EVTO}}$0 11 evto0_edge status bit is set whenever a falling or rising edge is detected on $\overline{\text{EVTO}}$0

Table 1118. JTAGM_MCR register field descriptions (continued)

Field	Description
3:4 evto1_sense	Determines JTAGM response to activity on $\overline{\text{EVTO}}$1 line 00 No action is taken by the JTAGM in response to any $\overline{\text{EVTO}}$1 edges (default setting) 01 evto1_edge status bit is set whenever a falling edge is detected on $\overline{\text{EVTO}}$1 10 evto1_edge status bit is set whenever a rising edge is detected on $\overline{\text{EVTO}}$1 11 evto1_edge status bit is set whenever a falling or rising edge is detected on $\overline{\text{EVTO}}$1
5 evto_IE	Enables $\overline{\text{EVTO}}$ triggered JTAGM interrupt 0 No JTAGM interrupt is generated In response to any $\overline{\text{EVTO}}$0 or $\overline{\text{EVTO}}$1 activity 1 JTAGM interrupt is generated when evto0_edge or evto1_edge are set
6 evti0_assert	EVTI0 Assert 0 Set jtagm_evti0 low/asserted 1 Set jtagm_evti0 high/de-asserted
7 evti1_assert	EVTI1 Assert 0 Set jtagm_evti1 low/asserted 1 Set jtagm_evti1 high/de-asserted **Note:** In the SW mode(when DTM =1) , regardless of the state of “evti1_assert” bit the internall “evti” signal remains de-asserted . Though this bit can be written, writing to this bit has no effect .
18:23 inter_jtag_frame_timer	TCK delay 000000 0 TCK cycle 000001 1 TCK cycle … 111111 63 TCK cycles
25 SIE	SPU Interrupt Enable 0 JTAGM does not generate an interrupt to the CPU 1 JTAGM generates an interrupt to the CPU if the SPU interrupt request is asserted
26 IIE	Idle Interrupt Enable 0 JTAGM does not generate an interrupt to the CPU upon completion of a 60-bit JTAG transfer 1 JTAGM generates an interrupt to the CPU upon completion of a 60-bit JTAG transfer
27:29 TCKSEL	TCK Divider division factor control value. (TCK Divider divides the clock input to JTAGM IP to generate TCK out(input to DCI)). 000 TCK Divider division factor is 1 001 TCK Divider division factor is 2 010 TCK Divider division factor is 3 011 TCK Divider division factor is 4 100 TCK Divider division factor is 5 101 TCK Divider division factor is 6 110 TCK Divider division factor is 7 111 TCK Divider division factor is 8 **Note:** It should be taken care that the maximum frequency of TCK input to DCI, should be equal or less than half of system clock.
30 jtagm_JCOMP	JTAG reset 0 JCOMP low/asserted 1 JCOMP high/not asserted (default)
31 DTM	Data Transer Mode 1 Data for JTAG transferred by software. 0 Data transferred by software disabled.

58.4.1.2 Status register (JTAGM_SR)

Figure 1156 shows the format of the JTAGM_SR .

Offset 0x04 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0						dci_status							
W																
Reset	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	evto0_edge	evto1_edge	SPU_INT_CLR	SPU_INT	0	Nexus_err	Idle	NR			0	0	0	0	0	0
W									evto0_clr	evto1_clr						
Reset	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0

Figure 1156. JTAGM_SR register

The JTAGM_SR register is described in *Table 1119*.

Table 1119. JTAGM_SR register field descriptions

Field	Description
8:15 dci_status	Status information from the DCI. The composition of these bits is as follows: bit 8 Reserved to 0 bit 9 Reserved bit 10 DCI JTAG port enable bit 11 Reserved bit 12 DCI JTAG safe mode bit 13 Reserved bit 14 Reserved bit 15 DCI tool present , set when a debug tool is connected
16 evto0_edge	In software mode (DTM = 1), the bit stays set until cleared by writing to evto0_clr bit 0 No activity sensed on $\overline{\text{EVTO0}}$ 1 Activity defined by evto0_sense detected on $\overline{\text{EVTO0}}$
17 evto1_edge	In software mode (DTM = 1), the bit stays set until cleared by writing to evto1_clr bit 0 No activity sensed on $\overline{\text{EVTO1}}$ 1 Activity defined by evto1_sense detected on $\overline{\text{EVTO1}}$
18 SPU_INT_CLR	Write 1 clears the SPU_INT This is a self clearing bit. If SPU_INT is not set then SPU_INT_CLR remains set until SPU_INT is set, and both are cleared.
19 SPU_INT	This bit is set when the SPU interrupt request input rising edge is detected. Writing a 1 to SPU_INT_CLR resets this bit.
21 Nexus_err	Nexus error bit. It is set when there is an error, else it is 0.

Table 1119. JTAGM_SR register field descriptions (continued)

Field	Description
22 Idle	Idle bit. Idle bit is cleared while a JTAG frame is in progress. This bit is set to '1' when a JTAG frame ends. Writing a 1 to the Idle bit clears the Idle_interrupt in SW Mode. A read always gives the JTAG frame status but does not give the write value.
23 NR	Status of Ready bit from Nexus RWA. This bit is automatically cleared when a new JTAG message is started. 0 RDY has not asserted indicating Nexus RWA is not ready **Note:** 1 RDY has asserted indicating a Nexus RWA has completed bus access and is ready for the data register to be read
24 evto0_clr	Used for clearing the evto0_edge bit.
25 evto1_clr	Used for clearing the evto1_edge bit.

58.4.1.3 Data out register 0 (JTAGM_DOR0)

Figure 1157 shows the format of the JTAGM_DOR0.

Offset 0x08 Access: User read/write

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

Figure 1157. JTAGM_DOR0 register

The JTAGM_DOR0 register is described in *Table 1120*.

Table 1120. JTAGM_DOR0 register field descriptions

Field	Description
0:31 TMS_HIGH	Higher word of data for TMS, bits 59-28. Writable by software only when DTM = 1

58.4.1.4 Data out register 1 (JTAGM_DOR1)

Figure 1158 shows the format of the JTAGM_DOR1.

Offset 0x0C Access: User read/write

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

Figure 1158. JTAGM_DOR1 register

The JTAGM_DOR1 register is described in *Table 1121*.

Table 1121. JTAGM_DOR1 register field descriptions

Field	Description
0:27 TMS_LOW	Lower word of data for TMS, bits 27–0. Writable by software only when DTM = 1. TMS_LOW[0] is shifted out first

58.4.1.5 Data out register 2 (JTAGM_DOR2)

Figure 1159 shows the format of the JTAGM_DOR2.

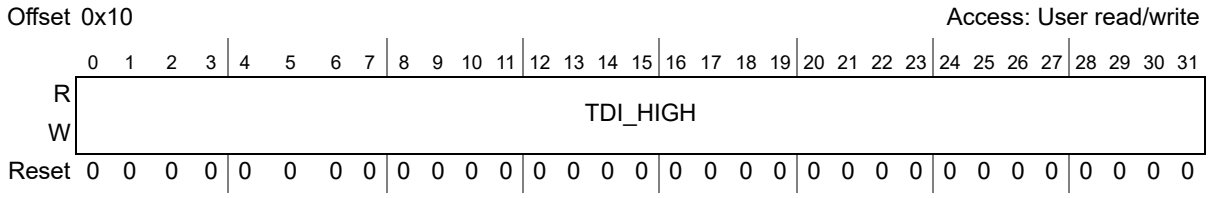


Figure 1159. JTAGM_DOR2 register

The JTAGM_DOR2 register is described in *Table 1122*.

Table 1122. JTAGM_DOR2 register field descriptions

Field	Description
0:31 TDI_HIGH	Higher word of data for TDI, bits 59–28. Writable by software only when DTM = 1

58.4.1.6 Data out register 3 (JTAGM_DOR3)

Figure 1160 shows the format of the JTAGM_DOR3.

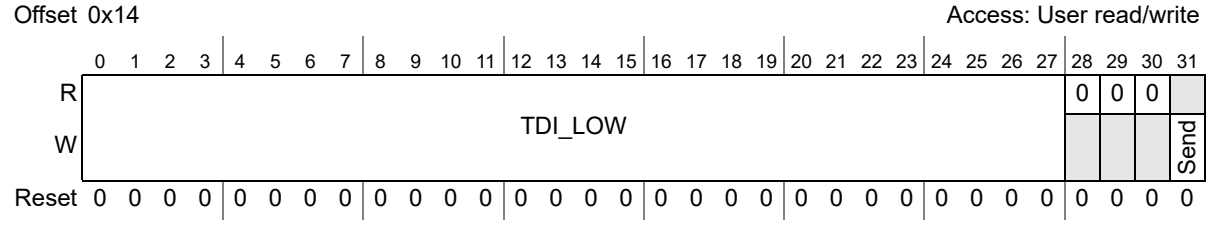


Figure 1160. JTAGM_DOR3 register

The JTAGM_DOR3 register is described in *Table 1123*.

Table 1123. JTAGM_DOR3 register field descriptions

Field	Description
0:27 TDI_LOW	Lower word of data for TDI, bits 27–0. Writable by software only when DTM = 1. TDI_LOW[0] is shifted out first
31 Send	Send bit. When this bit is set, data is sent to the DCI. It is a self-clearing bit and is always read as 0. Writable by software only when DTM = 1

58.4.1.7 Data input register 0 (JTAGM_DIR0)

Figure 1161 shows the format of the JTAGM_DIR0.

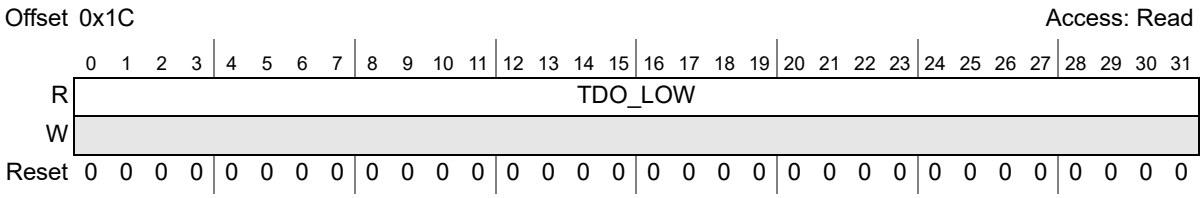


Figure 1161. JTAGM_DIR0 register

The JTAGM_DIR0 register is described in *Table 1124*.

Table 1124. JTAGM_DIR0 register field descriptions

Field	Description
0:31 TDO_LOW	Lower word of data received on TDO, bits 0–31

58.4.1.8 Data input register 1 (JTAGM_DIR1)

Figure 1162 shows the format of the JTAGM_DIR1.

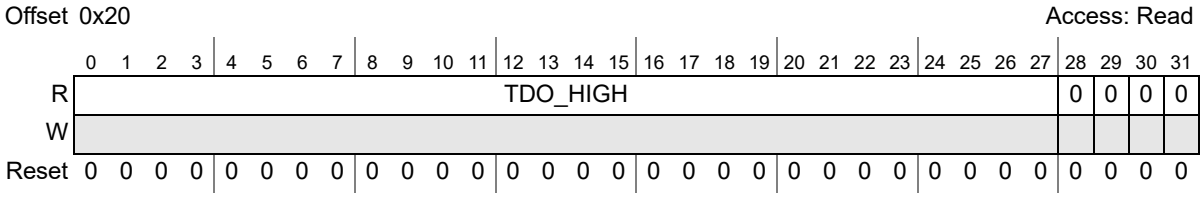


Figure 1162. JTAGM_DIR1 register

The JTAGM_DIR1 register is described in *Table 1125*.

Table 1125. JTAGM_DIR1 register field descriptions

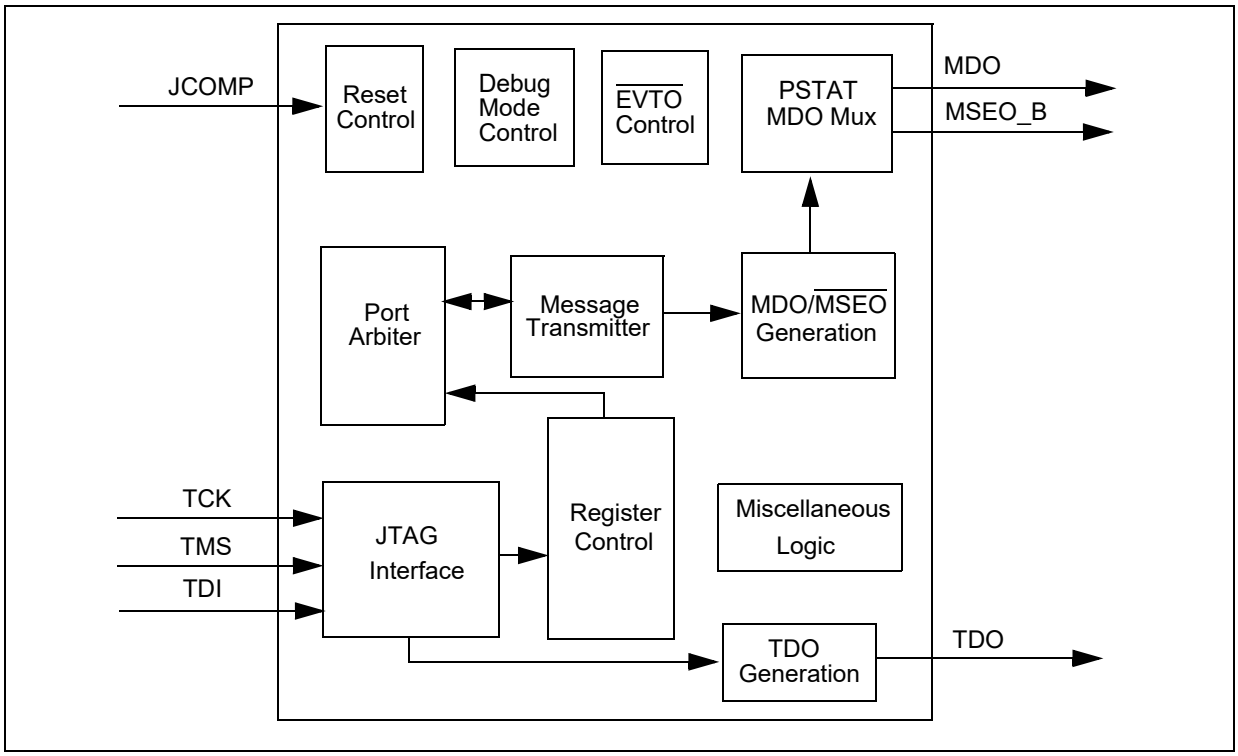
Field	Description
0:27 TDO_HIGH	Higher word of data received on TDO, bits 32–59

59 Nexus Port Controller (NPC)

59.1 Introduction

Figure 1163 is a block diagram of the Nexus Port Controller (NPC) block.

Figure 1163. Nexus Port Controller Block Diagram



59.1.1 Overview

On a system-on-a-chip device, there are often multiple blocks that require development support. Each of these blocks implements a development interface based on the IEEE-ISTO 5001-2001 standard. The blocks share input and output ports that interface with the development tool. The NPC controls the usage of the input and output port in a manner that allows all the individual development interface blocks to share the port, and appear to the development tool to be a single block.

59.1.2 Features

The NPC block performs the following functions:

- Controls arbitration for ownership of the Nexus Auxiliary Output Port.
- Nexus Device Identification Register and Messaging.
- Generates MCKO enable and frequency division control signals.
- Controls sharing of $\overline{\text{EVTO}}$.
- Generates an MCKO clock gating control signal to enable gating of MCKO when the auxiliary output port is idle.
- Control of the device-wide debug mode.
- $\overline{\text{RDY}}$ pin support to increase the transfer rate of the IEEE 1149.1 port for large data read requests.

59.1.3 Modes of Operation

The NPC block uses the JCOMP input, and an internal power-on reset indication as its primary reset signals. Upon exit of reset, the mode of operation is determined by the Port Configuration Register (PCR) settings.

59.1.3.1 Reset

The NPC block is asynchronously placed in reset when either power-on reset is asserted, JCOMP is not set for Nexus access (ipp_ind_jcomp is set to a value other than npc_jcomp_plug) or the TAP controller state machine is in the Test-Logic-Reset state. Holding TMS high for 5 consecutive rising edges of TCK guarantees entry into the Test-Logic-Reset state regardless of the current TAP controller state. Following negation of power-on reset, the NPC remains in reset until the system clock achieves lock. The NPC is unaffected by other sources of reset. While in reset, the following actions occur:

- The TAP controller is forced into the Test-Logic-Reset state.
- The auxiliary output port pins are negated.
- The TDI, TMS, and TCK TAP inputs are ignored (when in power-on reset or JCOMP not set for NPC operation only).
- Registers default back to their reset values.

59.1.3.2 Disabled-Port Mode

In disabled-port mode, auxiliary output pin port enable signals are negated, thereby disabling message transmission. Any debug feature that generates messages can not be used. The primary features available are class 1 features and read/write access to the registers. Class 1 features include the ability to trigger a breakpoint event indication through $\overline{\text{EVTO}}$.

59.1.3.3 Full-Port Mode

Full-port mode (FPM) is entered by asserting the MCKO_EN and FPM bits in the PCR. All trace features are enabled or can be enabled by writing the configuration registers via the TAP. The number of MDO pins available is device-specific.

59.1.3.4 Reduced-Port Mode

Reduced-port mode (RPM) is entered by asserting the MCKO_EN bit and negating the FPM bit in the PCR.All trace features are enabled or can be enabled by writing the configuration registers via the TAP. The number of MDO pins available is device-specific. All MSEO signals need to be connected.

59.2 External signal description

59.2.1 Overview

The NPC pin interface provides for the transmission of messages from Nexus blocks to the external development tools and for access to Nexus client registers. The NPC pin definition is outlined in *Table 1126*.

Table 1126. NPC Signal Properties

Name	Port	Function	Reset State	Pull(1)
EVTO_B	Auxiliary	Event Out pin	0b1	—
JCOMP	JTAG	JTAG Compliancy and TAP Sharing Control	—	Down
MDO	Auxiliary	Message Data Out pins	0	—
MSEO	Auxiliary	Message Start/End Out pins	0b11	—
TCK	JTAG	Test Clock Input	—	Down
TDI	JTAG	Test Data Input	—	Up
TDO	JTAG	Test Data Output	High Z(2)	—
TMS	JTAG	Test Mode Select Input	—	Up

1. The pull is not implemented in this block. Pullup/pulldown devices are implemented in the pads.
2. TDO output buffer enable is negated when the NPC is not in the Shift-IR or Shift-DR states. A weak pull may be implemented on TDO at the SoC level.

59.2.2 Detailed Signal Descriptions

This section describes each of the signals listed in *Table 1126* in more detail. Note that the JTAG test clock (TCK) input from the pin is not a direct input to the NPC. The NPC requires two separate input clocks for TCK clocked logic, one for posedge (rising edge TCK) logic and one for negedge (falling edge TCK) logic. Both clocks are derived from the pin TCK, and generated external to the NPC.

59.2.2.1 EVTO_B — Event Out

Event Out ($\overline{\text{EVTO}}$) is an output pin that is asserted upon breakpoint occurrence to provide breakpoint status indication. The $\overline{\text{EVTO}}$ output of the NPC is generated based on the values of the individual $\overline{\text{EVTO}}$ signals from all Nexus blocks that implement the signal.

59.2.2.2 JCOMP — JTAG Compliancy

The JCOMP signal provides the ability to share the TAP. The NPC TAP controller remains in reset until it is enabled using the JCOMP signal.

59.2.2.3 MDO — Message Data Out

Message Data Out (MDO) are output pins used for uploading OTM, BTM, DTM, and other messages to the development tool. The development tool should sample MDO on the rising edge of MCKO. The width of the MDO bus used is determined by reset configuration.

59.2.2.4 MSEO_B — Message Start/End Out

Message Start/End Out ($\overline{\text{MSEO}}$) is an output pin that indicates when a message on the MDO pins has started, when a variable length packet has ended, or when the message has ended. The development tool should sample $\overline{\text{MSEO}}$ on the rising edge of MCKO.

59.2.2.5 TCK — Test Clock Input

Test Clock Input (TCK) pin is used to synchronize the test logic and control register access through the JTAG port.

59.2.2.6 TDI — Test Data Input

Test Data Input (TDI) pin receives serial test instruction and data. TDI is sampled on the rising edge of TCK.

59.2.2.7 TDO — Nexus Test Data Output

Test Data Output (TDO) pin transmits serial output for instructions and data. TDO is three-stateable and is actively driven in the SHIFT-IR and SHIFT-DR controller states. TDO is updated on the falling edge of TCK and sampled by the development tool on the rising edge of TCK.

59.2.2.8 TMS — Test Mode Select

Test Mode Select Input (TMS) pin is used to sequence the IEEE 1149.1-2001 TAP controller state machine. TMS is sampled on the rising edge of TCK.

59.3 Register definition

This section provides a detailed description of the NPC registers accessible to the end user. Individual bit-level descriptions and reset states of the registers are included.

Table 1127 shows the NPC registers by index values. The registers are not memory-mapped and can only be accessed via the TAP. The NPC block does not implement the client select control register because the value does not matter when accessing the registers.

Note that the bypass and instruction registers have no index values. These registers are not accessed in the same manner as Nexus client registers. Refer to the individual register descriptions for more detail.

Table 1127. NPC registers

Index	Register
0	Device ID Register (DID)
127	Port Configuration Register (PCR)

59.3.1 Register descriptions

This section consists of NPC register descriptions.

59.3.1.1 Bypass Register

The bypass register is a single-bit shift register path selected for serial data transfer between TDI and TDO when the BYPASS instruction or any unimplemented instructions are active. After entry into the Capture-DR state, the single-bit shift register is set to a logic 0. Therefore, the first bit shifted out after selecting the bypass register is always a logic 0.

59.3.1.2 Instruction Register

The NPC block uses a 4-bit instruction register as shown in *Figure 1164*. The instruction register is accessed via the SELECT_IR_SCAN path of the tap controller state machine, and allows instructions to be loaded into the block to enable the NPC for register access (NEXUS_ENABLE) or select the bypass register as the shift path from TDI to TDO (BYPASS or unimplemented instructions).

Instructions are shifted in through TDI while the TAP controller is in the Shift-IR state, and latched on the falling edge of TCK in the Update-IR state. The latched instruction value can only be changed in the Update-IR and Test-Logic-Reset TAP controller states. Synchronous entry into the Test-Logic-Reset state results in synchronous loading of the BYPASS instruction. Asynchronous entry into the Test-Logic-Reset state results in asynchronous loading of the BYPASS instruction. During the Capture-IR TAP controller state, the instruction register is loaded with the value of the previously executed instruction, making this value the register's read value when the TAP controller is sequenced into the Shift-IR state.

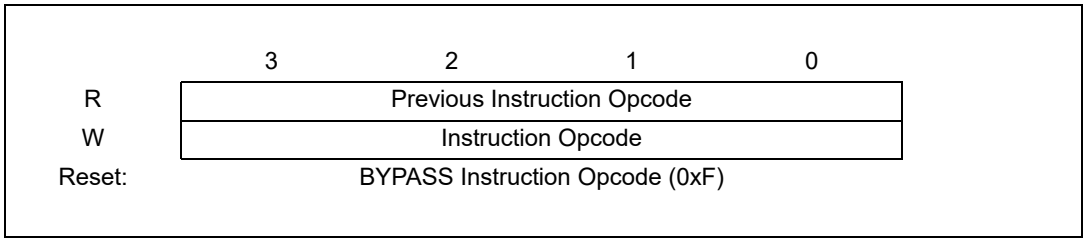


Figure 1164. 4-Bit Instruction Register

59.3.1.3 Nexus Device ID register (DID)

The device identification register, shown in *Figure 1165*, allows the part revision number, design center, part identification number, and manufacturer identity code of the part to be determined through the auxiliary output port.

Register index: 0

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Minor Mask				Major Mask				Design Center		Technology Node		Application Family			Sub Family
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	Sub Family			0	Manufacturer ID											0
W																
Reset	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0

Figure 1165. Nexus Device ID register(DID)

Table 1128. DID field descriptions

Bit	Description
0:3	Minor Mask These bits contain the minor mask of the part
4:7	Major Mask These bits contain the major mask of the part
8:9	Design Center These bits indicate the device design center
10:11	Technology Node These bits indicate the device technology node 00 55nm 01 40nm
12:14	Application Family These bits indicate the device application family 101 SPC57S - C&S 55nm 110 SPC58S - C&S 40nm
15:18	Sub Family These bits indicate the device sub family 0001 Velvety 0010 Sphaero 0011 Tarento 0100 Caledo
20:30	Manufacturer Identity Code These bits contain the reduced Joint Electron Device Engineering Council (JEDEC) ID 00000100000: STMicroelectronics

59.3.1.4 Port Configuration Register (PCR)

Register index: 127

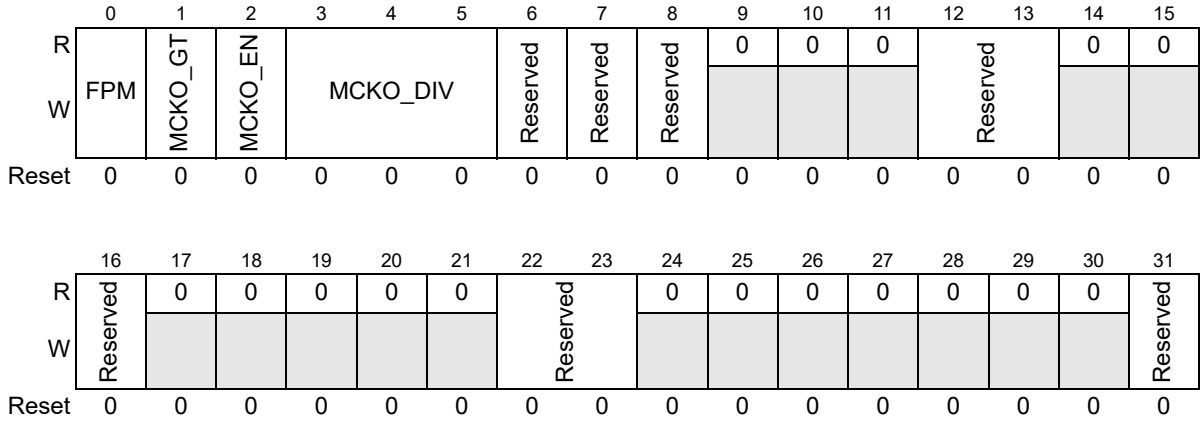


Figure 1166. Port Configuration Register (PCR)

The PCR, shown in *Figure 1166*, is used to select the NPC mode of operation, enable MCKO and select the MCKO frequency, and enable or disable MCKO gating. This register should be configured as soon as the NPC is enabled.

The PCR register may be rewritten by the debug tool subsequent to the enabling of the NPC for low power debug support. In this case, the debug tool may set and clear the LP_DBG and LPn_SYN bits, but must preserve the original state of the remaining bits in the register.

Note: The mode or clock division must not be modified after MCKO has been enabled. Changing the mode or clock division while MCKO is enabled can produce unpredictable results.

Table 1129. PCR field descriptions

Bit	Name	Description
0	FPM	Full Port Mode The value of the FPM bit determines if the auxiliary output port uses the full MDO port or a reduced MDO port to transmit messages. 1 All MDO pins are used to transmit messages 0 A subset of MDO pins are used to transmit messages
1	MCKO_GT	MCKO Clock Gating Control This bit is used to enable or disable MCKO clock gating. If clock gating is enabled, the MCKO clock is gated when the NPC is in enabled mode but not actively transmitting messages on the auxiliary output port. When clock gating is disabled, MCKO is allowed to run even if no auxiliary output port messages are being transmitted. 1 MCKO gating is enabled 0 MCKO gating is disabled

Table 1129. PCR field descriptions (continued)

Bit	Name	Description
2	MCKO_EN	MCKO Enable This bit enables the MCKO clock to run. When enabled, the frequency of MCKO is determined by the MCKO_DIV field. 1 MCKO clock is enabled 0 MCKO clock is driven to zero
3:5	MCKO_DIV	MCKO Division Factor The value of this signal determines the frequency of MCKO relative to the system clock frequency when MCKO_EN is asserted. *Table 1130* shows the meaning of MCKO_DIV Values. In this table, SYS_CLK represents the system clock frequency.

Table 1130. MCKO_DIV Values

MCKO_DIV[2:0]	MCKO Frequency
0	SYS_CLK(1)
1	SYS_CLK/2
2	Reserved
3	SYS_CLK/4
4	Reserved
5	Reserved
6	Reserved
7	SYS_CLK/8

1. The SYS_CLK setting for MCKO frequency should only be used if this setting does not violate the maximum operating frequency of the auxiliary port pins.

59.4 Functional description

59.4.1 NPC reset configuration

The NPC is placed in disabled mode upon exit of reset. If message transmission via the auxiliary port is desired, a write to the PCR is then required to enable the NPC and select the mode of operation. Asserting MCKO_EN places the NPC in enabled mode and enables MCKO. The frequency of MCKO is selected by writing the MCKO_DIV field. Asserting or negating the FPM bit selects full-port or reduced-port mode, respectively.

Table 1131 describes the NPC reset configuration options.

Table 1131. NPC reset configuration options

Power On Reset Status Asserted?	MCKO_EN bit of the Port Configuration Register	FPM bit of the Port Configuration Register	Configuration
yes	X	X	Reset
no	0	X	Disabled

Table 1131. NPC reset configuration options (continued)

Power On Reset Status Asserted?	MCKO_EN bit of the Port Configuration Register	FPM bit of the Port Configuration Register	Configuration
no	1	1	Full-Port Mode
no	1	0	Reduced-Port Mode

59.4.2 Auxiliary Output Port

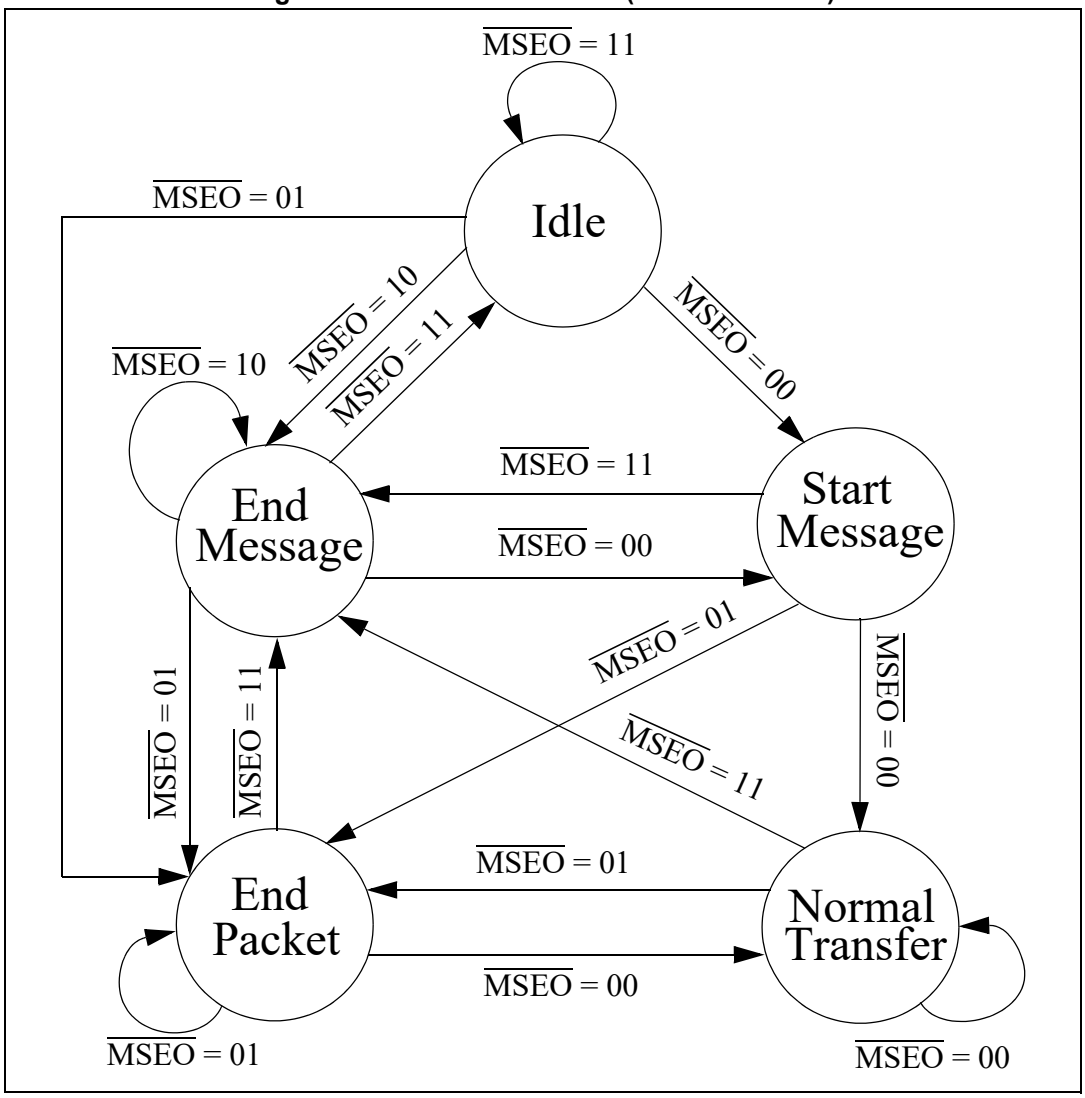
The auxiliary output port is shared by each of the Nexus modules on the device. The NPC communicates with each of the Nexus modules and arbitrates for access to the port.

59.4.2.1 Output Message Protocol

The protocol for transmitting messages via the auxiliary port is accomplished with the MSEO functions. The MSEO pins are used to signal the end of variable-length packets and the end of messages. They are not required to indicate the end of fixed-length packets. MDO and MSEO are sampled on the rising edge of MCKO.

Figure 1167 illustrates the state diagram for MSEO transfers. All transitions not included in the figure are reserved, and must not be used.

Figure 1167. $\overline{\text{MSEO}}$ Transfers (for 2-bit $\overline{\text{MSEO}}$)



59.4.2.2 Output Messages

In addition to sending out messages generated in other Nexus blocks, the NPC can also output the device ID message contained in the device ID register on the MDO pins. The device ID message can also be sent out serially through TDO.

Table 1132 describes the device ID output message that the NPC can transmit on the auxiliary port. The TCODE is the first packet transmitted.

Table 1132. NPC output messages

Message Name	Min. Packet Size (bits)	Max Packet Size (bits)	Packet Type	Packet Name	Packet Description
Device ID Message	6	6	fixed	TCODE	Value = 1
	32	32	fixed	ID	DID register contents

Table 1133 shows the various message formats that the pin interface formatter has to encounter. Note that for variable-length fields, the transmitted size of the field is determined from the range of the least significant bit to the most significant non-zero-valued bit (that is most significant zero-valued bits are not transmitted).

Table 1133. Message field sizes

Message	TCODE	Field #1	Field #2	Field #3	Field #4	Field #5	Min. Size(1) (bits)	Max. Size(2) (bits)
Device ID Message	1	Fixed = 32	NA	NA	NA	NA	38	38

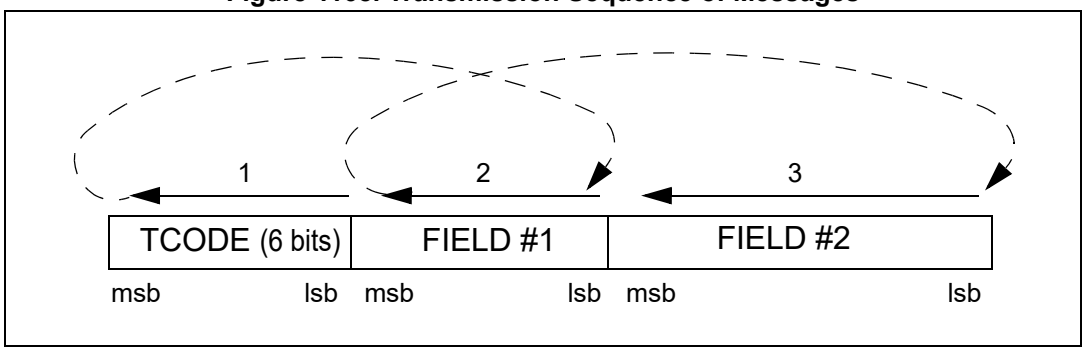
1. Minimum information size. The actual number of bits transmitted depends on the number of MDO pins
2. Maximum information size. The actual number of bits transmitted depends on the number of MDO pins

The double edges in *Table 1133* indicate the starts and ends of messages. Fields without shaded areas between them are grouped into super-fields and can be transmitted together without end-of-packet indications between them.

59.4.2.3 Rules of Message

- A variable-sized field within a message must end on a port boundary. (Port boundaries depend on the number of MDO pins active with the current reset configuration).
- A variable-sized field may start within a port boundary only when following a fixed-length field.
- Super-fields must end on a port boundary.
- When a variable-length field is sized such that it does not end on a port boundary, it is necessary to extend and zero fill the remaining bits after the highest order bit so that it can end on a port boundary.
- Multiple fixed-length packets may start and/or end on a single clock.
- When any packet follows a variable-length packet, it must start on a port boundary.
- The field containing the TCODE number is always transferred out first, followed by subsequent fields of information.
- Within a field, the lowest significant bits are shifted out first. *Figure 1168* shows the transmission sequence of a message that is made up of a TCODE followed by two fields.

Figure 1168. Transmission Sequence of Messages



59.4.3 IEEE 1149.1-2001 (JTAG) TAP

The NPC block uses the IEEE 1149.1-2001 TAP for accessing registers. Each of the individual Nexus blocks on the device implements a TAP controller for accessing its registers as well. TAP signals include TCK, TDI, TMS, and TDO. There may also be other blocks on the MCU that use the TAP and implement a TAP controller.

Refer to the IEEE 1149.1-2001 specification for further detail on electrical and pin protocol compliance requirements.

The NPC implements a Nexus controller state machine that transitions based on the state of the IEEE 1149.1-2001 state machine shown in *Figure 1170*. The Nexus controller state machine is defined by the IEEE-ISTO 5001-2001 standard. It is shown in *Figure 1171*.

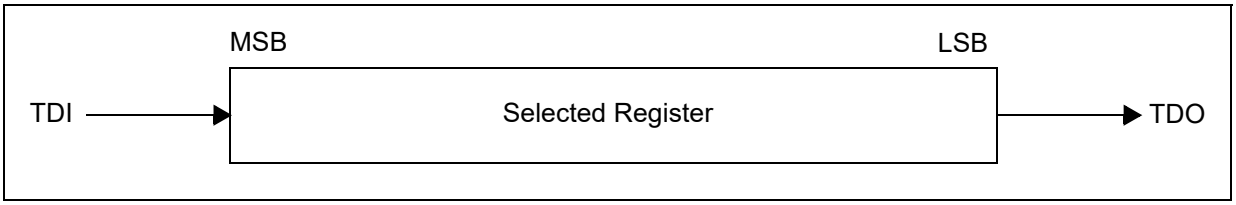
The instructions implemented by the NPC TAP controller are listed in *Table 1134*. The value of the NEXUS-ENABLE instruction is 0b0000. Each unimplemented instruction acts like the BYPASS instruction. The size of the NPC instruction register is 4-bits.

Table 1134. Implemented Instructions

Instruction Name	Private/ Public	Opcode	Description
NEXUS-ENABLE	public	0x0	Activate Nexus controller state machine to read and write NPC registers.
BYPASS	private	0xF	NPC BYPASS instruction. Also the value loaded into the NPC IR upon exit of reset.

Data is shifted between TDI and TDO starting with the least significant bit as illustrated in *Figure 1169*. This applies for the instruction register and all Nexus tool-mapped registers.

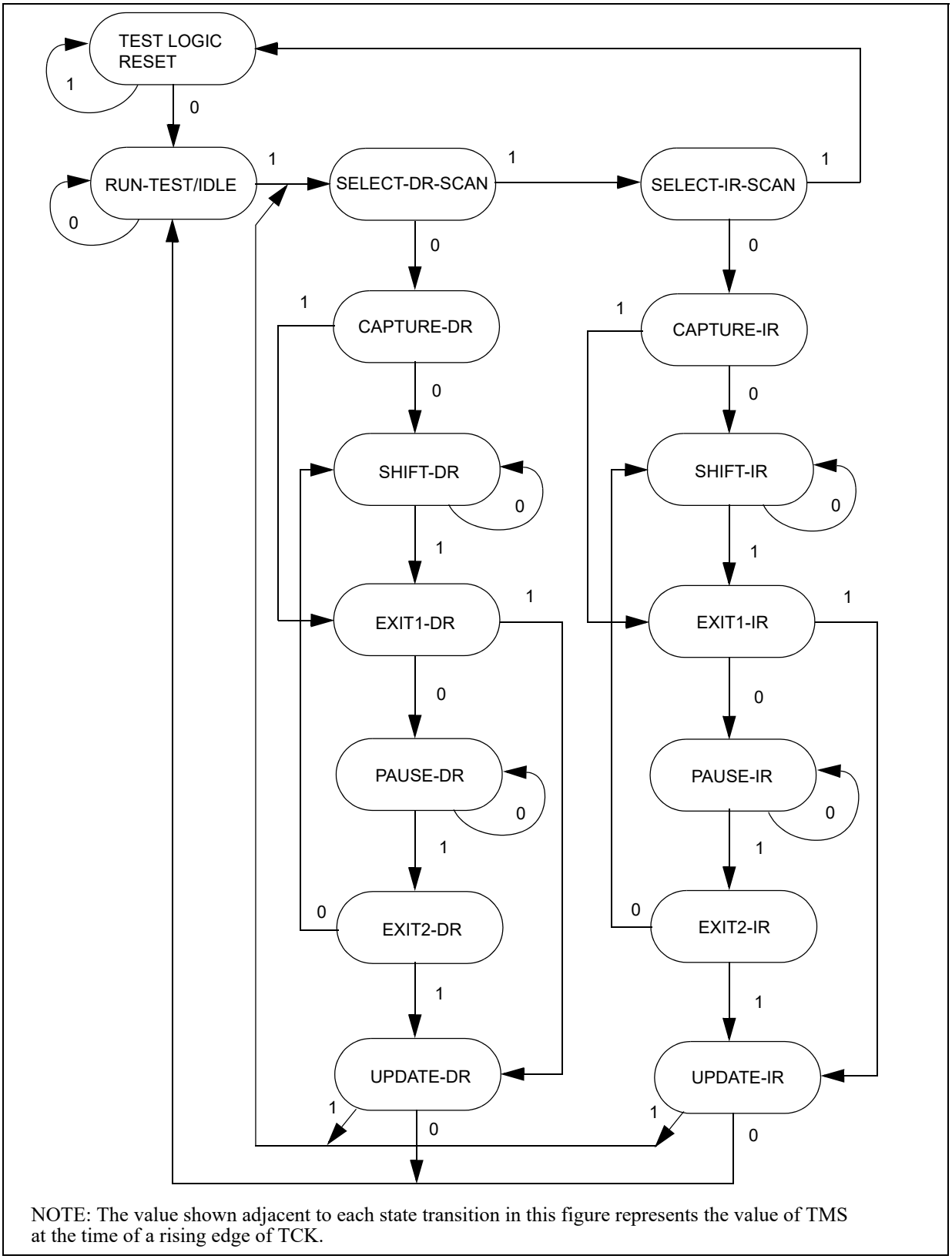
Figure 1169. Shifting Data Into Register



59.4.3.1 Enabling the NPC TAP controller

Assertion of the power-on reset signal or setting JCOMP to a value other than the NPC enable encoding resets the NPC TAP controller. When not in power-on reset, the NPC TAP controller is enabled by driving JCOMP with the NPC enable value and exiting the Test-Logic-Reset state. Loading the NEXUS-ENABLE instruction then grants access to Nexus debug.

Figure 1170. IEEE 1149.1-2001 TAP Controller State Machine



NOTE: The value shown adjacent to each state transition in this figure represents the value of TMS at the time of a rising edge of TCK.

59.4.3.2 Retrieving device IDCODE

The Nexus TAP controller does not implement the IDCODE instruction. However, the device identification message can be output by the NPC through the auxiliary output port or shifted out serially by accessing the Nexus Device ID register through the TAP. Transmission of the device identification message on the auxiliary output port MDO pins occurs immediately after a write to the PCR, if the NPC is enabled. Transmission of the device identification message serially via TDO is achieved by performing a read of the register contents as described in *Section 59.4.3.4*.

59.4.3.3 Loading NEXUS-ENABLE Instruction

Access to the NPC registers is enabled when the TAP controller instruction register is loaded with the NEXUS-ENABLE instruction. This instruction is shifted in via the SELECT-IR-SCAN path and loaded in the UPDATE-IR state. At this point, the Nexus controller state machine, shown in *Figure 1171*, transitions to the REG_SELECT state. The Nexus controller has three states: idle, register select, and data access. *Table 1135* illustrates the IEEE 1149.1 sequence to load the NEXUS-ENABLE instruction.

Figure 1171. NEXUS Controller State Machine

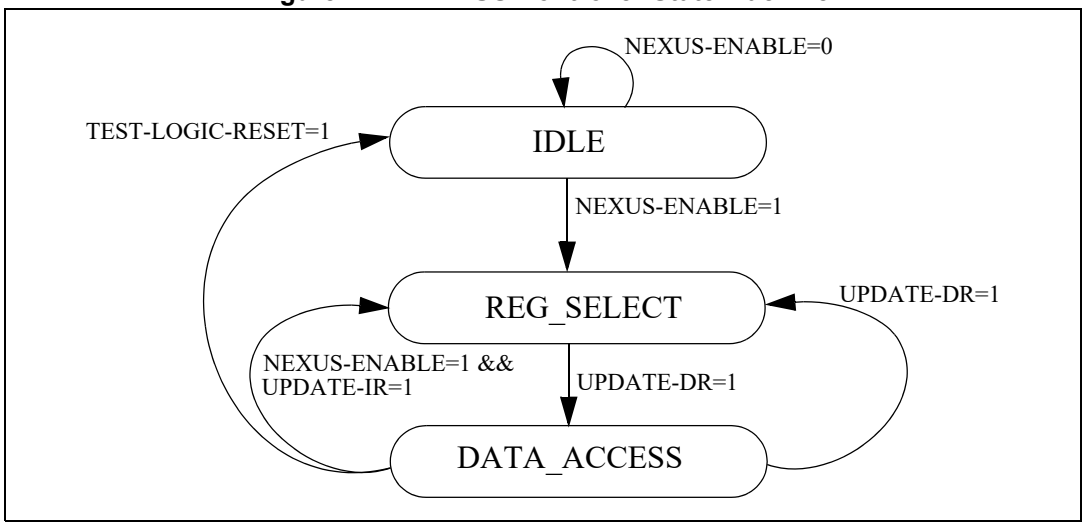


Table 1135. Loading NEXUS-ENABLE instruction

Clock	TMS	IEEE 1149.1 State	Nexus State	Description
0	0	RUN-TEST/IDLE	IDLE	IEEE 1149.1-2001 TAP controller in idle state
1	1	SELECT-DR-SCAN	IDLE	Transitional state
2	1	SELECT-IR-SCAN	IDLE	Transitional state
3	0	CAPTURE-IR	IDLE	Internal shifter loaded with current instruction
4	0	SHIFT-IR	IDLE	TDO becomes active, and the IEEE 1149.1-2001 shifter is ready. Shift in all but the last bit of the NEXUS_ENABLE instruction.
3 TCKS				
12	1	EXIT1-IR	IDLE	Last bit of instruction shifted in
13	1	UPDATE-IR	IDLE	NEXUS-ENABLE loaded into instruction register
14	0	RUN-TEST/IDLE	REG_SELECT	Ready to be read/write Nexus registers

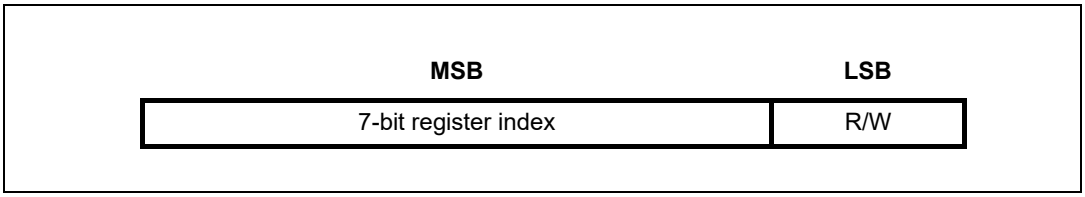
59.4.3.4 Selecting a Nexus Client Register

When the NEXUS-ENABLE instruction is decoded by the TAP controller, the input port allows development tool access to all Nexus registers. Each register has a 7-bit address index.

All register access is performed via the SELECT-DR-SCAN path. The Nexus Controller defaults to the REG_SELECT state when enabled. Accessing a register requires two passes through the SELECT-DR-SCAN path: one pass to select the register and the second pass to read/write the register.

The first pass through the SELECT-DR-SCAN path is used to enter an 8-bit Nexus command consisting of a read/write control bit in the LSB followed by a 7-bit register address index, as illustrated in *Figure 1172*. The read/write control bit is set to 1 for writes and 0 for reads.

Figure 1172. IEEE 1149.1 Controller Command Input



The second pass through the SELECT-DR-SCAN path is used to read or write the register data by shifting in the data (LSB first) during the SHIFT-DR state. When reading a register, the register value is loaded into the IEEE 1149.1-2001 shifter during the CAPTURE-DR state. When writing a register, the value is loaded from the IEEE 1149.1-2001 shifter to the register during the UPDATE-DR state. When reading a register, there is no requirement to shift out the entire register contents. Shifting may be terminated once the required number of bits have been acquired.

Table 1136 illustrates a sequence which writes a 32-bit value to a register

Table 1136. Write to a 32-Bit Nexus Client Register

Clock	TMS	IEEE 1149.1 State	Nexus State	Description
0	0	RUN-TEST/IDLE	REG_SELECT	IEEE 1149.1-2001 TAP controller in idle state
1	1	SELECT-DR-SCAN	REG_SELECT	First pass through SELECT-DR-SCAN path
2	0	CAPTURE-DR	REG_SELECT	Internal shifter loaded with current value of controller command input.
3	0	SHIFT-DR	REG_SELECT	TDO becomes active, and write bit and 6 bits of register index shifted in.
7 TCKs				
12	1	EXIT1-DR	REG_SELECT	Last bit of register index shifted into TDI
13	1	UPDATE-DR	REG_SELECT	Controller decodes and selects register
14	1	SELECT-DR-SCAN	DATA_ACCESS	Second pass through SELECT-DR-SCAN path
15	0	CAPTURE-DR	DATA_ACCESS	Internal shifter loaded with current value of register
16	0	SHIFT-DR	DATA_ACCESS	TDO becomes active, and outputs current value of register while new value is shifted in through TDI
31 TCKs				

Table 1136. Write to a 32-Bit Nexus Client Register (continued)

Clock	TMS	IEEE 1149.1 State	Nexus State	Description
48	1	EXIT1-DR	DATA_ACCESS	Last bit of current value shifted out TDO. Last bit of new value shifted in TDI.
49	1	UPDATE-DR	DATA_ACCESS	Value written to register
50	0	RUN-TEST/IDLE	REG_SELECT	Controller returned to idle state. It could also return to SELECT-DR-SCAN to write another register.

59.4.4 Nexus JTAG Port Sharing

Each of the individual Nexus blocks on the device implements a TAP controller for accessing its registers. When Nexus has ownership of the TAP, only the block whose NEXUS-ENABLE instruction is loaded has control of the TAP. This allows the interface to all of these individual TAP controllers to appear to be a single port from outside the device. If no register is selected as the shift path for a Nexus block, that block acts like a single-bit shift register, or bypass register.

59.4.5 MCKO Control

MCKO is an output clock to the development tools used for the timing of $\overline{\text{MSEO}}$ and MDO pin functions. MCKO is derived from the system clock and its frequency is determined by the value of the MCKO_DIV field in the PCR. Possible operating frequencies include system clock, one-half system clock, one-quarter system clock, and one-eighth system clock speed.

The NPC also generates an MCKO clock gating control output signal. This output can be used by the MCKO generation logic to gate the transmission of MCKO when the auxiliary port is enabled but not transmitting messages. The setting of the MCKO_GT bit inside the PCR determines whether or not MCKO gating control is active. The MCKO_GT bit resets to a logic 0. In this state gating of MCKO is disabled. To enable gating of MCKO, the MCKO_GT bit in the PCR is written to a logic 1.

59.4.6 $\overline{\text{EVTO}}$ Sharing

The NPC block controls sharing of the $\overline{\text{EVTO}}$ output between all Nexus clients that produce an $\overline{\text{EVTO}}$ signal. The NPC assumes incoming $\overline{\text{EVTO}}$ signals will be asserted for one system clock period. After receiving a single clock period of asserted $\overline{\text{EVTO}}$ from any Nexus client, the NPC latches the result, and drives $\overline{\text{EVTO}}$ for one MCKO period on the following clock. When there is no active MCKO, such as in disabled mode, the NPC drives $\overline{\text{EVTO}}$ for two system clock periods. $\overline{\text{EVTO}}$ sharing is active as long as the NPC is not in reset.

59.4.7 Nexus Reset Control

The JCOMP input that is used as the primary reset signal for the NPC is also used by the NPC to generate a single-bit reset signal for other Nexus blocks. If JCOMP is negated, an internal reset is asserted, indicating that all Nexus modules should be held in reset.

59.5 Initialization/Application Information

59.5.1 Accessing NPC tool-mapped registers

To initialize the TAP for Nexus register accesses, the following sequence is required:

1. Enable the Nexus TAP controller
2. Load the TAP controller with the NEXUS-ENABLE instruction

To write control data to NPC tool-mapped registers, the following sequence is required:

1. Write the 7-bit register index and set the write bit to select the register with a pass through the SELECT-DR-SCAN path in the TAP controller state machine.
2. Write the register value with a second pass through the SELECT-DR-SCAN path. Note that the prior value of this register is shifted out during the write.

To read status and control data from NPC tool-mapped registers, the following sequence is required:

1. Write the 7-bit register index and clear the write bit to select register with a pass through SELECT-DR-SCAN path in the TAP controller state machine.
2. Read the register value with a second pass through the SELECT-DR-SCAN path. Data shifted in is ignored.

See the IEEE-ISTO 5001-2001 standard for more detail.

60 Sequence Processing Unit (SPU)

60.1 Introduction

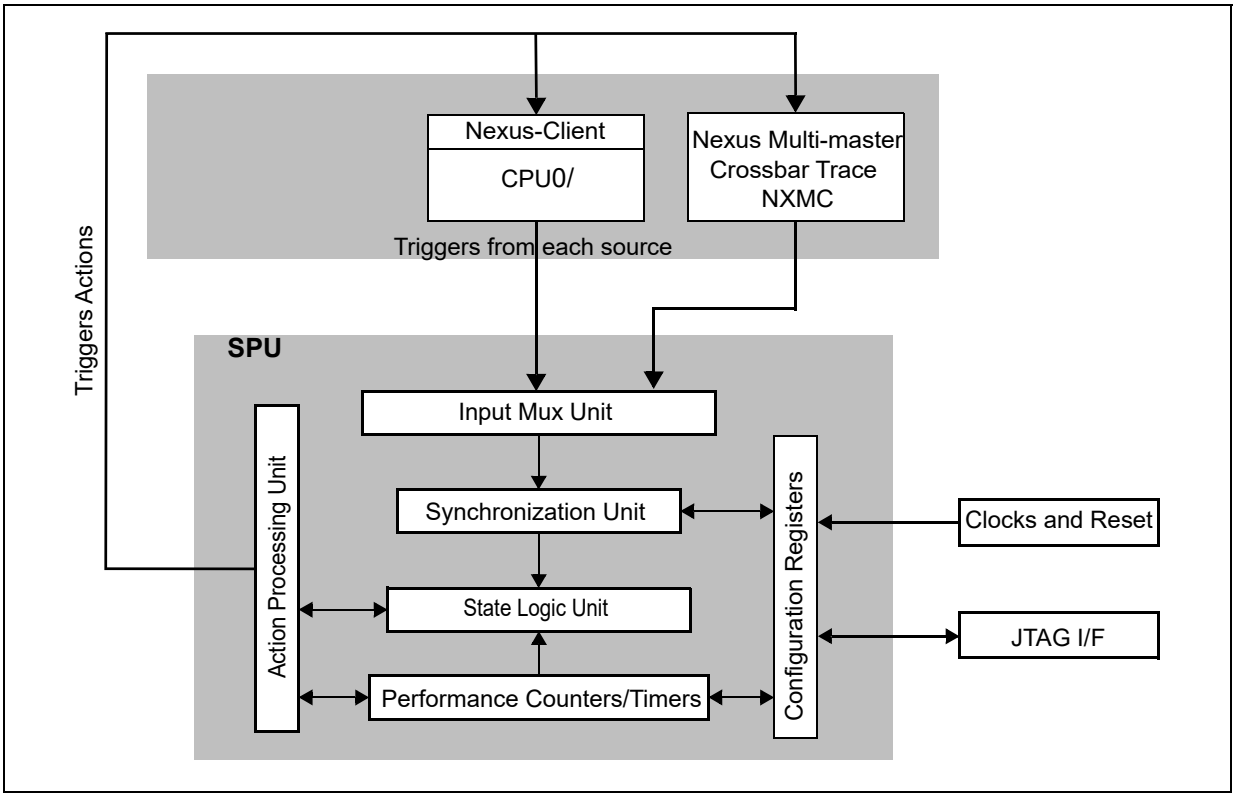
The Sequence Processing Unit (SPU) provides an on-device trigger functions similar to those found on a logic analyzer. In a complex SoC, performance monitor functions are available at two levels: system level and CPU level. Complex trigger and system performance monitor functions are implemented in the SPU module. System level performance monitor functions are integrated into the SPU complex trigger logic, and thus allow counting and timing of any debug trigger combinations supported by the SPU.

Various clients such as the cores or crossbar switch, generate watchpoints and triggers when operating in their debug mode. The SPU collects these triggers (such as interrupt occurrence, address watchpoint, and so on) and uses them as conditions to sequence through states, with resultant actions (such as start/stop trace, start/stop counter, capture timebase, and so on). The SPU provides the capability to create complex debug triggers. By configuring each of the events to trigger specific actions, complex logic-analyzer-like behavior can be achieved, providing vital real-time visibility and debug ability of the activities of the system.

60.1.1 SPU block diagram

Figure 1173 shows a top-level block diagram of the SPU.

Figure 1173. SPU block diagram



60.1.2 Overview

The SPU is able to generate complex debug events, based upon input events from sources (CPUs, and system bus clients) throughout the SoC. The SPU has the ability to create a state machine to trigger a debug action. Single or multiple actions can be triggered by a state machine, which results in debug events being created. There are a selection of timers and counters available to assist comparison events.

Complex trigger support is implemented in the State Logic Unit (SLU). There are numerous inputs coming from different clients. Users can choose 63 inputs from a set of 209 inputs at the level one Mux. There are actually 64 inputs to the Muxes, but one of the inputs is permanently connected to a high level so users can only choose 63 of the inputs. Of these 63 inputs, users can select 16 inputs for each SLU, shown in *Figure 1208*.

One SLU consists of four input AND gates, the output of which are ORed together. One of the inputs of each AND gate has the option to invert it. Also, the output of each AND gate can be optionally inverted. Finally, the output of each SLU has the option of inversion. The block diagram for one SLU state is shown in *Figure 1209*. The SPU includes eight SLUs states.

Users can utilize these states to form a configurable state machine. This allows users to join states together with the if-then-else operations to create a sequence. Each state in a sequence has the ability to move to another state based on the true condition from the state logic, and the ability to route to another state based on the a false condition from the state logic. This is shown in *Figure 1210* and *Figure 1211*. At a given time, a maximum of four sequences are allowed. Each sequence can have one state or maximum of eight states. Each state can only be used in one unique sequence. The number of states in all the sequences cannot exceed eight. Each state has the ability to trigger one to four actions based on a true/false condition in the active sequence.

An example of typical complex trigger for the SPU would be: if FunctionA is entered, monitor CPU writes to VariableB. If CPU writes a value to VariableB which is 0x100, then enable trace for the halt CPU.

The SPU consists of the following sub-blocks:

- Input Mux Unit: receives the numerous triggers from various clients and reduces them down into a user selected set. The muxing unit selects sixteen inputs for each sequencing SLU from a set of 63 static inputs. This selection is done using two levels of muxing. At the first level of muxing, users select 63 out of 209 inputs. At the second level of muxing, out of these 63 inputs, one input can be selected for each of the four input AND gates. There are 128 64 $\times$ 1 Muxes to cater the needs of 128 inputs of all the eight states. This input mux selection logic is shown in *Figure 1208*.
- State Logic Units (SLU): responsible for performing operations on the selected input events to produce desired events. Each SLU has a control register that manages how the various inputs are combined to produce a desired event. Only one input of each four input AND gate can be independently inverted. The combined event can also be optionally inverted and, coupled with DeMorgan's law, allows for generalized signal combinations. This is shown in *Figure 1209*.
- Action Processing Unit: takes an action request from a SLU state and converts this into one or multiple actions. The action unit can be programmed to trigger various actions in response to each processed event from the SLU unit. The user can define the actions associated with each state from each SLU.
- Counters/Timers: the SPU implements 16 32-bit timer/counter functionality. Timer or counter selection is done via a configurable register in the SPU. Each counter/timer

has its own control register. A 32-bit comparator value can optionally be compared against the timer/counter. These registers are accessible via JTAG.

60.1.3 Features

The features of the SPU are as follows:

- clients:
  - 1 CPU
  - 2 System Bus clients (NXMC)
- Input watchpoints/triggers selection
- Input debug triggers for instruction execution selection
- Input interrupt selection
- Sequences and states criteria
  - Up to 4 user programmable number of active/simultaneous sequences
  - Up to 8 user programmable states that can be split between all four sequences
  - Each state can only be used in one unique sequence
  - A sequence can consist of a single state or any number of states up to 8
- State logic
  - Each state logic consist of sum of products (SOP)
  - SOP consist of four 4-input AND gates, and one 4-input OR gate
  - Optional inversion on one input of each AND gate of each state logic
  - Optional inversion on output of each AND gate
  - Optional inversion on output of OR gate of each state
  - States are joined together with if-then-else statement to create a sequence
- SLU events
  - Up to 8 user programmable derived events/actions
  - Any state can optionally trigger of 1 to 4 actions based on a true condition
  - Any state can optionally trigger of 1 to 4 actions based on a false condition
  - User selection of trigger actions generated in response to each programmed trigger
- SPU counters/timers
  - 16 32-bit counters/timers
  - Timer/counter configurability selection via a register
  - Start/increment when used as timer/counter respectively
  - Reset when used as timer/counter respectively
  - Counter value compare match
  - Counter overflow event status
  - Generation of a greater than condition against a 32-bit comparator value (no support for range condition from single counter)
  - Range support using two counters
  - Use of counter/timer registers to generate counted or timed triggers
  - Use of counter/timer registers to capture system performance statistics
  - Each timer/counter value has the ability to be inserted into the Nexus trace stream upon an SPU action
  - Access to timer/counter values via JTAG
- Prescaler for timer mode only

  - Configurable prescaler of /1, /4, /16 or /32 of the clock for timer operation inside the SPU
- Asynchronous interfaces
  - SPU to/from CPU (fastest clock)
  - SPU to/from NXMC
  - SPU to/from DCI
  - SPU to/from JTAGM
- Interfaces
  - CPU
  - System bus
  - DCI
- Configuration registers: 32-bit double word aligned configuration registers (byte and halfword access not supported)
- SPU actions: SPU block triggers various actions based on the derived events, these actions are defined in *Section 60.2*

60.2 SPU actions

Sequence actions describe what is triggered. A sequence state may only have a GoTo action to determine the next sequence state. The complete set of actions are as follows:

- Start/stop trace for a source
- Start/stop/increment counter/timer counter/timer[0–15]
- Reset counter/timer[0–15]
- Halt1: single signal (EVTI0) that can be used to halt a pre-configured DCI group1
- Halt2: single signal (EVTI1) that can be used to halt a pre-configured DCI group2
- Single interrupt for INTC to interrupt a CPU: de-assert (cleared) via JTAGM
- Generate a pulse on the EVTO pin: output pulse over EVTO1 and EVTO2
- Start or Stop Performance Counters (PMC) for a CPU source: Start/Stop CPUn
- Optional trace group 1 (Start/Stop Trace a pre-configured set of trace signals): four (DTM,PTM,WTM,OTM) independent start/stop trace groups can be formed. One or more clients (CPUn and Nexus) can be part of a particular group.

Note: GoTo is not part of the Action list described above but it is implemented in the state logic to move from one state to another. If there is nothing defined for GoTo, it signifies the end of the sequence.

In the condition where a state (State n) updates a counter and the next state in the sequence (State n+1) evaluates the value of the same counter (for example, to compare if the counter is less-than a certain value), the update to the counter (from State n) is not applied before the counter evaluation in the next state (State n+1). This results in the old value of the counter being evaluated (in State n+1), rather than the desired updated value. This is due to hardware limitations of the SPU as the update to the counter does not complete until the clock cycle after the evaluation of the next state.

To workaround this hardware limitation, a delay should be inserted between the update to a counter and the evaluation of this counter. A dummy state can be used between the counter update (State n) and the counter evaluation (State n+1). A dummy state is a state with any

input, that on both then/else conditions, the action proceeds to the next state (State n+1) where the counter evaluation occurs. The insertion of this dummy state provides sufficient time for the update of the counter to be applied, in time for the counter evaluation. The same outcome is also possible in a complex sequence with multiple states by scheduling states so that counter update and counter evaluation are not performed on consecutive states in a sequence.

60.3 Operation mode

60.3.1 SPU enable/disable

By default, the SPU is in disabled mode. In this state:

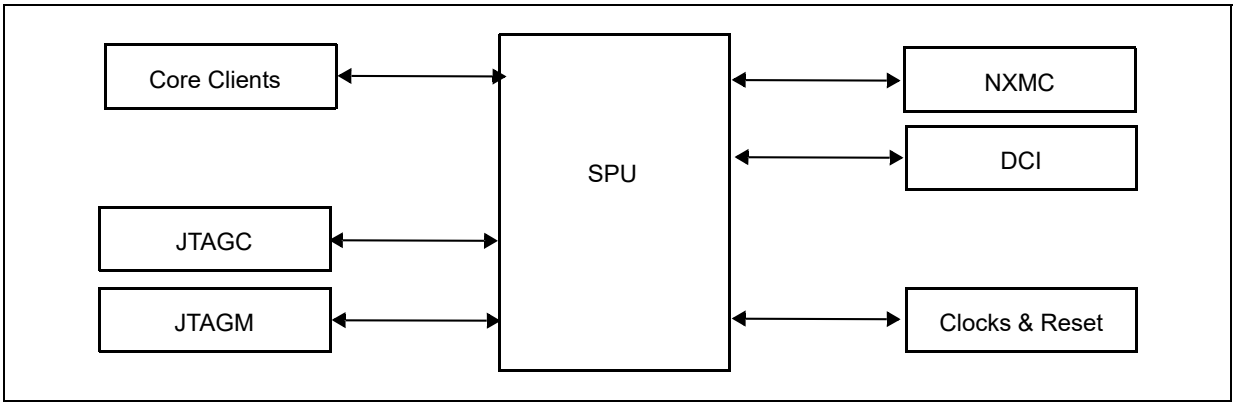
- All events/inputs are disabled such that no counting or other actions occur.
- All the SPU control registers are accessible.

The user must configure the control registers before enabling the SPU via the SPU Enable/Disable register.

60.4 SPU module interface

The SPU is connected to many clients such as the cores, system bus and Nexus clients. The SPU also interfaces to the DCI and the JTAG modules as shown in *Figure 1174*.

Figure 1174. SPU module interface



60.5 Register definition

The SPU contains 32-bit aligned registers (byte and half-word access are not supported). All configuration registers in the SPU are JTAG registers such that these registers are accessible via JTAG interface only. Write access to reserved locations have no impact while read access to reserved locations always return 0. These registers have Nexus reset and JTAG reset control.

Note: The SPU does not check for correctness of the programmed values in the registers and programmers must ensure that correct values are being written.

Table 1137 lists the memory mapped registers which control the operation of the SPU. There are two instructions for the SPU registers:

- Instruction to enable the SLU (first set of registers with offsets 0x01–0x49) registers is 4'h3.
- Instruction to enable the Counter Control Unit (second set of registers with offsets 0x01–0x72) is 4'h2.

Table 1137. SPU register summary

Offset	Register	Location
0x01	Level1 Mux Selection 0 (L1SEL0)	*Section 60.5.1.1.1*
0x02	Level1 Mux Selection 1 (L1SEL1)	*Section 60.5.1.1.2*
0x03	Level1 Mux Selection 2 (L1SEL2)	*Section 60.5.1.1.3*
0x04	Level1 Mux Selection 3 (L1SEL3)	*Section 60.5.1.1.4*
0x05	Level1 Mux Selection 4(L1SEL4)	*Section 60.5.1.1.5*
0x06	Level1 Mux Selection 5 (L1SEL5)	*Section 60.5.1.1.6*
0x07	Level1 Mux Selection 6 (L1SEL6)	*Section 60.5.1.1.7*
0x08	Level1 Mux Selection 7 (L1SEL7)	*Section 60.5.1.1.8*
0x09–0x0A	Reserved for future use	
0x0B	Input Trigger Level Detection 2 (ITLD2)	*Section 60.5.1.1.9*
0x0C	Input Trigger Level Detection 3 (ITLD3)	*Section 60.5.1.1.10*
0x0D	CPU0 Processor Except Vector Prefix (C0PEVP)	*Section 60.5.1.1.11*
0x0E	Reserved	
0x0F	Reserved	
0x10–0x11	Reserved for future use	
0x12	CPU0 Interrupt Priority Selection (C0PIS)	*Section 60.5.1.1.12*
13	Reserved	
14	Reserved	
0x15–0x16	Reserved for future use	
0x17	Level2 Mux State0 Selection 0 (L20SEL0)	*Section 60.5.1.2.1*
0x18	Level2 Mux State0 Selection 1 (L20SEL1)	*Section 60.5.1.2.2*
0x19	Level2 Mux State0 Selection 2 (L20SEL2)	*Section 60.5.1.2.3*
0x1A	Level2 Mux State0 Selection 3 (L20SEL3)	*Section 60.5.1.2.4*
0x1B	Level2 Mux State1 Selection 0 (L21SEL0)	*Section 60.5.1.2.1*
0x1C	Level2 Mux State1 Selection 1 (L21SEL1)	*Section 60.5.1.2.2*
0x1D	Level2 Mux State1 Selection 2 (L21SEL2)	*Section 60.5.1.2.3*
0x1E	Level2 Mux State1 Selection 3 (L21SEL3)	*Section 60.5.1.2.4*
0x1F	Level2 Mux State2 Selection 0 (L22SEL0)	*Section 60.5.1.2.1*
0x20	Level2 Mux State2 Selection 1 (L22SEL1)	*Section 60.5.1.2.2*

Table 1137. SPU register summary (continued)

Offset	Register	Location
0x21	Level2 Mux State2 Selection 2 (L22SEL2)	*Section 60.5.1.2.3*
0x22	Level2 Mux State2 Selection 3 (L22SEL3)	*Section 60.5.1.2.4*
0x23	Level2 Mux State3 Selection 0 (L23SEL0)	*Section 60.5.1.2.1*
0x24	Level2 Mux State3 Selection 1 (L23SEL1)	*Section 60.5.1.2.2*
0x25	Level2 Mux State3 Selection 2 (L23SEL2)	*Section 60.5.1.2.3*
0x26	Level2 Mux State3 Selection 3 (L23SEL3)	*Section 60.5.1.2.4*
0x27	Level2 Mux State4 Selection 0 (L24SEL0)	*Section 60.5.1.2.1*
0x28	Level2 Mux State4 Selection 1 (L24SEL1)	*Section 60.5.1.2.2*
0x29	Level2 Mux State4 Selection 2 (L24SEL2)	*Section 60.5.1.2.3*
0x2A	Level2 Mux State4 Selection 3 (L24SEL3)	*Section 60.5.1.2.4*
0x2B	Level2 Mux State5 Selection 0 (L25SEL0)	*Section 60.5.1.2.1*
0x2C	Level2 Mux State5 Selection 1 (L25SEL1)	*Section 60.5.1.2.2*
0x2D	Level2 Mux State5 Selection 2 (L25SEL2)	*Section 60.5.1.2.3*
0x2E	Level2 Mux State6 Selection 3 (L25SEL3)	*Section 60.5.1.2.4*
0x2F	Level2 Mux State6 Selection 0 (L26SEL0)	*Section 60.5.1.2.1*
0x30	Level2 Mux State6 Selection 1 (L26SEL1)	*Section 60.5.1.2.2*
0x31	Level2 Mux State6 Selection 2 (L26SEL2)	*Section 60.5.1.2.3*
0x32	Level2 Mux State6 Selection 3 (L26SEL3)	*Section 60.5.1.2.4*
0x33	Level2 Mux State7 Selection 0 (L27SEL0)	*Section 60.5.1.2.1*
0x34	Level2 Mux State7 Selection 1 (L27SEL1)	*Section 60.5.1.2.2*
0x35	Level2 Mux State7 Selection 2 (L27SEL2)	*Section 60.5.1.2.3*
0x36	Level2 Mux State7 Selection 3 (L27SEL3)	*Section 60.5.1.2.4*
0x37	Input/Output Inversion Control State0 (IOIC0)	*Section 60.5.1.3.1*
0x38	Input/Output Inversion Control State1 (IOIC1)	*Section 60.5.1.3.1*
0x39	Input/Output Inversion Control State2 (IOIC2)	*Section 60.5.1.3.1*
0x3A	Input/Output Inversion Control State3 (IOIC3)	*Section 60.5.1.3.1*
0x3B	Input/Output Inversion Control State4 (IOIC4)	*Section 60.5.1.3.1*
0x3C	Input/Output Inversion Control State5 (IOIC5)	*Section 60.5.1.3.1*
0x3D	Input/Output Inversion Control State6 (IOIC6)	*Section 60.5.1.3.1*
0x3E	Input/Output Inversion Control State7 (IOIC7)	*Section 60.5.1.3.1*
0x3F	Sequence Control (SCTRL)	*Section 60.5.1.3.2*
0x40	State0 Goto Control (S0GC)	*Section 60.5.1.3.3*
0x41	State1 Goto Control (S1GC)	*Section 60.5.1.3.3*
0x42	State2 Goto Control (S2GC)	*Section 60.5.1.3.3*
0x43	State3 Goto Control (S3GC)	*Section 60.5.1.3.3*

Table 1137. SPU register summary (continued)

Offset	Register	Location
0x44	State4 Goto Control (S4GC)	*Section 60.5.1.3.3*
0x45	State5 Goto Control (S5GC)	*Section 60.5.1.3.3*
0x46	State6 Goto Control (S6GC)	*Section 60.5.1.3.3*
0x47	State7 Goto Control (S7GC)	*Section 60.5.1.3.3*
0x48	SLU Status (SS)	*Section 60.5.1.4*
0x49	SPU Enable (SE)	*Section 60.5.1.5*
0x01	State0 True Action 0 (S0TA0)	*Section 60.5.1.6.1*
0x02	State0 True Action 1 (S0TA1)	*Section 60.5.1.6.2*
0x03	State1 True Action 0 (S1TA0)	*Section 60.5.1.6.1*
0x04	State1 True Action 1 (S1TA1)	*Section 60.5.1.6.2*
0x05	State2 True Action 0 (S2TA0)	*Section 60.5.1.6.1*
0x06	State2 True Action 1 (S2TA1)	*Section 60.5.1.6.2*
0x07	State3 True Action 0 (S3TA0)	*Section 60.5.1.6.1*
0x08	State3 True Action 1 (S3TA1)	*Section 60.5.1.6.2*
0x09	State4 True Action 0 (S4TA0)	*Section 60.5.1.6.1*
0x0A	State4 True Action 1 (S4TA1)	*Section 60.5.1.6.2*
0x0B	State5 True Action 0 (S5TA0)	*Section 60.5.1.6.1*
0x0C	State5 True Action 1(S5TA1)	*Section 60.5.1.6.2*
0x0D	State6 True Action 0 (S6TA0)	*Section 60.5.1.6.1*
0x0E	State6 True Action 1(S6TA1)	*Section 60.5.1.6.2*
0x0F	State7 True Action 0 (S7TA0)	*Section 60.5.1.6.1*
0x10	State7 True Action 1(S7TA1)	*Section 60.5.1.6.2*
0x11	State0 False Action 0 (S0FA0)	*Section 60.5.1.6.3*
0x12	State0 False Action 1 (S0FA1)	*Section 60.5.1.6.4*
0x13	State1 False Action 0 (S1FA0)	*Section 60.5.1.6.3*
0x14	State1 False Action 1(S1FA1)	*Section 60.5.1.6.4*
0x15	State2 False Action 0 (S2FA0)	*Section 60.5.1.6.3*
0x16	State2 False Action 1(S2FA1)	*Section 60.5.1.6.4*
0x17	State3 False Action 0 (S3FA0)	*Section 60.5.1.6.3*
0x18	State3 False Action 1(S3FA0)	*Section 60.5.1.6.4*
0x19	State4 False Action 0 (S4FA0)	*Section 60.5.1.6.3*
0x1A	State4 False Action 1 (S4FA1)	*Section 60.5.1.6.4*
0x1B	State5 False Action 0 (S5FA0)	*Section 60.5.1.6.3*
0x1C	State5 False Action 1 (S5FA1)	*Section 60.5.1.6.4*
0x1D	State6 False Action 0 (S6FA0)	*Section 60.5.1.6.3*

Table 1137. SPU register summary (continued)

Offset	Register	Location
0x1E	State6 False Action 1 (S6FA1)	*Section 60.5.1.6.4*
0x1F	State7 False Action 0 (S7FA0)	*Section 60.5.1.6.3*
0x20	State7 False Action 1 (S7FA1)	*Section 60.5.1.6.4*
0x21	DTM Trace Group Configuration (DTMTGC)	*Section 60.5.1.6.5*
0x22	PTM Trace Group Configuration (PTMTGC)	*Section 60.5.1.6.5*
0x23	OTM Trace Group Configuration (OTMTGC)	*Section 60.5.1.6.5*
0x24	WTM Trace Group Configuration (WTMTGC)	*Section 60.5.1.6.5*
0x25	Interrupt Status (INTS)	*Section 60.5.1.6.6*
0x26	Counter Control 0 (CCTRL0)	*Section 60.5.1.7*
0x27	Counter Control 1 (CCTRL1)	*Section 60.5.1.7*
0x28	Counter Control 2 (CCTRL2)	*Section 60.5.1.7*
0x29	Counter Control 3 (CCTRL3)	*Section 60.5.1.7*
0x2A	Counter Control 4 (CCTRL4)	*Section 60.5.1.7*
0x2B	Counter Control 5 (CCTRL5)	*Section 60.5.1.7*
0x2C	Counter Control 6 (CCTRL6)	*Section 60.5.1.7*
0x2D	Counter Control 7 (CCTRL7)	*Section 60.5.1.7*
0x2E	Counter Control 8 (CCTRL8)	*Section 60.5.1.7*
0x2F	Counter Control 9 (CCTRL9)	*Section 60.5.1.7*
0x30	Counter Control 10 (CCTRL10)	*Section 60.5.1.7*
0x31	Counter Control 11 (CCTRL11)	*Section 60.5.1.7*
0x32	Counter Control 12 (CCTRL12)	*Section 60.5.1.7*
0x33	Counter Control 13 (CCTRL13)	*Section 60.5.1.7*
0x34	Counter Control 14 (CCTRL14)	*Section 60.5.1.7*
0x35	Counter Control 15 (CCTRL15)	*Section 60.5.1.7*
0x36	Counter Compare 0(CCMP0)	*Section 60.5.1.8*
0x37	Counter Compare 1 (CCMP1)	*Section 60.5.1.8*
0x38	Counter Compare 2 (CCMP2)	*Section 60.5.1.8*
0x39	Counter Compare 3 (CCMP3)	*Section 60.5.1.8*
0x3A	Counter Compare 4 (CCMP4)	*Section 60.5.1.8*
0x3B	Counter Compare 5 (CCMP5)	*Section 60.5.1.8*
0x3C	Counter Compare 6 (CCMP6)	*Section 60.5.1.8*
0x3D	Counter Compare 7 (CCMP7)	*Section 60.5.1.8*
0x3E	Counter Compare 8 (CCMP8)	*Section 60.5.1.8*
0x3F	Counter Compare 9 (CCMP9)	*Section 60.5.1.8*
0x40	Counter Compare 10 (CCMP10)	*Section 60.5.1.8*

Table 1137. SPU register summary (continued)

Offset	Register	Location
0x41	Counter Compare 11 (CCMP11)	*Section 60.5.1.8*
0x42	Counter Compare 12 (CCMP12)	*Section 60.5.1.8*
0x43	Counter Compare 13 (CCMP13)	*Section 60.5.1.8*
0x44	Counter Compare 14 (CCMP14)	*Section 60.5.1.8*
0x45	Counter Compare 15 (CCMP15)	*Section 60.5.1.8*
0x46	Counter Compare Status (CCOMS)	*Section 60.5.1.9.1*
0x47	Counter Overflow Status (COS)	*Section 60.5.1.9.2*
0x48	Counter Capture Status (CCAPS)	*Section 60.5.1.9.3*

60.5.1 Registers description

60.5.1.1 Level1 input Mux configurations registers

A set of input Mux control registers is used to narrow down the large number of SPU watchpoints to a manageable set. There are sixty-four 8 × 1 (eight inputs and one output) multiplexers. Three bits per Mux are used to select one input out of 8 inputs. 8 Muxes are configured with each 32-bit register; in other words, 8 inputs out of 64 for each register. This provides the flexibility to the programmer to route any input trigger to any state input. The detail of this selection is shown in *Figure 1208*.

Note: Users cannot program MUX 63 in L1SEL7 (the 64th mux). All eight inputs of the 63rd mux are tied to high and the output is always high (1'b1).

60.5.1.1.1 Level1 Mux Selection 0 (L1SEL0)

Figure 1175 shows the format of the L1SEL0 register. The values for the fields of the L1SEL0 register are device dependent. For a description of these fields, see Calibration and Debug (or Debug and Trace) chapter that describes how the modules are configured.

Offset: 0x01 Access: Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	MUX 7			0	MUX 6			0	MUX 5			0	MUX 4		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	MUX 3			0	MUX 2			0	MUX 1			0	MUX 0		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1175. Level1 Mux Selection 0 (L1SEL0)

60.5.1.1.2 Level1 Mux Selection 1 (L1SEL1)

Figure 1176 shows the format of the L1SEL1 register. The values for the fields of the L1SEL1 register are device dependent. For a description of these fields, see Calibration and Debug (or Debug and Trace) chapter that describes how the modules are configured.

Offset: 0x02 Access: Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	MUX 15			0	MUX 14			0	MUX 13			0	MUX 12		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	MUX 11			0	MUX 10			0	MUX 9			0	MUX 8		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1176. Level1 Mux Selection 1 (L1SEL1)

60.5.1.1.3 Level1 Mux Selection 2 (L1SEL2)

Figure 1177 shows the format of the L1SEL2 register. The values for the fields of the L1SEL2 register are device dependent. For a description of these fields, see Calibration and Debug (or Debug and Trace) chapter that describes how the modules are configured.

Offset: 0x03 Access: Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	MUX 23			0	MUX 22			0	MUX 21			0	MUX 20		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	MUX 19			0	MUX 18			0	MUX 17			0	MUX 16		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1177. Level1 Mux Selection 2 (L1SEL2)

60.5.1.1.4 Level1 Mux Selection 3 (L1SEL3)

Figure 1178 shows the format of the L1SEL3 register. The values for the fields of the L1SEL3 register are device dependent. For a description of these fields, see Calibration and Debug (or Debug and Trace) chapter that describes how the modules are configured.

Offset: 0x04 Access: Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	MUX 31			0	MUX 30			0	MUX 29			0	MUX 28		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	MUX 27			0	MUX 26			0	MUX 25			0	MUX 24		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1178. Level1 Mux Selection 3 (L1SEL3)

60.5.1.1.5 Level1 Mux Selection 4 (L1SEL4)

Figure 1179 shows the format of the L1SEL4 register. The values for the fields of the L1SEL4 register are device dependent. For a description of these fields, see Calibration and Debug (or Debug and Trace) chapter that describes how the modules are configured.

Offset: 0x05 Access: Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	MUX 39			0	MUX 38			0	MUX 37			0	MUX 36		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	MUX 35			0	MUX 34			0	MUX 33			0	MUX 32		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1179. Level1 Mux Selection 4 (L1SEL4)

60.5.1.1.6 Level1 Mux Selection 5 (L1SEL5)

Figure 1180 shows the format of the L1SEL5 register. The values for the fields of the L1SEL5 register are device dependent. For a description of these fields, see Calibration and Debug (or Debug and Trace) chapter that describes how the modules are configured.

Offset: 0x06 Access: Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	MUX 47			0	MUX 46			0	MUX 45			0	MUX 44		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	MUX 43			0	MUX 42			0	MUX 41			0	MUX 40		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1180. Level1 Mux Selection 5 (L1SEL5)

60.5.1.1.7 Level1 Mux Selection 6 (L1SEL6)

Figure 1181 shows the format of the L1SEL6 register.The values for the fields of the L1SEL6 register are device dependent. For a description of these fields, see Calibration and Debug (or Debug and Trace) chapter that describes how the modules are configured.

Offset: 0x07 Access: Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	MUX 55			0	MUX 54			0	MUX 53			0	MUX 52		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	MUX 51			0	MUX 50			0	MUX 49			0	MUX 48		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1181. Level1 Mux Selection 6 (L1SEL6)

60.5.1.1.8 Level1 Mux Selection 7 (L1SEL7)

Figure 1182 shows the format of the L1SEL7 register. The values for the fields of the L1SEL7 register are device dependent. For a description of these fields, see Calibration and Debug (or Debug and Trace) chapter that describes how the modules are configured. All eight inputs of MUX 63 are tied to logic high (1'b1) and output is always high.

Offset: 0x08 Access: Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	MUX 63			0	MUX 62			0	MUX 61			0	MUX 60		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	MUX 59			0	MUX 58			0	MUX 57			0	MUX 56		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1182. Level1 Mux Selection 7 (L1SEL7)

60.5.1.1.9 Input triggers level detection control 2 (ITLD2)

Only a rising edge is detected on the outputs of the first 32 level1 Muxes and therefore these values are not programmable. The last 32 level1 Muxes (Mux 32–63) trigger level is programmable in the Input Triggers Level Detection Control 2 and 3 registers. *Figure 1183* shows the format of the ITLD2 register. All of the Mux fields in the ITLD2 register have the same description.

Offset: 0x0B Access: Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Mux47		Mux46		Mux45		Mux44		Mux43		Mux42		Mux41		Mux40	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	Mux39		Mux38		Mux37		Mux36		Mux35		Mux34		Mux33		Mux32	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1183. Input triggers level detection control 2 (ITLD2)

The ITLD2 registers Mux fields are described in *Table 1138*.

Table 1138. ITLD2 register field descriptions

Field	Description
0:1 Mux*47*	Mux47 Output Signal Detection 00 Level is passed after synchronization (default) 01 Posedge detection 10 Toggle detection 11 Level is passed after synchronization
2:3 Mux*46*	Mux46 Output Signal Detection 00 Level is passed after synchronization (default) 01 Posedge detection 10 Toggle detection 11 Level is passed after synchronization

Table 1138. ITLD2 register field descriptions (continued)

Field	Description
4:5 Mux*45*	Mux45 Output Signal Detection 00 Level is passed after synchronization (default) 01 Posedge detection 10 Toggle detection 11 Level is passed after synchronization
6:7 Mux*44*	Mux44 Output Signal Detection 00 Level is passed after synchronization (default) 01 Posedge detection 10 Toggle detection 11 Level is passed after synchronization
8:9 Mux*43*	Mux43 Output Signal Detection 00 Level is passed after synchronization (default) 01 Posedge detection 10 Toggle detection 11 Level is passed after synchronization
10:11 Mux*42*	Mux42 Output Signal Detection 00 Level is passed after synchronization (default) 01 Posedge detection 10 Toggle detection 11 Level is passed after synchronization
12:13 Mux*41*	Mux41 Output Signal Detection 00 Level is passed after synchronization (default) 01 Posedge detection 10 Toggle detection 11 Level is passed after synchronization
14:15 Mux*40*	Mux40 Output Signal Detection 00 Level is passed after synchronization (default) 01 Posedge detection 10 Toggle detection 11 Level is passed after synchronization
16:17 Mux*39*	Mux39 Output Signal Detection 00 Level is passed after synchronization (default) 01 Posedge detection 10 Toggle detection 11 Level is passed after synchronization
18:19 Mux*38*	Mux38 Output Signal Detection 00 Level is passed after synchronization (default) 01 Posedge detection 10 Toggle detection 11 Level is passed after synchronization
20:21 Mux*37*	Mux37 Output Signal Detection 00 Level is passed after synchronization (default) 01 Posedge detection 10 Toggle detection 11 Level is passed after synchronization

Table 1138. ITLD2 register field descriptions (continued)

Field	Description
22:23 Mux*36*	Mux36 Output Signal Detection 00 Level is passed after synchronization (default) 01 Posedge detection 10 Toggle detection 11 Level is passed after synchronization
24:25 Mux*35*	Mux35 Output Signal Detection 00 Level is passed after synchronization (default) 01 Posedge detection 10 Toggle detection 11 Level is passed after synchronization
26:27 Mux*34*	Mux34 Output Signal Detection 00 Level is passed after synchronization (default) 01 Posedge detection 10 Toggle detection 11 Level is passed after synchronization
28:29 Mux*33*	Mux33 Output Signal Detection 00 Level is passed after synchronization (default) 01 Posedge detection 10 Toggle detection 11 Level is passed after synchronization
30:31 Mux*32*	Mux32 Output Signal Detection 00 Level is passed after synchronization (default) 01 Posedge detection 10 Toggle detection 11 Level is passed after synchronization

60.5.1.1.10 Input triggers level detection control 3 (ITLD3)

Only a rising edge is detected on the outputs of the first 32 level1 Muxes and so these values are not programmable. The last 32 level1 Muxes (Mux 32–63) trigger level is programmable in the Input Triggers Level Detection Control 2 and 3 registers. *Figure 1184* shows the format of the ITLD3 register.All of the Mux fields in the ITLD3 register have the same description.

Offset: 0x0C Access: Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	Mux63				Mux62				Mux61				Mux60			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	Mux55				Mux54				Mux53				Mux52			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
R W	Mux51				Mux50				Mux49				Mux48			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1184. Input triggers level detection control 3 (ITLD3)

The ITLD3 registers Mux fields are described in *Table 1139*.

Table 1139. ITLD3 register field descriptions

Field	Description
0:1 Mux*63*	Mux63 Output Signal Detection 00 Level is passed after synchronization (default) 01 Posedge detection 10 Toggle detection 11 Level is passed after synchronization
2:3 Mux*62*	Mux62 Output Signal Detection 00 Level is passed after synchronization (default) 01 Posedge detection 10 Toggle detection 11 Level is passed after synchronization
4:5 Mux*61*	Mux61 Output Signal Detection 00 Level is passed after synchronization (default) 01 Posedge detection 10 Toggle detection 11 Level is passed after synchronization
6:7 Mux*60*	Mux60 Output Signal Detection 00 Level is passed after synchronization (default) 01 Posedge detection 10 Toggle detection 11 Level is passed after synchronization
8:9 Mux*59*	Mux59 Output Signal Detection 00 Level is passed after synchronization (default) 01 Posedge detection 10 Toggle detection 11 Level is passed after synchronization
10:11 Mux*58*	Mux58 Output Signal Detection 00 Level is passed after synchronization (default) 01 Posedge detection 10 Toggle detection 11 Level is passed after synchronization
12:13 Mux*57*	Mux57 Output Signal Detection 00 Level is passed after synchronization (default) 01 Posedge detection 10 Toggle detection 11 Level is passed after synchronization
14:15 Mux*56*	Mux56 Output Signal Detection 00 Level is passed after synchronization (default) 01 Posedge detection 10 Toggle detection 11 Level is passed after synchronization
16:17 Mux*55*	Mux55 Output Signal Detection 00 Level is passed after synchronization (default) 01 Posedge detection 10 Toggle detection 11 Level is passed after synchronization

Table 1139. ITLD3 register field descriptions (continued)

Field	Description
18:19 Mux*54*	Mux54 Output Signal Detection 00 Level is passed after synchronization (default) 01 Posedge detection 10 Toggle detection 11 Level is passed after synchronization
20:21 Mux*53*	Mux53 Output Signal Detection 00 Level is passed after synchronization (default) 01 Posedge detection 10 Toggle detection 11 Level is passed after synchronization
22:23 Mux*52*	Mux52 Output Signal Detection 00 Level is passed after synchronization (default) 01 Posedge detection 10 Toggle detection 11 Level is passed after synchronization
24:25 Mux*51*	Mux51 Output Signal Detection 00 Level is passed after synchronization (default) 01 Posedge detection 10 Toggle detection 11 Level is passed after synchronization
26:27 Mux*50*	Mux50 Output Signal Detection 00 Level is passed after synchronization (default) 01 Posedge detection 10 Toggle detection 11 Level is passed after synchronization
28:29 Mux*49*	Mux49 Output Signal Detection 00 Level is passed after synchronization (default) 01 Posedge detection 10 Toggle detection 11 Level is passed after synchronization
30:31 Mux*48*	Mux48 Output Signal Detection 00 Level is passed after synchronization (default) 01 Posedge detection 10 Toggle detection 11 Level is passed after synchronization

60.5.1.1.11 CPU0 processor exception vector prefix (C0PEVP)

The SPU captures the exception vector from CPU0 when the exception enable signal is asserted. The exception vector decoding is defined in *Table 1163*. The SPU compares this captured vector with the value in the C0PEVP register. *Figure 1185* shows the format of the C0PEVP register.

Offset: 0x0D Access: Read/Write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	CPU0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1185. CPU0 processor exception vector prefix (C0PEVP)

The C0PEVP register is described in *Table 1140*.

Table 1140. C0PEVP register field descriptions

Field	Description
16:31 CPU0	CPU0 processor exception vector Program this field with one of the vector values from *Table 1163* to select the corresponding processor exception. This field must use the byte swapped values in the CnPEVP register CPUn field value column in *Table 1163*.

60.5.1.1.12 CPU0 interrupt priority selection (C0PIS)

Figure 1186 shows the format of the C0PIS register. The SPU compares the input CPR vector from CPU0 with the value programmed to the C0PIS register and generates a match signal if necessary. The typical use case would be for posedge detection on match (to detect when the priority level is set to the compare value) or high detection while match remains.

Offset: 0x12 Access: Read/Write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	CPU0							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1186. CPU0 interrupt priority selection (C0PIS)

The C0PIS register field is described in *Table 1141*.

Table 1141. C0PIS register field descriptions

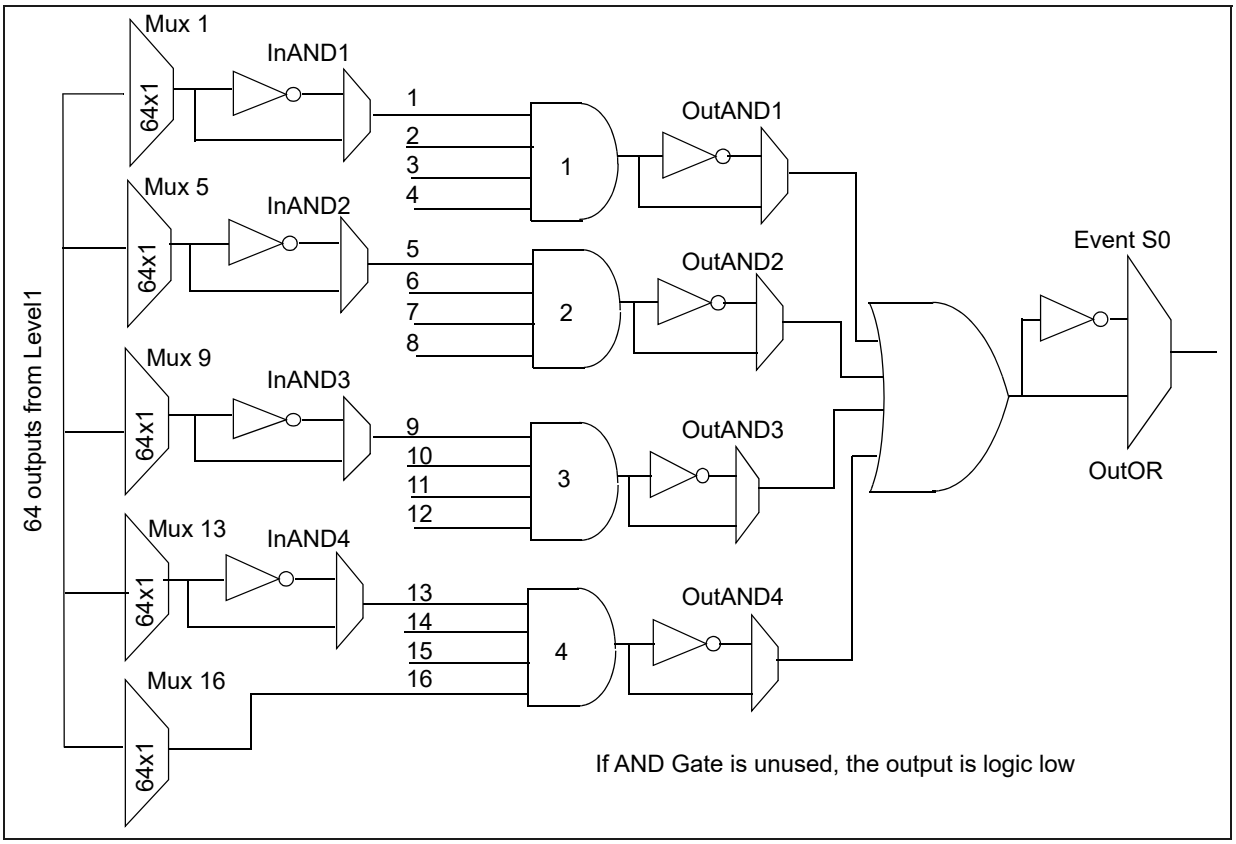
Field	Description
24:31 CPU0	CPU0 Priority Interrupt Selection

60.5.1.2 Level2 input Mux configurations registers

At level1, the set of input Mux control registers (described in *Section 60.5.1.1*) narrows down the large number of SPU watchpoints to a manageable set using sixty-four 8 × 1 multiplexers. At Level2, 16 inputs are selected for each SLU from the 64 level1 outputs using 16 64 × 1 multiplexers. Six bits per Mux select one input out of 64 (in fact 63) inputs. Each 32-bit register configures the AND gates of one state; therefore, eight registers are used to configure inputs for the 8 SLUs. All the 63 inputs are static for all the eight states. Any 16 inputs can be selected from the 63 input triggers for a state.

The detail selection is shown in the *Figure 1208* for State0. Similar logic is provided for States 1 to 7.

Figure 1187. Level2 Mux AND gates input pin routing for State0



Note: If any SLU AND gate is unused, then its output is logic low (1'b0). Thus, if all four inputs of a particular AND gate are at the default value 1'b1 (none of the inputs have been routed from the Level 2 mux to the inputs of AND gates), then the output of that AND gate is logic low. This provides protection to avoid the masking of other inputs to the OR gate.

60.5.1.2.1 Level2 Mux state n selection 0 (L2nSEL0)

Figure 1188 shows the format of the L2nSEL0 register where the state number, n = 0 to 7.

Offset: 0x17 +n*0x4 (n=0 to 7) Access: Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	FirstANDInput4						0	0	FirstANDInput3					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	FirstANDInput2						0	0	FirstANDInput1					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1188. Level2 Mux state n selection 0 (L2nSEL0)

The L2nSEL0 register fields are described in *Table 1142*.

Table 1142. L2nSEL0 register field descriptions

Field	Description
2:7 FirstANDInput4	First AND Gate Input 4 Selection 000000 No Input is selected (tie to 1'b1) 000001 Level1 MUX 0 output 000010 Level1 MUX 1 output 000011 Level1 MUX 2 output … 111110 Level1 MUX 61 output 111111 Level1 MUX 62 output
10:15 FirstANDInput3	First AND Gate Input 3 Selection 000000 No Input is selected (tie to 1'b1) 000001 Level1 MUX 0 output 000010 Level1 MUX 1 output 000011 Level1 MUX 2 output … 111110 Level1 MUX 61 output 111111 Level1 MUX 62 output

Table 1142. L2nSEL0 register field descriptions (continued)

Field	Description
18:23 FirstANDInput2	First AND Gate Input 2 Selection 000000 No Input is selected (tie to 1'b1) 000001 Level1 MUX 0 output 000010 Level1 MUX 1 output 000011 Level1 MUX 2 output … 111110 Level1 MUX 61 output 111111 Level1 MUX 62 output
26:31 FirstANDInput1	First AND Gate Input 1 Selection 000000 No Input is selected (tie to 1'b1) 000001 Level1 MUX 0 output 000010 Level1 MUX 1 output 000011 Level1 MUX 2 output … 111110 Level1 MUX 61 output 111111 Level1 MUX 62 output

60.5.1.2.2 Level2 Mux state n selection 1 (L2nSEL1)

Figure 1189 shows the format of the L2nSEL1 register where the state number, n = 0 to 7.

Offset: 0x18 +n*0x4 (n=0 to 7) Access: Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	SecondANDInput4						0	0	SecondANDInput3					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	SecondANDInput2						0	0	SecondANDInput1					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1189. Level2 Mux state n selection 1 (L2nSEL1)

The L2nSEL1 register fields are described in *Table 1143*.

Table 1143. L2nSEL1 register field descriptions

Field	Description
2:7 SecondANDInput4	Second AND Gate Input 4 Selection 000000 No Input is selected (tie to 1'b1) 000001 Level1 MUX 0 output 000010 Level1 MUX 1 output 000011 Level1 MUX 2 output … 111110 Level1 MUX 61 output 111111 Level1 MUX 62 output
10:15 SecondANDInput3	Second AND Gate Input 3 Selection 000000 No Input is selected (tie to 1'b1) 000001 Level1 MUX 0 output 000010 Level1 MUX 1 output 000011 Level1 MUX 2 output … 111110 Level1 MUX 61 output 111111 Level1 MUX 62 output
18:23 SecondANDInput2	Second AND Gate Input 2 Selection 000001 Level1 MUX 1 output 000011 Level1 MUX 2 output … 111110 Level1 MUX 61 output 111111 Level1 MUX 62 output
26:31 SecondANDInput1	Second AND Gate Input 1Selection 000000 No Input is selected (tie to 1'b1) 000001 Level1 MUX 0 output 000010 Level1 MUX 1 output 000011 Level1 MUX 2 output … 111110 Level1 MUX 61 output 111111 Level1 MUX 62 output

60.5.1.2.3 Level2 Mux state n selection 2 (L2nSEL2)

Figure 1190 shows the format of the L2nSEL2 register where the state number, n = 0 to 7.

Offset: 0x19 n*0x4 (n=0 to 7) Access: Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	ThirdANDInput4						0	0	ThirdANDInput3					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	ThirdANDInput2						0	0	ThirdANDInput1					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1190. Level2 Mux state n selection 2 (L2nSEL2)

The L2nSEL2 register fields are described in *Table 1144*.

Table 1144. L2nSEL2 register field descriptions

Field	Description
2:7 ThirdANDInput4	Third AND Gate Input 4 Selection 000000 No Input is selected (tie to 1'b1) 000001 Level1 MUX 0 output 000010 Level1 MUX 1 output 000011 Level1 MUX 2 output … 111110 Level1 MUX 61 output 111111 Level1 MUX 62 output
10:15 ThirdANDInput3	Third AND Gate Input 3 Selection 000000 No Input is selected (tie to 1'b1) 000001 Level1 MUX 0 output 000010 Level1 MUX 1 output 000011 Level1 MUX 2 output … 111110 Level1 MUX 61 output 111111 Level1 MUX 62 output
18:23 ThirdANDInput2	Third AND Gate Input 2 Selection 000000 No Input is selected (tie to 1'b1) 000001 Level1 MUX 0 output 000010 Level1 MUX 1 output 000011 Level1 MUX 2 output … 111110 Level1 MUX 61 output 111111 Level1 MUX 62 output
26:31 ThirdANDInput1	Third AND Gate Input 1 Selection 000000 No Input is selected (tie to 1'b1) 000001 Level1 MUX 0 output 000010 Level1 MUX 1 output 000011 Level1 MUX 2 output … 111110 Level1 MUX 61 output 111111 Level1 MUX 62 output

60.5.1.2.4 Level2 Mux state n selection 3 (L2nSEL3)

Figure 1191 shows the format of the L2nSEL3 register where the state number, n = 0 to 7.

Offset: 0x1A +n*0x4 (n=0 to 7) Access: Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	FourthANDInput4						0	0	FourthANDInput3					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	FourthANDInput2						0	0	FourthANDInput1					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1191. Level2 Mux state n selection 3 (L2nSEL3)

The L2nSEL3 register fields are described in *Table 1145*.

Table 1145. L2nSEL3 register field descriptions

Field	Description
2:7 FourthANDInput4	Fourth AND Gate Input 4 Selection 000000 No Input is selected (tie to 1'b1) 000001 Level1 MUX 0 output 000010 Level1 MUX 1 output 000011 Level1 MUX 2 output … 111110 Level1 MUX 61 output 111111 Level1 MUX 62 output
10:15 FourthANDInput3	Fourth AND Gate Input 3 Selection 000000 No Input is selected (tie to 1'b1) 000001 Level1 MUX 0 output 000010 Level1 MUX 1 output 000011 Level1 MUX 2 output … 111110 Level1 MUX 61 output 111111 Level1 MUX 62 output
18:23 FourthANDInput2	Fourth AND Gate Input 2 Selection 000000 No Input is selected (tie to 1'b1) 000001 Level1 MUX 0 output 000010 Level1 MUX 1 output 000011 Level1 MUX 2 output … 111110 Level1 MUX 61 output 111111 Level1 MUX 62 output
26:31 FourthANDInput1	Fourth AND Gate Input 1 Selection 000000 No Input is selected (tie to 1'b1) 000001 Level1 MUX 0 output 000010 Level1 MUX 1 output 000011 Level1 MUX 2 output … 111110 Level1 MUX 61 output 111111 Level1 MUX 62 output

60.5.1.3 Sequence unit registers

60.5.1.3.1 Input/output Inversion control state n (IOICn)

An input control register is provided for each event generated by the SLU. This register is programmed to specify how the input signals to each SLU are combined to generate the resulting event for that particular sequence. In total, there are 16 inputs for each state and only one (the first) of the inputs can be used with or without inversion. The input can be left unused if not required for that state by selecting 000000 (No input selected option) in the L2nSEL registers. The output of each AND gate has the optional inversion control as well. The output can be routed directly or with inversion from each AND gate to the input of OR gate. *Figure 1192* shows the format of the IOICn register where the state number, n = 0 to 7.

Offset: 0x37 n*0x1 (n=0 to 7) Access: Read/Write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	OutOR	OutAND4	OutAND3	OutAND2	OutAND1	InAND4	InAND3	InAND2	InAND1
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1192. Input/output Inversion control state n (IOICn)

The IOICn register fields are described in *Table 1146*.

Table 1146. IOICn register field descriptions

Field	Description
23 OutOR	Output OR GATE Control 0 Output is used directly (without inversion) for this event 1 Output is inverted for this event
24 OutAND4	Output of AND Gate 4 Control 0 Output is used directly (without inversion) for this event 1 Output is inverted for this event
25 OutAND3	Output AND Gate 3 Control 0 Output is used directly (without inversion) for this event 1 Output is inverted for this event
26 OutAND2	Output AND Gate 2 Control 0 Output is used directly (without inversion) for this event 1 Output is inverted for this event
27 OutAND1	Output AND Gate 1 Control 0 Output is used directly (without inversion) for this event 1 Output is inverted for this event

Table 1146. IOICn register field descriptions (continued)

Field	Description
28 InAND4	Input 1 AND Gate 4 Control 0 Input 1 is used directly (without inversion) for this event 1 Input 1 is inverted for this event
29 InAND3	Input 1 AND Gate 3 Control 0 Input 1 is used directly (without inversion) for this event 1 Input 1 is inverted for this event
30 InAND2	Input 1 AND Gate 2 Control 0 Input 1 is used directly (without inversion) for this event 1 Input 1 is inverted for this event
31 InAND1	Input 1 AND Gate 1 Control 0 Input 1 is used directly (without inversion) for this event 1 Input 1 is inverted for this event

60.5.1.3.2 Sequence control (SCTRL)

This configuration register is provided to enable the sequence and to select the states that are part of a particular sequence. There can be at most four active sequences. Each sequence can have one state or a maximum of eight states. However, each state can only be used in one unique sequence. In other words, if one state has been selected for a sequence, then this state cannot be used for any other defined sequence. Each sequence has a start state and end state. *Figure 1193* shows the format of the SCTRL register.

Offset: 0x3F Access: Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	S4ST2			S4ST1			S4ST0	0	S3ST2			S3ST1			S3ST0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	S2ST2			S2ST1			S2ST0	0	S1ST2			S1ST1			S1ST0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1193. Sequence control (SCTRL)

The SCTRL register fields are described in *Table 1147*.

Table 1147. SCTRL register field descriptions

Field	Description
1:3 S4ST2	Sequence 4 End State 000 State 0 is the Stop state for sequence 4 001 State 1 is the Stop state for sequence 4 010 State 2 is the Stop state for sequence 4 011 State 3 is the Stop state for sequence 4 100 State 4 is the Stop state for sequence 4 101 State 5 is the Stop state for sequence 4 110 State 6 is the Stop state for sequence 4 111 State 7 is the Stop state for sequence 4
4:6 S4ST1	Sequence 4 Start State(1) 000 State 0 is the Start state for sequence 4 001 State 1 is the Start state for sequence 4 010 State 2 is the Start state for sequence 4 011 State 3 is the Start state for sequence 4 100 State 4 is the Start state for sequence 4 101 State 5 is the Start state for sequence 4 110 State 6 is the Start state for sequence 4 111 State 7 is the Start state for sequence 4
7 S4ST0	Sequence 4 Enable/Disable Once enabled, this bit is disabled at the end of the Sequence 4. 0 Sequence 4 is disabled 1 Sequence 4 is enabled
9:11 S3ST2	Sequence 3 End State 000 State 0 is the Stop state for sequence 3 001 State 1 is the Stop state for sequence 3 010 State 2 is the Stop state for sequence 3 011 State 3 is the Stop state for sequence 3 100 State 4 is the Stop state for sequence 3 101 State 5 is the Stop state for sequence 3 110 State 6 is the Stop state for sequence 3 111 State 7 is the Stop state for sequence 3
12:14 S3ST1	Sequence 3 Start State(1) 000 State 0 is the Start state for sequence 3 001 State 1 is the Start state for sequence 3 010 State 2 is the Start state for sequence 3 011 State 3 is the Start state for sequence 3 100 State 4 is the Start state for sequence 3 101 State 5 is the Start state for sequence 3 110 State 6 is the Start state for sequence 3 111 State 7 is the Start state for sequence 3
15 S3ST0	Sequence 3 Enable/Disable Once enabled, this bit is disabled at the end of the Sequence 3. 0 Sequence 3 is disabled 1 Sequence 3 is enabled

Table 1147. SCTRL register field descriptions (continued)

Field	Description
17:19 S2ST2	Sequence 2 End State 000 State 0 is the Stop state for sequence 2 001 State 1 is the Stop state for sequence 2 010 State 2 is the Stop state for sequence 2 011 State 3 is the Stop state for sequence 2 100 State 4 is the Stop state for sequence 2 101 State 5 is the Stop state for sequence 2 110 State 6 is the Stop state for sequence 2 111 State 7 is the Stop state for sequence 2
20:22 S2ST1	Sequence 2 Start State(1) 000 State 0 is the Start state for sequence 2 001 State 1 is the Start state for sequence 2 010 State 2 is the Start state for sequence 2 011 State 3 is the Start state for sequence 2 100 State 4 is the Start state for sequence 2 101 State 5 is the Start state for sequence 2 110 State 6 is the Start state for sequence 2 111 State 7 is the Start state for sequence 2
23 S2ST0	Sequence2 Enable/Disable Once enabled, this bit is disabled at the end of the Sequence 2. 0 Sequence 2 is disabled 1 Sequence 2 is enabled
25:27 S1ST2	Sequence 1 End State 000 State 0 is the Stop state for sequence 1 001 State 1 is the Stop state for sequence 1 010 State 2 is the Stop state for sequence 1 011 State 3 is the Stop state for sequence 1 100 State 4 is the Stop state for sequence 1 101 State 5 is the Stop state for sequence 1 110 State 6 is the Stop state for sequence 1 111 State 7 is the Stop state for sequence 1
28:30 S1ST1	Sequence 1 Start State 000 State 0 (Always)
31 S1ST0	Sequence1 Enable/Disable Once enabled, this bit is disabled at the end of the Sequence 1. 0 Sequence 1 is disabled 1 Sequence 1 is enabled

1. Ideally, the start state for a sequence should be the state immediately after the stop state for the previous sequence. For example, if sequence 1 starts at state 0 and ends at state3, sequence 2 should start at state4.

60.5.1.3.3 State n goto control (SnGC)

SLU states are linked to each other based on if-then-else conditions. There are two conditions: true and false. A SLU state can remain in the same state or jump to any other state. The state goto control register is programmed to specify the GoTo a particular state from current state.

Figure 1194 shows the format of the SnGC register where the state number, n = 0 to 7.

Offset: 0x40 +n*0x1 (n= 0 to 7) Access: Read/Write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	NF			FE	NT			TE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1194. State n goto control (SnGC)

The SnGC register fields are described in *Table 1148*.

Table 1148. SnGC register field descriptions

Field	Description
24:26 NF	Next State on False Condition Defines the Goto next state in case of false condition from the present state. States may remain in the same state on a false condition if the same state is chosen as the next state. 000 Goto State 0 001 Goto State 1 010 Goto State 2 011 Goto State 3 100 Goto State 4 101 Goto State 5 110 Goto State 6 111 Goto State 7
27 FE	GOTO Control of State Under False Condition Defines the Goto control of a state under the false condition. If this bit is enabled, then the state moves to the next state based on bits[7:5]. If disabled, then this is the is end of a sequence when the sequence comes to this state. Actions associated with states under false condition are executed. Disabling this bit end the sequence under a false condition. 0 Disable Goto control 1 Enable Goto control

Table 1148. SnGC register field descriptions (continued)

Field	Description
28:30 NT	Next State on True Condition Defines the Goto next state in case of true condition from the present state. States may remain in the same state on a true condition if the same state is chosen as the next state. 000 Goto State 0 001 Goto State 1 010 Goto State 2 011 Goto State 3 100 Goto State 4 101 Goto State 5 110 Goto State 6 111 Goto State 7
31 TE	GoTo Control of State Under True Condition Defines the Goto control of a state under the true condition. If this bit is enabled, then the State moves to the next state based on bits[3:1]. If disabled, then this is the end of a sequence when the sequence comes to this state. Actions associated with states under true condition are executed. Disabling this bit ends the sequence under a true condition. 0 Disable Goto control 1 Enable Goto control

60.5.1.4 SLU status register (SS)

The SLU status register indicates the states that have been hit/encountered in any of the active sequences. *Figure 1195* shows the format of the SS register.

Offset: 0x48 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	IM0	0	0	EM0	SAS4			SAS3
W									w1c			w1c	w1c			w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	SAS3		SAS2			SAS1			SH7	SH6	SH5	SH4	SH3	SH2	SH1	SH0
W	w1c		w1c			w1c			w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1195. SLU status register (SS)

The SS register fields are described in *Table 1149*.

Table 1149. SS register field descriptions

Field	Description
8 IM0	Interrupt Match 0 This bit indicates the status of the match value on interrupt priority register of CPU0. 0 No match 1 Match occurred
11 EM0	Exception Match 0 This bit indicates the status of the processor exception vector match value of CPU0. 0 No match 1 Match occurred
12:14 SAS4	Sequence 4 Active State Two consecutive read cycles are required to know the active state.These bits capture the active state of sequence 4 at the clock when these bits are read. The active state might be different in the previous clock and the next clock. 000 State0 is the active state 001 State1 is the active state 010 State2 is the active state 011 State3 is the active state 100 State4 is the active state 101 State5 is the active state 110 State6 is the active state 111 State7 is the active state
15:17 SAS3	Sequence 3 Active State Two consecutive read cycles are required to know the active state. These bits capture the active state of sequence 3 at the clock when these bits are read. The active state might be different in the previous clock and the next clock. 000 State0 is the active state 001 State1 is the active state 010 State2 is the active state 011 State3 is the active state 100 State4 is the active state 101 state5 is the active state 110 State6 is the active state 111 State7 is the active state
18:20 SAS2	Sequence 2 Active State Two consecutive read cycles are required to know the active state.These bits capture the active state of sequence 2 at the clock when these bits are read. The active state might be different in the previous clock and the next clock. 000 State0 is the active state 001 State1 is the active state 010 State2 is the active state 011 State3 is the active state 100 State4 is the active state 101 State5 is the active state 110 State6 is the active state 111 State7 is the active state

Table 1149. SS register field descriptions (continued)

Field	Description
21:23 SAS1	Sequence 1 Active State Two consecutive read cycles are required to know the active state. These bits capture the active state of sequence 1 at the clock when these bits are read. The active state might be different in the previous clock and the next clock. 000 State0 is the active state 001 State1 is the active state 010 State2 is the active state 011 State3 is the active state 100 State4 is the active state 101 State5 is the active state 110 State6 is the active state 111 State7 is the active state
24 SH7	State Hit 7 This bit is set when State7 is hit/encountered in any of the active sequences 0 State not hit 1 State hit
25 SH6	State Hit 6 This bit is set when State6 is hit/encountered in any of the active sequences 0 State not hit 1 State hit
26 SH5	State Hit 5 This bit is set when State5 is hit/encountered in any of the active sequences 0 State not hit 1 State hit
27 SH4	State Hit 4 This bit is set when State4 is hit/encountered in any of the active sequences 0 State not hit 1 State hit
28 SH3	State Hit 3 This bit is set when State3 is hit/encountered in any of the active sequences 0 State not hit 1 State hit
29 SH2	State Hit 2 This bit is set when State2 is hit/encountered in any of the active sequences 0 State not hit 1 State hit
30 SH1	State Hit 1 This bit is set when State1 is hit/encountered in any of the active sequences 0 State not hit 1 State hit
31 SH0	State Hit 0 This bit is set when State0 is hit/encountered in any of the active sequences 0 State not hit 1 State hit

60.5.1.5 SPU enable register (SE)

The SPU enable/disable register is used to enable and disable the SPU. If disabled, all inputs are masked. *Figure 1196* shows the format of the SE register.

Offset: 0x49 Access: Read/Write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	(1)	SED
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. This bit is writable.

Figure 1196. SPU enable register (SE)

The SE register fields are described in *Table 1150*.

Table 1150. SE register field descriptions

Field	Description
31 SED	SPU Enable/Disable This bit should be set after programming all of the SPU configuration registers. 0 SPU is disabled 1 SPU is enabled

60.5.1.6 SLU action unit registers

The action unit of the SPU is responsible for generating the action triggers supported by various actions, listed in *Section 60.2*. The action triggers are mapped to the events by programming the action registers. Each event can be programmed to trigger interrupts, watchpoints, trace signals, counter stop/reset/increment and many other general actions. *Table 1151* shows the bit mapping for all the possible actions divided in three groups: client based, counter based, system related.

Table 1151. SPU action decoder

Group bit [11:10]		Client/counter bit [9:5]		Action bit [4:1]			Enable bit [0]
ID	Name	ID	Name	ID	Name	Comment	
00	Client group	00000	CPU0	0000	No action	—	0=Stop 1=Start
		00001	Reserved	—	DTM_gen0_trace—	Valid for Core, AHB and trace group actions	
		00010	Reserved	—	OTM_gen1_trace—		
		00011–00111	Reserved	0011	PTM_gen2_trace—		
		01000	Reserved	0100	WTM_gen3_trace—		
		01001–01111	Reserved	0101	PMC1—	Valid for the cores as PMC—	For PMCn 0= Stop 1= Start
		10000	NXMC1	0110	PMC2—		
		10001	NXMC2	0111	PMC3—		
		10010–10111	Reserved	1000	PMC4—		
		11000	Reserved	1001	—	—	0=Stop 1=Start
		11001–11110	Reserved	1010	—		
				1011	—		
				1100	—		
		11111	Trace group(1)	1101–1111	Reserved	Valid for CPU core and AHB	
01	Counter group	00000	Counter0	0000	No action	—	Don’t care
		00001	Counter1	0001	Start timer/Incr counter	Valid for all counters/timers	
		00010	Counter2	0010	Stop timer		
		…	…	0011	Reset counter		
		01111	Counter15	0100	Capture		
		10000–11111	Reserved	0101–1111	Reserved		

Table 1151. SPU action decoder (continued)

Group bit [11:10]		Client/counter bit [9:5]		Action bit [4:1]			Enable bit [0]
ID	Name	ID	Name	ID	Name	Comment	
10	System group	00000	System action	0000	No action	—	Don't care
		00001–11111	Reserved	0001	Halt1_dci_evti	—	
				0010	Halt2_dci_evti	—	
				0011	—	—	
				0100	—	—	
				0101	—	—	
				0110	—	—	
				0111	Evto1	—	
				1000	Evto2	—	
				1001	Reserved	—	
				1010	Core INTR	—	
				1011–1111	Reserved	—	—
11	Reserved						

1. The valid actions for Trace group are OTM trace, PTM trace, DTM trace or WTM trace. No other actions can be selected with Trace group. The group can be defined using the Trace group configuration registers.

60.5.1.6.1 State n true action 0 (SnTA0)

The SPU can be programmed to enable up to four different actions against a true condition of every state. All the possible actions and their encoding are listed in *Table 1151*. A total of 12 bits are needed to code every action. The first two actions for true condition are defined in SnTA0 and the next two actions are defined in SnTA1.

Figure 1197 shows the format of the SnTA0 register where the state number, n = 0 to 7.

Offset: 0x01 + n*0x2 (n=0 to 7) Access: Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	ACTION2											
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	ACTION1											
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1197. State n true action 0 (SnTA0)

The SnTA0 register fields are described in *Table 1152*.

Table 1152. SnTA0 register field descriptions

Field	Description
4:15 ACTION2	Action 2 to be performed for State n true condition The decoding of these bits is defined in *Table 1151*.
20:31 ACTION1	Action 1 to be performed for State n true condition The decoding of these bits is defined in *Table 1151*.

60.5.1.6.2 State n true action 1 (SnTA1)

The SPU can be programmed to enable up to four different actions against a true condition of every state. All the possible actions and their encoding are listed in *Table 1151*. A total of 12 bits are needed to code every action. The first two actions for true condition are defined in SnTA0 and the next two actions are defined in SnTA1. *Figure 1198* shows the format of the SnTA1 register where the state number, n = 0 to 7.

Offset: 0x02 + n*0x2 (n= 0 to 7) Access: Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	ACTION4											
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	ACTION3											
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1198. State n true action 1 (SnTA1)

The SnTA1 register fields are described in *Table 1153*.

Table 1153. SnTA1 register field descriptions

Field	Description
4:15 ACTION4	Action 4 to be performed for State n true condition The decoding of these bits is defined in *Table 1151*.
20:31 ACTION3	Action 3 to be performed for State n true condition The decoding of these bits is defined in *Table 1151*.

60.5.1.6.3 State n false action 0 (SnFA0)

The SPU can be programmed to enable up to four different actions against a false condition of every state. All the possible actions and their encoding are listed in *Table 1151*. A total of 12 bits are needed to code every action. The first two actions for a false condition are defined in SnFA0 and the next two actions are defined in SnFA1. *Figure 1199* shows the format of the SnFA0 register where the state number, n = 0 to 7.

Offset: 0x11 + n*0x2 (n=0 to 7) Access: Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	ACTION2											
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	ACTION1											
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1199. State n false action 0 (SnFA0)

The SnFA0 register fields are described in *Table 1154*.

Table 1154. SnFA0 register field descriptions

Field	Description
4:15 ACTION2	Action 2 to be performed for State n false condition The decoding of these bits is defined in *Table 1151*.
20:31 ACTION1	Action 1 to be performed for State n false condition The decoding of these bits is defined in *Table 1151*.

60.5.1.6.4 State n false action 1 (SnFA1)

The SPU can be programmed to enable up to four different actions against a false condition of every state. All the possible actions and their encoding are listed in *Table 1151*. A total of 12 bits are needed to code every action. The first two actions for false condition are defined in SnFA0 and the next two actions are defined in SnFA1. *Figure 1200* shows the format of the SnFA1 register where the state number, n = 0 to 7.

Offset: 0x12 + n*0x2 (n=0 to 7) Access: Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	ACTION4											
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	ACTION3											
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1200. State n false action 1 (SnFA1)

The SnFA1 register fields are described in *Table 1155*.

Table 1155. SnFA1 register field descriptions

Field	Description
5:15 ACTION4	Action 4 to be performed for State n false condition The decoding of these bits is defined in *Table 1151*.
20:31 ACTION3	Action 3 to be performed for State n false condition The decoding of these bits is defined in *Table 1151*.

60.5.1.6.5 Trace group configuration registers (DTMTGC, PTMTGC, OTMTGC, WTMTGC)

There are four action trace groups and a configuration register for each trace group:

- DTM trace: clients are configured with the DTM trace group configuration (DTMTGC) register
- PTM trace: clients are configured with the PTM trace group configuration (PTMTGC) register
- OTM trace: clients are configured with the OTM trace group configuration (OTMTGC) register
- WTM trace: clients are configured with the WTM trace group configuration (WTMTGC) register

Each of these groups are specific to a trace. Any number of clients can be enabled for one trace using the action trace group.

Figure 1201 shows the format of the trace group configuration registers.

Offset: 0x21 (DTMTGC), 0x22 (PTMTGC), 0x23 (OTMTGC), 0x24 (WTMTGC)

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	AHB2	AHB1
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	CPU0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1201. Trace group configuration register format (DTMTGC, PTMTGC, OTMTGC, WTMTGC)

The trace group configuration register fields are described in *Table 1156*.

Table 1156. DTMTGC, PTMTGC, OTMTGC, WTMTGC field descriptions

Field	Description
14 AHB2	0 Disable trace for AHB2 1 Enable trace for AHB2
15 AHB1	0 Disable trace for AHB1 1 Enable trace for AHB1
31 CPU0	0 Disable trace for CPU0 1 Enable trace for CPU0

60.5.1.6.6 Interrupt status (INTS)

The status of the external interrupt request to the processor core through the JTAGM is registered in the Interrupt status (INTS) register. This status bit is cleared when the acknowledge is received from the JTAGM. *Figure 1202* shows the format of the INTS register.

Offset: 0x25 Access: Read/Write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	INT
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1202. Interrupt status (INTS)

The INTS register fields are described in *Table 1157*.

Table 1157. INTS register field descriptions

Field	Description
31 INT	1 Interrupt status bit is cleared from JTAGM

60.5.1.7 Counters control n registers (CCTRLn)

There are 16 configurable counters/timers in the SPU. Configuration control of these counters is defined in the counter control registers. *Figure 1203* shows the format of the CCTRLn registers where the counter number, n = 0 to 15.

Offset: 0x26 + n*0x1 (n = 0 to 15) Access: Read/Write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	SW_RESET	PRESCALING			0	COUNT_INCR	MODE	ENABLE
W								0								
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1203. Counters control n registers (CCTRLn)

The CCTRLn register fields are described in *Table 1158*.

Table 1158. CCTRLn register field descriptions

Field	Description
23	Reserved. This bit is for internal use only and must be set to 0.
24 SW_RESET	Reset Counter can be reset by software. 0 Software reset is disabled 1 Software reset
25:27 PRESCALING	Prescaling Divides the timer clock by a defined value. 000 No scaling, divide by 1 001 Divide by 4 010 Divide by 16 011 Divide by 32 100 Reserved 101 Reserved 110 Reserved 111 Reserved
29 COUNT_INCR	Count Increment 0 Increment the counter based on the actions generated from the sequences 1 Increment the counter based on the PMC input. The mapping of the PMC inputs with the counters is shown in *Figure 1215*.
30 MODE	Mode Select Select counter or timer operation. 0 Counter 1 Timer
31 ENABLE	Enable 0 Counter disabled 1 Counter enabled

60.5.1.8 Counters compare n registers (CCMPn)

A 32-bit comparator value can be programmed for each of the counters. This value is compared with the corresponding counter/timer to generate an event for the SPU. *Figure 1204* shows the format of the CCMPn registers where the counter number, n = 0 to 15.

Offset: 0x36 + n*0x1 (n = 0 to 15) Access: Read/Write

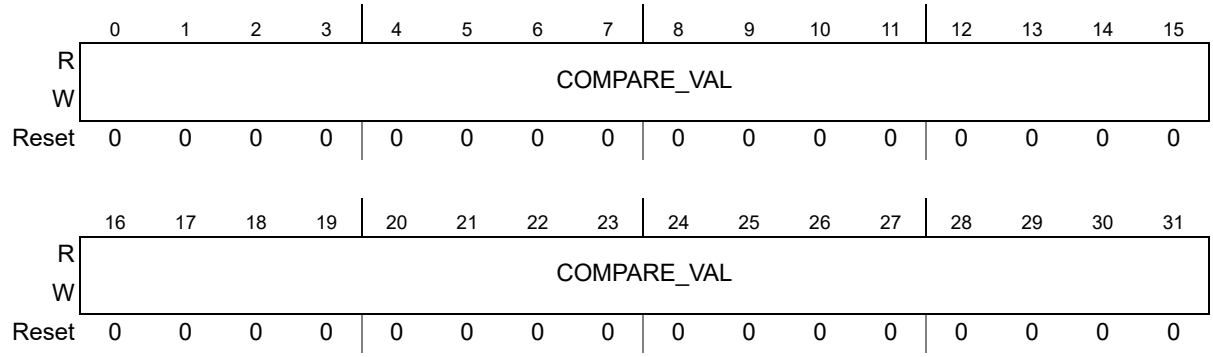


Figure 1204. Counters compare n registers (CCMPn)

The CCMPn register fields are described in *Table 1159*.

Table 1159. CCMPn register field descriptions

Field	Description
0:31 COMPARE_VAL	Compare value This value is compared with the corresponding counter/timer value to generate an event for the SPU.

60.5.1.9 Status registers

60.5.1.9.1 Counter compare status (CCOMS)

The counter compare status register shows the status of the comparisons for all the counters. The bits in this register can be cleared by writing a 1 to them. *Figure 1205* shows the format of the CCOMS register. All of the CC fields in the CCOMS register have the same description as shown in *Table 1160*.

Note: Counter compare values and other settings are not recommended to change dynamically, so w1c on CCOMS should be performed in either condition:

- *Counter is in disabled state (ENABLE bit of CCTRLn set to 0)*
- *SW-Reset bit (SW_RESET) is set*

Offset: 0x46 Access: Read/Write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	CC15	CC14	CC13	CC12	CC11	CC10	CC9	CC8	CC7	CC6	CC5	CC4	CC3	CC2	CC1	CC0
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1205. Counter compare status (CCOMS)

The CCOMS register fields are described in *Table 1160*.

Table 1160. CCOMS register field descriptions

Field	Description
16 CC15	Counter 15 Compare 0 Counter status is not true 1 Counter status is true
17 CC14	Counter 14 Compare 0 Counter status is not true 1 Counter status is true
18 CC13	Counter 13 Compare 0 Counter status is not true 1 Counter status is true
19 CC12	Counter 12 Compare 0 Counter status is not true 1 Counter status is true
20 CC11	Counter 11 Compare 0 Counter status is not true 1 Counter status is true
21 CC10	Counter 10 Compare 0 Counter status is not true 1 Counter status is true
22 CC9	Counter 9 Compare 0 Counter status is not true 1 Counter status is true
23 CC8	Counter 8 Compare 0 Counter status is not true 1 Counter status is true
24 CC7	Counter 7 Compare 0 Counter status is not true 1 Counter status is true

Table 1160. CCOMS register field descriptions (continued)

Field	Description
25 CC6	Counter 6 Compare 0 Counter status is not true 1 Counter status is true
26 CC5	Counter 5 Compare 0 Counter status is not true 1 Counter status is true
27 CC4	Counter 4 Compare 0 Counter status is not true 1 Counter status is true
28 CC3	Counter 3 Compare 0 Counter status is not true 1 Counter status is true
29 CC2	Counter 2 Compare 0 Counter status is not true 1 Counter status is true
30 CC1	Counter 1 Compare 0 Counter status is not true 1 Counter status is true
31 CC0	Counter 0 Compare 0 Counter status is not true 1 Counter status is true

60.5.1.9.2 Counter overflow status (COS)

Whenever a counter overflow occurs, the corresponding counter overflow status register bit is set. The bits in this register can be cleared by writing a 1 to them. *Figure 1206* shows the format of the COS register. All of the CO fields in the COS register have the same description as shown in *Table 1161*.

Offset: 0x47 Access: Read/Write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	CO15	CO14	CO13	CO12	CO11	CO10	CO9	CO8	CO7	CO6	CO5	CO4	CO3	CO2	CO1	CO0
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1206. Counter overflow status (COS)

The COS register fields are described in *Table 1161*.

Table 1161. COS register field descriptions

Field	Description
16 CO15	Counter 15 Overflow 0 Counter did not overflow 1 Counter overflow occurred
17 CO14	Counter 14 Overflow 0 Counter did not overflow 1 Counter overflow occurred
18 CO13	Counter 13 Overflow 0 Counter did not overflow 1 Counter overflow occurred
19 CO12	Counter 12 Overflow 0 Counter did not overflow 1 Counter overflow occurred
20 CO11	Counter 11 Overflow 0 Counter did not overflow 1 Counter overflow occurred
21 CO10	Counter10 Overflow 0 Counter did not overflow 1 Counter overflow occurred
22 CO9	Counter 9 Overflow 0 Counter did not overflow 1 Counter overflow occurred
23 CO8	Counter 8 Overflow 0 Counter did not overflow 1 Counter overflow occurred
24 CO7	Counter 7 Overflow 0 Counter did not overflow 1 Counter overflow occurred
25 CO6	Counter 6 Overflow 0 Counter did not overflow 1 Counter overflow occurred
26 CO5	Counter 5 Overflow 0 Counter did not overflow 1 Counter overflow occurred
27 CO4	Counter 4 Overflow 0 Counter did not overflow 1 Counter overflow occurred
28 CO3	Counter 3 Overflow 0 Counter did not overflow 1 Counter overflow occurred
29 CO2	Counter 2 Overflow 0 Counter did not overflow 1 Counter overflow occurred

Table 1161. COS register field descriptions (continued)

Field	Description
30 CO1	Counter 1 Overflow 0 Counter did not overflow 1 Counter overflow occurred
31 CO0	Counter 0 Overflow 0 Counter did not overflow 1 Counter overflow occurred

60.5.1.9.3 Counter capture status (CCAPS)

Whenever a counter value is captured, the corresponding capture status register bit for that counter is set. If the counter capture value causes a FIFO overflow, the captured value is lost and the status bit is not set. The bits in this register can be cleared by writing a 1 to them. *Figure 1207* shows the format of the CCAPS register.

Offset: 0x48 Access: Read/Write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	CC15	CC14	CC13	CC12	CC11	CC10	CC9	CC8	CC7	CC6	CC5	CC4	CC3	CC2	CC1	CC0
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1207. Counter capture status (CCAPS)

The CCAPS register fields are described in *Table 1162*.

Table 1162. CCAPS register field descriptions

Field	Description
16 CC15	Counter 15 Captured 0 Counter not captured 1 Counter captured
17 CC14	Counter 14 Captured 0 Counter not captured 1 Counter captured
18 CC13	Counter13 Captured 0 Counter not captured 1 Counter captured
19 CC12	Counter 12 Captured 0 Counter not captured 1 Counter captured

Table 1162. CCAPS register field descriptions (continued)

Field	Description
20 CC11	Counter 11 Captured 0 Counter not captured 1 Counter captured
21 CC10	Counter 10 Captured 0 Counter not captured 1 Counter captured
22 CC9	Counter 9 Captured 0 Counter not captured 1 Counter captured
23 CC8	Counter 8 Captured 0 Counter not captured 1 Counter captured
24 CC7	Counter 7 Captured 0 Counter not captured 1 Counter captured
25 CC6	Counter 6 Captured 0 Counter not captured 1 Counter captured
26 CC5	Counter 5 Captured 0 Counter not captured 1 Counter captured
27 CC4	Counter 4 Captured 0 Counter not captured 1 Counter captured
28 CC3	Counter 3 Captured 0 Counter not captured 1 Counter captured
29 CC2	Counter 2 Captured 0 Counter not captured 1 Counter captured
30 CC1	Counter 1 Captured 0 Counter not captured 1 Counter captured
31 CC0	Counter 0 Captured 0 Counter not captured 1 Counter captured

60.6 Functional description

60.6.1 Input Mux unit

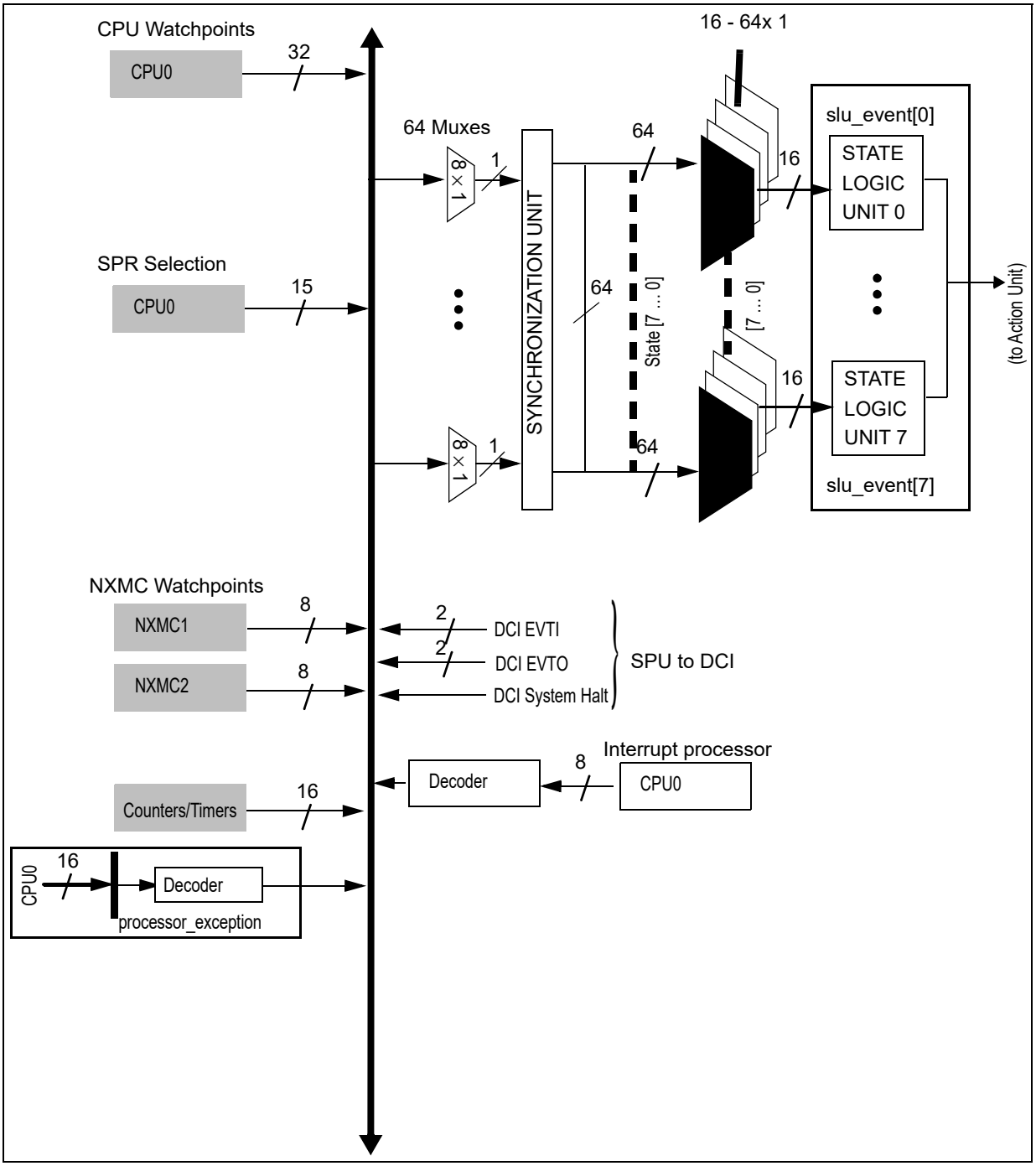
The SPU receives numerous inputs in the form of complex triggers from various clients to create a predefined sequence. One state logic can accommodate 16 inputs at one point of time. Therefore, an input Muxing unit becomes essential for reducing the debug and performance events to a manageable set of 16 inputs for each SLU. In addition to client

input signals, there are certain performance events which are fed directly to the different counters inside the SPU.

The selection of 16 inputs for each state logic is done at two level of muxing both of which are shown in *Figure 1208*.

- Level 1 Muxing: At this level, there are 209 inputs. Out of these possible inputs, 63 inputs are selected. This selection is done using an 8:1 Mux. Therefore, sixty-four 8×1 muxes are required. The set of 63 watchpoints are passed to the second level of muxing. The selection mechanism is shown in *Figure 1208*.
- Level 2 Muxing: At this level, 16 inputs are selected out of the 63 possible inputs for each SLU. These 63 inputs consist of multiple set of 8 watchpoints from different clients, processor exception signals, interrupt from clients, and event IN/OUT signals. In addition, certain events from counters are used as feedback in the present state logic of each state.
  At level two there are 16 banks of 64:1 muxes for the selection of 16 inputs for one state logic. The inputs to these Muxes are expected to be somewhat unique. In general, care should be take to choose signal assignments that allow for key use cases to be achieved without encountering concurrency restrictions. Since there are eight such states, 16×8 banks of 64:1 mux are required. The selection mechanism is shown in *Figure 1208*.

Figure 1208. Input Mux unit block diagram



60.6.2 Processor exception vector encoding

Active-high input signals from the CPU indicate the particular exception being processed when the CPU exception enable signal is asserted. These signals can be used for debug triggering mechanisms. The SPU captures the CPU exception vector whenever the CPU exception enable signal is asserted and the SPU uses the exception vector as an input

condition as shown in *Table 1163*. The CPU exception vector values from the CPU must be byte swapped before they are programmed to the CPUn field of the CnPEVP (n = 0) registers as shown in *Table 1163*.

Table 1163. Processor exception processing conditions

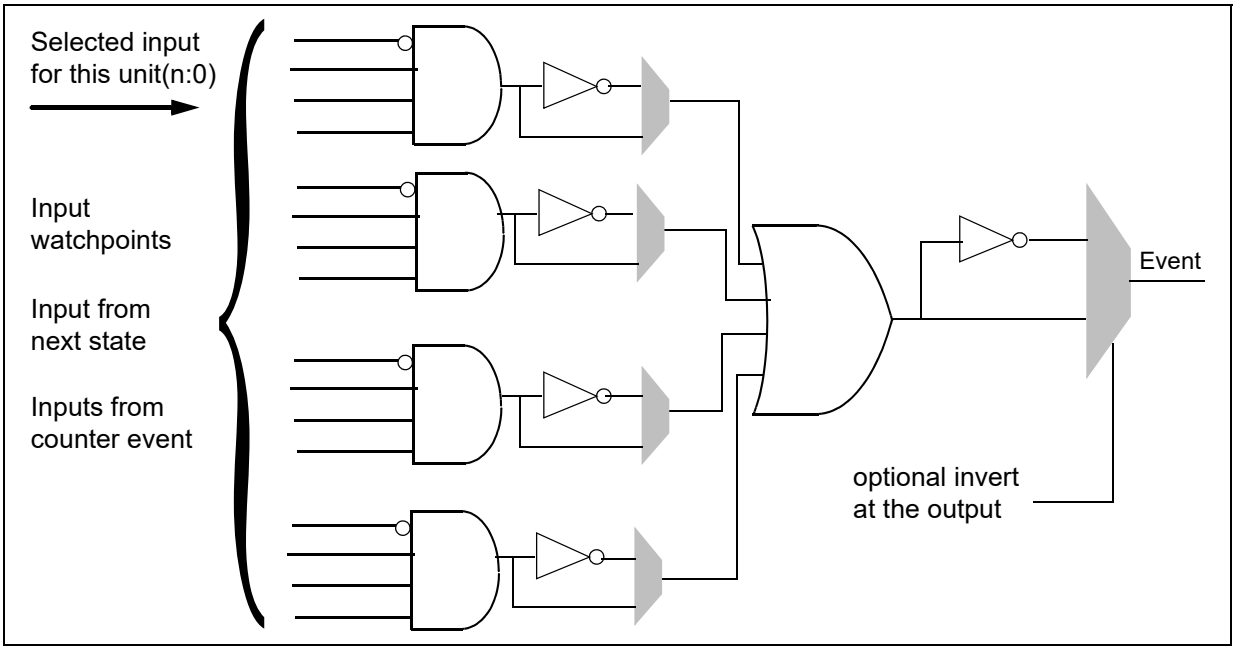
CPU exception vector	CnPEVP register CPUn field value	Exception condition
0x0000	0x0000	Critical Input: autovectored
0x0010	0x1000	Machine check
0x0020	0x2000	Data storage
0x0030	0x3000	Instruction storage
0x0040	0x4000	External input: autovectored
0x0050	0x5000	Alignment
0x0060	0x6000	Program trap
0x0061	0x6100	Program illegal
0x0062	0x6200	Program privileged
0x0070	0x7000	Performance monitor
0x0080	0x8000	System call
0x0090	0x9000	Debug
0x00A0	0xA000	EFPU data exception
0x00B0	0xB000	EFPU round exception
0xVVVV3, low two bits = 01	0xVVVV3, high two bits = 01	External input: vectored
0xVVVV3, low two bits = 01	0xVVVV3, high two bits = 01	Critical input: vectored

60.6.3 State logic unit (SLU)

Complex triggers and system performance monitor functions are implemented in the SLU. The SPU creates debug events based upon states in a sequence. A sequence can consist of a single state, or any number of states up to 8. Each state consists of combinational logic allowing AND/OR operations on inputs from the trigger source unit. Single or multiple (up to 8) actions can be triggered by a state machine.

A logic diagram of one SLU state is shown in *Figure 1209*.

Figure 1209. SLU block diagram



Each state consists of AND gates, the outputs of which are ORed together.

- Each AND gate can have up to 4 inputs, selected from the trigger source unit by the user.
- The user has an option to invert one input signal into the AND gate.
- The OR gate has a user selectable option to invert the output.
- The output of the OR gate is used as the trigger to move to the next state in the sequence.

The SLUs are responsible for performing operations on the selected input events to produce derived events. Each SLU has a control register that manages how the various inputs are combined to produce a derived event.

An event is triggered from each state based on all the possible selected inputs for each state. This event is used to trigger an Action.

Note: With only one optional inversion at the input of each AND gate and optional inversion at the output of the OR gate, many combinations are not possible.

60.6.4 Sequence formation

Event sequencing can be accomplished with the if-then-else operations. The next event can be fed back as a necessary condition for the current event. For example, event5 can be designated as a necessary condition for event4 in case test condition fails in event5. Event5 can be in turn be designated as necessary for event6 in case test condition passes, and so on. In this way, events can be chained together to create a sequence.

To create complex triggers, a configurable state machine is implemented. This allows the user to join states together with if-then-else operations to create a sequence. The SPU supports up to four simultaneous sequences, however only eight states are supported regardless of the number of active sequences, and each state can only be used in one unique sequence.

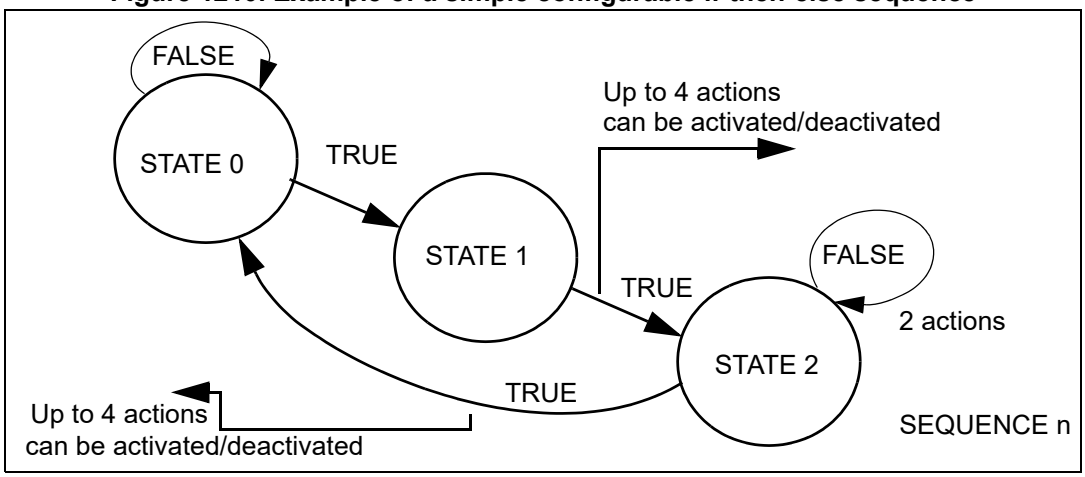
For example, the following state configurations are possible:

- 1 active sequence, with 8 states
- 2 active sequences, each with 4 states
- 2 active sequences, the first with 3 states and the second with 5 states
- 4 active sequences, each with 2 states
- 4 active sequences, each with 1 state

Each sequence has the following options which are shown in *Figure 1210*:

- The ability to optionally trigger one or more actions based on a true condition from any state in the sequence.
- The ability to optionally trigger one or more actions based on a false condition from any state in the sequence.
- Each state in a sequence has the ability to route to another state on a true condition from the state logic, and the ability to route to a different state based on a false condition from the state logic.

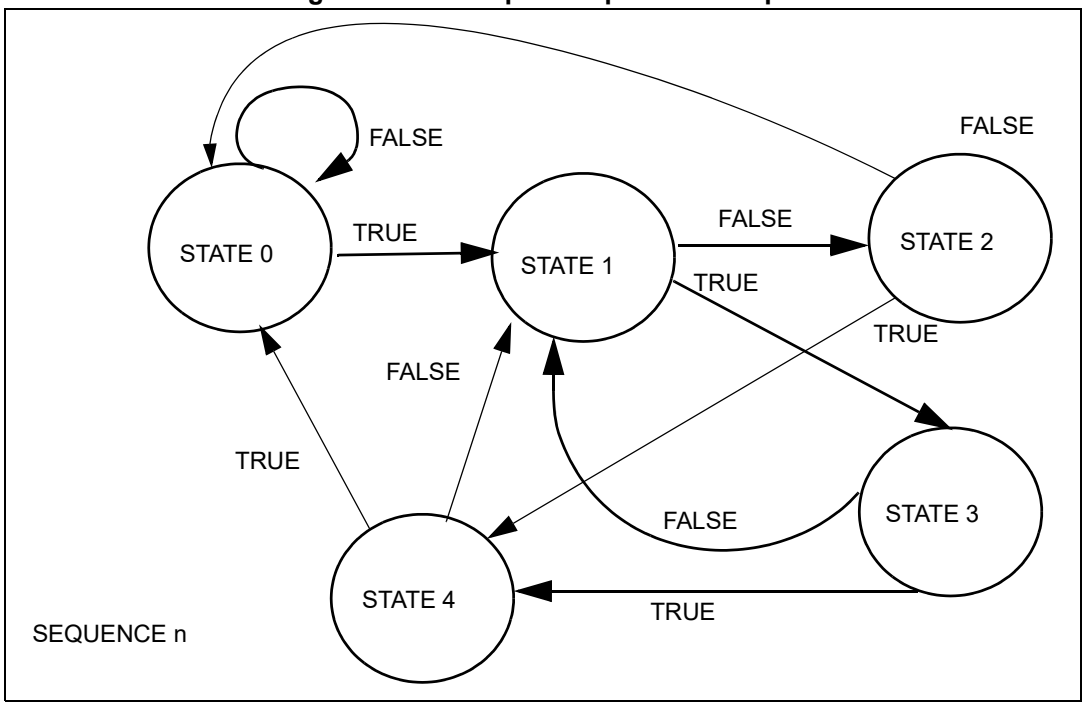
Figure 1210. Example of a simple configurable if-then-else sequence



60.6.4.1 Complex sequence example

A user can create a complex sequence using five states as shown in the state transition diagram in *Figure 1211*.

Figure 1211. Complex sequence example



60.6.4.2 Typical sequence example

For this example, the user would like to create a timestamp in the trace stream each time a function is entered and stop the trace signal on the 12th occurrence of the function. This use case can be implemented as follows:

- State 0 takes an instruction address compare as its input. On a match of the instruction address to a specific function, called function_x, the sequence moves to State 1. Upon this action, State0 is configured to insert a timestamp into the trace stream and also increment an SPU counter, say counter[0].
- The sequence is now in State 1. State 1 takes an input from counter[0]. Counter[0] has been pre-configured to create a match event when the counter value is equal to 12.
  - If the value of counter[0] is less than 12, then the sequence moves back to State0.
  - If the value of couner[0] is equal to 12, then the sequence completes and an action from State1 disables the trace signal.

60.6.4.3 Sequence status register description

A status register providing feedback on each sequence is provided to allow debug tools to identify the source of a debug event from the SPU. The status register consists of two field types:

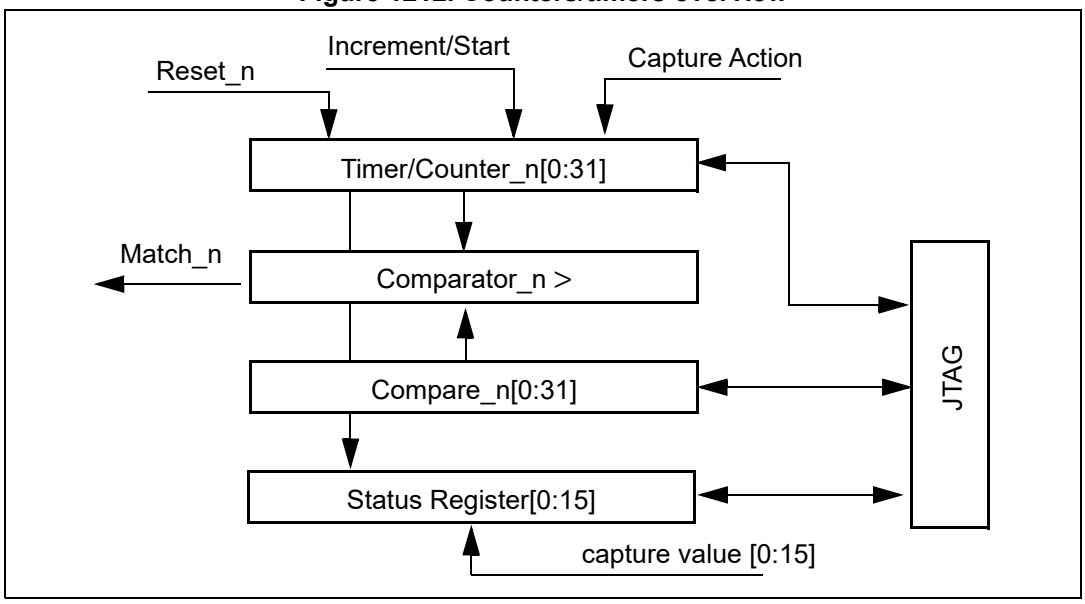
- States: each state has a sticky bit that is set upon a true match of the state logic. This sticky bit is cleared upon a write to this register
- Sequence status: each sequence has a field indicating the active state of the state machine. This is a read only field

60.6.5 Performance counters/timers unit

The 32-bit SPU counters may be used to delay events, count distances, count latencies or count events as shown in *Figure 1212*. Each counter can be configured as either a counter or a timer via the configuration register. The basic counting operation is straightforward: once enabled (default is disabled mode), the counter increments using the platform clock whenever the selected input is driven to a logic 1.

Access (via the JTAG interface) to each of the counter/timer registers allows the configuration of the timer/counter, the compare registers and global SPU timer/counter status registers. These registers are accessible via the JTAG interface even when the SPU is in disabled mode.

Figure 1212. Counters/timers overview



60.6.5.1 Counter events and actions

Each counter is capable of detecting one event condition, such as compare value match. The compare value match is used for triggering actions in the SPU. Each counter can be reset in response to a reset action generated from the SPU. Capture registers are available for capturing the counter state in response to a SPU trigger condition. This capture information is updated in the status register, which is accessible to users via JTAG interface. These status registers are type write 1 to clear (W1C).

There are two classes of signals, actions and events:

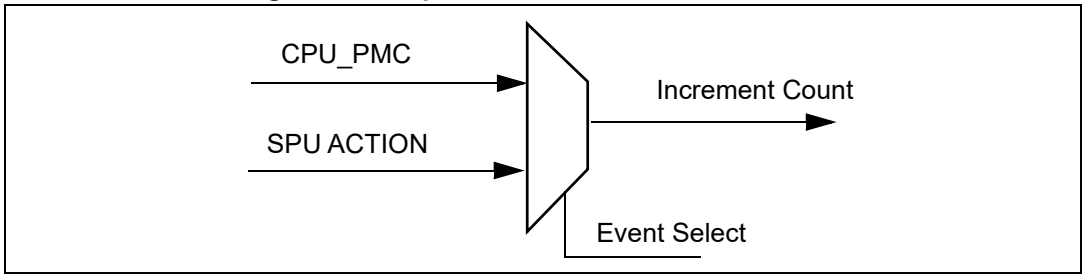
- Actions: each timer/counter has an interface from the action unit to start/increment and reset as appropriate. The start/increment signal is a single signal, with the operation selected in the SPU configuration registers. The generation of the appropriate count/reset signal is generated in the SPU.

  On an action to insert the timer/counter value into the trace stream, the value stored in the timer/counter is buffered inside the SPU and routed to the appropriate SoC block to insert the value into the trace stream one by one.
- Event: on a match event where a timer/counter value matches the value of the respective compare register, a signal is sent to the trigger source unit to indicate this.

60.6.5.2 Counter input selection at SPU level

Counter increment inputs derive from the SPU action or direct from the CPU performance measurement counters (CPU_PMC), as shown in *Figure 1213*. These inputs include signals from the primary SPU muxes or SPU events of counters.

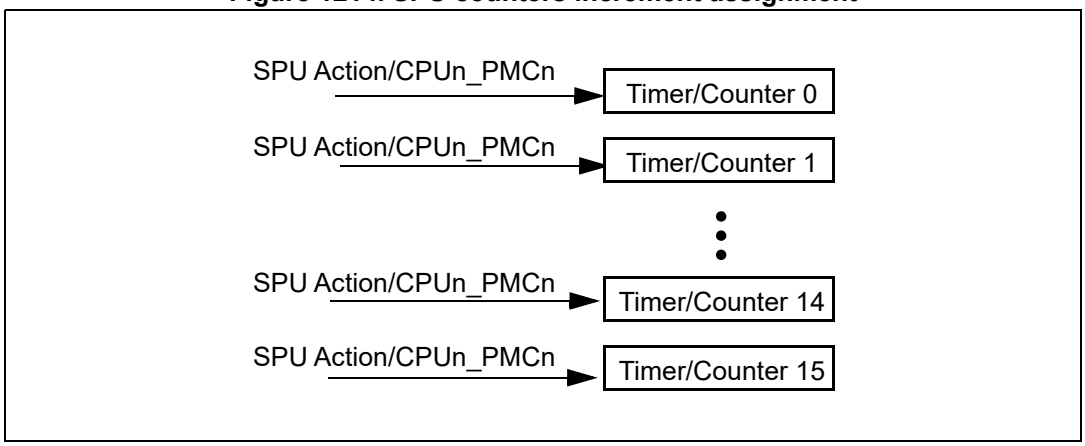
Figure 1213. nput selection for count increment



60.6.5.3 Counters increment assignment

When configured as a counter, the counter has an increment signal. This can optionally be routed from the SPU action unit, or directly from a CPU performance counter signal. The assignment of which signal is used to increment a counter is handled in the SPU, and is configurable by user. The assignment of increment signals to counters is shown in *Figure 1214*.

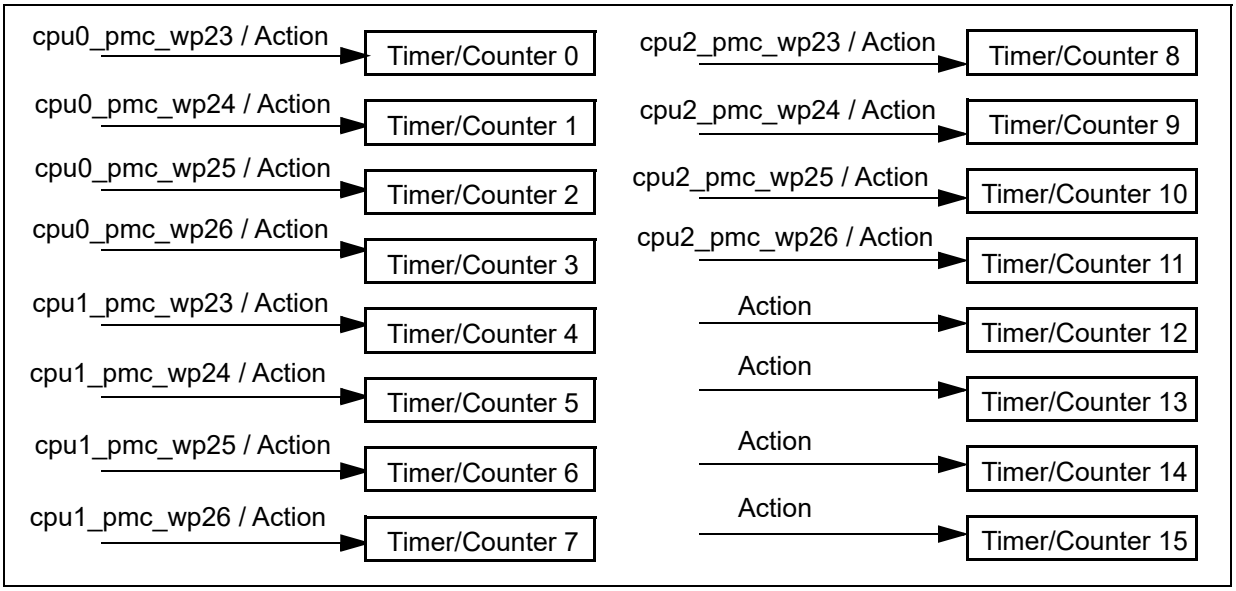
Figure 1214. SPU counters increment assignment



60.6.5.4 CPU PMC mapping to counters

Each of the CPUs in the system has four performance measurement counters (PMC). The outputs (overflow signals) from these counters (of CPUn) are routed to the SPU debug unit. The counters inside the SPU can be configured to count the pulses from the PMCs. There is a one-to-one mapping between the PMC inputs and the counters that they trigger. Their mapping is shown in the *Figure 1215*.

Figure 1215. CPU PMC mapping inside the SPU unit



60.6.6 Action unit

The action unit block of the SPU is responsible for the generating the action triggers supported by various actions. A state has true and false (if-else-than) condition. Four simultaneous actions can be generated under true/false condition of a particular state. The list of the actions is given in the *Section 60.2*. The actions triggers are mapped to the events by programming the set of SPU group action control for events registers**.** Each event can be programmed to trigger interrupts, timestamp watch points, trace signals, PMCs, start/reset/increment counters and many other general actions.

60.6.7 Optional trace group

User defined trace group can be defined through the configuration register. Configuration registers, defined in *Table 1156*, are available where the user can group the different traces from same or different clients.

61 e200z420n3 Nexus 3 Module

The e200z420n3 Nexus 3 module provides real-time development capabilities for corresponding core processors in compliance with the IEEE-ISTO 5001 standard. This module provides development support capabilities without requiring the use of address and data pins for internal visibility.

A portion of the pin interface (the JTAG port) is also shared with the OnCE / Nexus 1 unit. The IEEE-ISTO 5001 standard defines an extensible auxiliary port which is used in conjunction with the JTAG port in these processors.

61.1 Introduction

61.1.1 General Description

This chapter defines the auxiliary pin functions, transfer protocols and standard development features of a Class 3 device in compliance with the IEEE-ISTO 5001 standard. The development features supported are Program Trace, Data Trace, Watchpoint Messaging, Ownership Trace, Data Acquisition Messaging, and Read/Write Access via the JTAG interface. The Nexus 3 module also supports two Class 4 features: Watchpoint Triggering, and Processor Overrun Control.

61.1.2 Terms and Definitions

Table 1164 contains a set of terms and definitions associated with the Nexus 3 module.

Table 1164. Terms and Definitions

Term	Description
IEEE-ISTO 5001	Consortium & standard for real-time embedded system design. World wide Web documentation at http://www.ieee-isto.org/Nexus5001
Auxiliary Port	Refers to Nexus auxiliary port. Used as auxiliary port to the IEEE 1149.1 JTAG interface.
Branch Trace Messaging (BTM)	Visibility of addresses for taken branches and exceptions, and the number of sequential instructions executed between each taken branch.
Data Read Message (DRM)	External visibility of data reads to memory-mapped resources.
Data Write Message (DWM)	External visibility of data writes to memory-mapped resources.
Data Trace Messaging (DTM)	External visibility of how data flows through the embedded system. This may include DRM and/or DWM.
Data Acquisition Messaging (DQM)	Data Acquisition Messaging (DQM) allows code to be instrumented to export customized information to the Nexus Auxiliary Output Port.
JTAG Compliant	Device complying to IEEE 1149.1 JTAG standard
JTAG IR & DR Sequence	JTAG Instruction Register (IR) scan to load an opcode value for selecting a development register. The JTAG IR corresponds to the OnCE command register (OCMD). The selected development register is then accessed via a JTAG Data Register (DR) scan.

Table 1164. Terms and Definitions (continued)

Term	Description
Nexus1	The (OnCE) debug module. This module integrated with each processor provides all static (core halted) debug functionality. This module is compliant with Class1 of the IEEE-ISTO 5001 standard.
Ownership Trace Message (OTM)	Visibility of process/function that is currently executing.
Public Messages	Messages on the auxiliary pins for accomplishing common visibility and controllability requirements
SoC	“System-on-a-Chip”. SoC signifies all of the modules on a single die. This generally includes one or more processors with associated peripherals, interfaces & memory modules.
Standard	The phrase “according to the standard” is used to indicate according to the IEEE-ISTO 5001 standard.
Transfer Code (TCODE)	Message header that identifies the number and/or size of packets to be transferred, and how to interpret each of the packets.
Watchpoint	A Data or Instruction Breakpoint or other debug event which does not cause the processor to halt. Instead, a pin is used to signal that the condition occurred. A Watchpoint Message may also be generated.

61.1.3 Feature List

The Nexus 3 module is compliant with Class 3 of the IEEE-ISTO 5001 standard, with additional Class 4 features available. The following features are implemented:

- Program Trace via Branch Trace Messaging (BTM). Branch trace messaging displays program flow discontinuities (direct and indirect branches, exceptions, etc.), allowing the development tool to interpolate what transpires between the discontinuities. Thus static code may be traced.
- Data Trace via Data Write Messaging (DWM) and Data Read Messaging (DRM). This provides the capability for the development tool to trace reads and/or writes to selected internal memory resources.
- Ownership Trace via Ownership Trace Messaging (OTM). OTM facilitates ownership trace by providing visibility of which process ID or operating system task is activated.

An Ownership Trace Message is transmitted when a new process/task is activated, allowing the development tool to trace ownership flow.

- Run-time access to embedded processor memory map via the JTAG port. This allows for enhanced download/upload capabilities.
- Watchpoint Messaging via the auxiliary pins
- Watchpoint Trigger enable of Program and/or Data Trace Messaging
- External hardware trigger inputs to independently enable/disable Program Trace, Data Trace, Ownership Trace, and Watchpoint Trace Messaging
- Auxiliary interface for higher data input/output
  - Configurable (min/max) Message Data Out pins (nex_mdo[n:0])
  - One (1) or two (2) Message Start/End Out pins (nex_mseo_b[1:0])
  - One (1) Read/Write Ready pin (nex_rdy_b) pin
  - One (1) Read/Write Error pin (nex_err_b) pin
  - One (1) Watchpoint Event output pin (nex_evto_b)
  - One (1) Event In pin (nex_evti_b)
  - One (1) MCKO (Message Clock Out) pin
- Registers for Program Trace, Data Trace, Ownership Trace and Watchpoint Trigger.
- All features controllable and configurable via the JTAG port
- Conditional software control of the module via SoC signaling input (nex_sfwcntl_en)

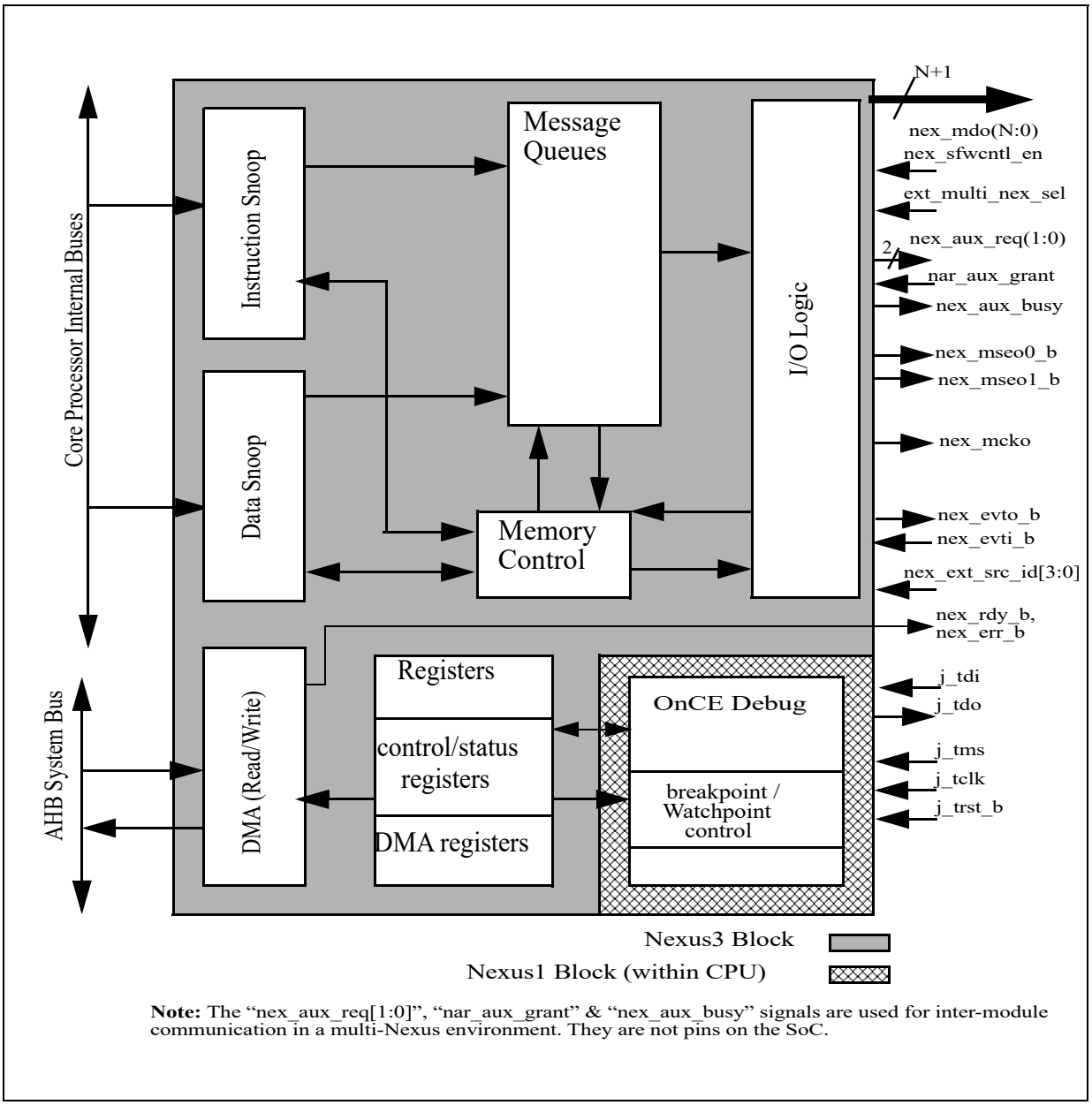
Note: For multi-Nexus implementations, the configuration of the Message Data Out pins is controlled by the Port Control Register (@ the SoC level). For single Nexus implementations (not normally implemented on an SoC), this configuration is controlled by Development Control Register 1 (DC1) within the Nexus 3 module.

In either implementation, Full Port Mode (FPM - maximum number of MDO pins) or Reduced Port Mode (RPM - minimum number of MDO pins) are supported. This setting should not be changed while the system is running.

The configuration of the Message Start/End Out pins (1 or 2) is determined at the SOC integration level. This option will be hard-wired based on SOC bandwidth requirements.

61.1.4 Functional Block Diagram

Figure 1216. Nexus 3 Functional Block Diagram



Note: The "nex_aux_req[1:0]", "nar_aux_grant" & "nex_aux_busy" signals are used for inter-module communication in a multi-Nexus environment. They are not pins on the SoC.

61.2 Enabling Nexus 3 Operation

The Nexus module is enabled by loading a single instruction (*NEXUS3-ACCESS*) into the JTAG Instruction Register (IR) (OnCE OCMD register). For the Nexus3 module, the OCMD value is 0b0001111100. Once enabled, the module will be ready to accept control input via the JTAG/OnCE pins.

Enabling the Nexus 3 module automatically enables the generation of Debug Status Messages.

The Nexus 3 module is disabled when the JTAG state machine initially reaches the Test-Logic-Reset state from any other state. This state can be reached by the assertion of the **j_trst_b** pin or by cycling through the state machine using the **j_tms pin**. The Nexus module will also be disabled if a Power-on-Reset (POR) event occurs. If the Nexus 3 module is disabled, no trace output will be provided, and the module will disable (drive inactive) auxiliary port output pins (**nex_mdo[n:0], nex_mseo[1:0], nex_mcko**). Nexus registers will not be available for reads or writes.

In order to support software control of the Nexus 3 module when no external development tool is present, the Nexus 3 module is not forced to be disabled when the JTAG state remains in the Test-Logic-Reset state. Software is allowed to control the Nexus 3 module when the input signal **nex_sfwcntl_en** is asserted by the SoC. This signal is intended to provide a mechanism for allowing software to use the Nexus 3 module to fill on-chip trace buffers or other visibility mechanisms. It is up to the SoC to determine whether a top-level Nexus 3 controller has been enabled by a hardware debugger, or whether appropriate security mechanisms have granted the capability for software to use these resources, and to drive the appropriate value to the **nex_sfwcntl_en** input. Software can enable the module by a write to any of the module's DCRs when **nex_sfwcntl_en** is asserted.

Reset of the Nexus 3 module is accomplished by a transition on **j_trst_b**, on initial entry into the **Test_Logic_Reset** state from another state, or if a Power-on-Reset (POR) event occurs. The module is not reset by the CPU's **p_reset_b** signal, even when software has control of the module.

61.2.1 Interaction with Low Power Modes

The Nexus 3 module will continue to operate in the Waiting and Halted states, as long as **nex_clk** remains active. In the Stopped state, **nex_clk** is gated off internally to the Nexus 3 logic, therefore watchpoint or hardware triggering recognition, message generation, and Nexus 3 read-write access to memory is suspended. The Nexus 3 logic will wait to enter the Stopped state until the message FIFOs are empty. Once the Stopped state has been entered, no further messages are queued and no triggering conditions are monitored. Upon exiting the Stopped state, normal functions will resume assuming **nex_clk** is active.

61.3 TCODEs supported

The Nexus 3 pins allow for flexible transfer operations via Public Messages. A TCODE defines the transfer format, the number and/or size of the packets to be transferred, and the purpose of each packet. The IEEE-ISTO 5001 2010 standard defines a set of public messages and allocates additional TCODEs for vendor-specific features outside the scope of the public messages. The Nexus 3 block supports the TCODEs shown in *Table 1165*.

Table 1165. Supported TCODEs

Message Name	Minimum Field Size (bits)	Maximum Field Size (bits)	Field Name	Field Type	Field Description
Debug Status	6	6	TCODE	fixed	TCODE number = 0
	4	4	SRC	fixed	source processor identifier
	8	8	STATUS	fixed	Development Status Register (DS[31:24])
Ownership Trace Message	6	6	TCODE	fixed	TCODE number = 2
	4	4	SRC	fixed	source processor identifier
	1	32	PROCESS	variable	Task/Process ID tag
Program Trace: Direct Branch Message	6	6	TCODE	fixed	TCODE number = 3
	4	4	SRC	fixed	source processor identifier
	1	8	ICNT	variable	# sequential instructions completed since last predicate instruction, transmitted instruction count, or taken change of flow
Program Trace: Indirect Branch Message	6	6	TCODE	fixed	TCODE number = 4
	4	4	SRC	fixed	source processor identifier
	1	1	MAP	fixed	(always set to 0)
	1	8	ICNT	variable	# sequential instructions completed since last predicate instruction, transmitted instruction count, or taken change of flow
	1	32	U-ADDR	variable	unique part of target address for taken branches/exceptions
Data Trace: Data Write Message	6	6	TCODE	fixed	TCODE number = 5
	4	4	SRC	fixed	source processor identifier
	1	1	MAP	fixed	(always set to 0)
	4	4	DSZ	fixed	data size (refer to *Table 1170*)
	1	32	U-ADDR	variable	unique portion of the data write address
	1	64	DATA	variable	data write value(s) (see Data Trace section for details)

Table 1165. Supported TCODEs (continued)

Message Name	Minimum Field Size (bits)	Maximum Field Size (bits)	Field Name	Field Type	Field Description
Data Trace: Data Read Message	6	6	TCODE	fixed	TCODE number = 6
	4	4	SRC	fixed	source processor identifier
	1	1	MAP	fixed	(always set to 0)
	4	4	DSZ	fixed	data size (refer to *Table 1170*)
	1	32	U-ADDR	variable	unique portion of the data read address
	1	64	DATA	variable	data read value(s) (see Data Trace section for details)
Data Acquisition Message	6	6	TCODE	fixed	TCODE number = 7
	4	4	SRC	fixed	source processor identifier
	8	8	DQTAG	fixed	identification tag taken from $DEVENT_{DQTAG}$ register field
	1	32	DQDATA	variable	exported data taken from DDAM register
Error Message	6	6	TCODE	fixed	TCODE number = 8
	4	4	SRC	fixed	source processor identifier
	4	4	ETYPE	fixed	error type
	8	8	ECODE	fixed	error code
Program Trace: Synchronization Message	6	6	TCODE	fixed	TCODE number = 9
	4	4	SRC	fixed	source processor identifier
	1	1	MAP	fixed	(always set to 0)
	1	8	I-CNT	variable	# sequential instructions completed since last predicate instruction, transmitted instruction count, or taken change of flow. Cleared for most sync conditions.
	1	32	F-ADDR	variable	full target address (leading zero (0) truncated)
Program Trace: Direct Branch Message w/ Sync	6	6	TCODE	fixed	TCODE number = 11
	4	4	SRC	fixed	source processor identifier
	1	8	ICNT	variable	# sequential instructions completed since last predicate instruction, transmitted instruction count, or taken change of flow
	1	32	F-ADDR	variable	full target address (leading zeros truncated)

Table 1165. Supported TCODEs (continued)

Message Name	Minimum Field Size (bits)	Maximum Field Size (bits)	Field Name	Field Type	Field Description
Program Trace: Indirect Branch Message w/ Sync	6	6	TCODE	fixed	TCODE number = 12
	4	4	SRC	fixed	source processor identifier
	1	1	MAP	fixed	(always set to 0)
	1	8	ICNT	variable	# sequential instructions completed since last predicate instruction, transmitted instruction count, or taken change of flow
	1	32	F-ADDR	variable	full target address (leading zeros truncated)
Data Trace: Data Write Message w/ Sync	6	6	TCODE	fixed	TCODE number = 13
	4	4	SRC	fixed	source processor identifier
	1	1	MAP	fixed	(always set to 0)
	4	4	DSZ	fixed	data size (refer to *Table 1170*)
	1	32	F-ADDR	variable	full access address (leading zeros truncated)
	1	64	DATA	variable	data write value(s) (see Data Trace section for details)
Data Trace: Data Read Message w/ Sync	6	6	TCODE	fixed	TCODE number = 14
	4	4	SRC	fixed	source processor identifier
	1	1	MAP	fixed	(always set to 0)
	4	4	DSZ	fixed	data size (refer to *Table 1170*)
	1	32	F-ADDR	variable	full access address (leading zeros truncated)
	1	64	DATA	variable	data read value(s) (see Data Trace section for details)
Watchpoint Message	6	6	TCODE	fixed	TCODE number = 15
	4	4	SRC	fixed	source processor identifier
	1	32	WPHIT	variable	Field indicating watchpoint source(s) (leading zeros truncated)
Resource Full Message	6	6	TCODE	fixed	TCODE number = 27
	4	4	SRC	fixed	source processor identifier
	4	4	RCODE	fixed	resource code (refer to *Table 1168*): indicates which resource is the cause of this message
	1	32	RDATA	variable	branch / predicate instruction history (see Section *Section 61.12.4*.)

Table 1165. Supported TCODEs (continued)

Message Name	Minimum Field Size (bits)	Maximum Field Size (bits)	Field Name	Field Type	Field Description
Program Trace: Indirect Branch History Message	6	6	TCODE	fixed	TCODE number = 28 (see Note below)
	4	4	SRC	fixed	source processor identifier
	1	1	MAP	fixed	(always set to 0)
	1	8	I-CNT	variable	# sequential instructions completed since last predicate instruction, transmitted instruction count, or taken change of flow
	1	32	U-ADDR	variable	unique part of target address for taken branches/exceptions
	1	32	HIST	variable	branch / predicate instruction history (see *Section 61.12.1*)
Program Trace: Indirect Branch History Message w/ Sync	6	6	TCODE	fixed	TCODE number = 29 (see Note below)
	4	4	SRC	fixed	source processor identifier
	1	1	MAP	fixed	(always set to 0)
	1	8	I-CNT	variable	# sequential instructions completed since last predicate instruction, transmitted instruction count, or taken change of flow
	1	32	F-ADDR	variable	full target address (leading zero (0) truncated)
	1	32	HIST	variable	branch / predicate instruction history (see *Section 61.12.1*.)
Program Trace: Program Correlation Message	6	6	TCODE	fixed	TCODE number = 33
	4	4	SRC	fixed	source processor identifier
	4	4	EVCODE	fixed	event correlated w/ program flow (refer to *Table 1169*)
	2	2	CDF	fixed	# fields of information in CDATA. 00 reserved, 01 one field (CDATA1) (reserved), 10 two fields (CDATA1 + CDATA2), 11 three fields (reserved)
	1	8	I-CNT	variable	# sequential instructions completed since last predicate instruction, transmitted instruction count, or taken change of flow
	1	32	CDATA1	variable	correlation data field 1 [branch / predicate instruction history] (see *Section 61.12.5*.)
	0	32	CDATA2	variable	correlation data field 2- PID info (see *Section 61.12.5*.)

Note: Program Trace can be implemented using either Branch History/Predicate Instruction Messages, or traditional Direct/Indirect Branch Messages. The user can select between the two types of Program Trace. The advantages for each are discussed in Section 61.12.1. If the Branch History method is selected, the shaded TCODES above will not be messaged out.

Table 1166 shows the error code encodings used when reporting an error via the Nexus 3 Error Message.

Table 1166. Error Code (ECODE) Encoding (TCODE = 8)

Error Code	Description
xxxxxxx1	Watchpoint Trace Message(s) Lost
xxxxxx1x	Data Trace Message(s) Lost
xxxxx1xx	Program Trace Message(s) Lost
xxxx1xxx	Ownership Trace Message(s) Lost
xxx1xxxx	Status Message(s) Lost (Debug Status messages, etc.)
xx1xxxxx	Data Acquisition Message(s) Lost
x1xxxxxx	Reserved
1xxxxxxx	Reserved

Table 1167 shows the error type encodings used when reporting an error via the Nexus 3 Error Message.

Table 1167. Error Type (ETYPE) Encoding (TCODE = 8)

Error Type	Description
0000	Message Queue Overrun caused one or more messages to be lost
0001	Contention with higher priority messages caused one or more messages to be lost
0010	Reserved
0011	Reserved
0100	Reserved
0101	Invalid access opcode (Nexus Register unimplemented)
0110-1111	Reserved

Table 1168 shows the encodings used for resource codes for certain messages.

Table 1168. Resource Code (RCODE) values (TCODE = 27)

Resource Code	Description
0000	Program Trace Instruction counter reached 255 and was reset.
0001	Program Trace, Branch / Predicate Instruction History full. This type of packet is terminated by a stop bit set to 1 after the last history bit.

Table 1169 shows the event code encodings used for certain messages.

Table 1169. Event Code (EVCODE) Encoding (TCODE = 33)

Event Code	Description
0000	Entry into Debug Mode
0001	Entry into Low Power Mode (CPU only)
0010-0011	Reserved for future functionality
0100	Disabling Program Trace
0101	Process ID value is established in PID0/NPIDR via **mtspr PID0/NPIDR**
0110-1000	Reserved for future functionality
1001	Begin masking of program trace messages due to MSR_{PMM}=0 and $DC4_{PTMARK}$=1
1010	Branch and link occurrence (direct branch function call)
1011-1111	Reserved for future functionality

Table 1170 shows the data trace size encodings used for certain messages.

Table 1170. Data Trace Size (DSZ) Encodings (TCODE = 5,6,13,14)

DTM Size Encoding	Transfer Size
0000	0 no data
0001	Byte
0010	Halfword (2 bytes)
0011	Three bytes
0100	Word (4 bytes)
0101	Five bytes
0110	Six bytes
0111	Seven bytes
1000	Doubleword (8 bytes)
1001-1111	Reserved

61.4 Nexus 3 Programmer's Model

This section describes the Nexus 3 programmers model. Nexus 3 registers are accessed using the JTAG/OnCE port in compliance with IEEE 1149.1, or by software via DCRs. See *Section 61.5* and *Section 61.6* for details on Nexus 3 register access.

Note: Nexus 3 registers and output signals are numbered using bit 0 as the least significant bit. This bit ordering is consistent with the ordering defined by the IEEE-ISTO 5001 standard.

Table 1171 details the register map for the Nexus 3 module.

Table 1171. Nexus 3 Register Map

Nexus Register	Nexus Access Opcode	Read/ Write	Read Address	Write Address	DCR #(1)
Client Select Control (CSC)(2)	0x1	R	0x02	—	—
Port Configuration Register (PCR)(3)	PCR_INDEX(2)	R/W	—	—	—
Development Control 1 (DC1)	0x2	R/W	0x04	0x05	368
Development Control 2 (DC2)	0x3	R/W	0x06	0x07	369
Development Control 3 (DC3)	0x4	R/W	0x08	0x09	370
Development Control 4 (DC4)	0x5	R/W	0x0A	0x0B	371
Reserved	0x6	R/W	0x18	0x19	—
Read/Write Access Control/Status (RWCS)	0x7	R/W	0x0E	0x0F	—
Reserved	0x8	R/W	0x18	0x19	—
Read/Write Access Address (RWA)	0x9	R/W	0x12	0x13	—
Read/Write Access Data (RWD)	0xA	R/W	0x14	0x15	—
Watchpoint Trigger (WT)	0xB	R/W	0x16	0x17	375
Reserved	0xC	R/W	0x18	0x19	—
Data Trace Control (DTC)	0xD	R/W	0x1A	0x1B	376
Data Trace Start Address 1 (DTSA1)	0xE	R/W	0x1C	0x1D	377
Data Trace Start Address 2 (DTSA2)	0xF	R/W	0x1E	0x1F	378
Data Trace Start Address 3 (DTSA3)	0x10	R/W	0x20	0x21	379
Data Trace Start Address 4 (DTSA4)	0x11	R/W	0x22	0x23	380
Data Trace End Address 1 (DTEA1)	0x12	R/W	0x24	0x25	381
Data Trace End Address 2 (DTEA2)	0x13	R/W	0x26	0x27	382
Data Trace End Address 3 (DTEA3)	0x14	R/W	0x28	0x29	383
Data Trace End Address 4 (DTEA4)	0x15	R/W	0x2A	0x2B	408
Reserved	0x16-0x2F	-	0x28-0x5E	0x29-0x5F	—
Development Status (DS)	0x30	R	0x60	—	409
Reserved	0x31	R/W	0x62	0x63	—
Overrun Control (OVCR)	0x32	R/W	0x64	0x65	410
Watchpoint Mask (WMSK)	0x33	R/W	0x66	0x67	411
Reserved	0x34	-	0x68	0x69	
Program Trace Start Trigger Control (PTSTC)	0x35	R/W	0x6A	0x6B	412
Program Trace End Trigger Control (PTETC)	0x36	R/W	0x6C	0x6D	413
Data Trace Start Trigger Control (DTSTC)	0x37	R/W	0x6E	0x6F	414
Data Trace End Trigger Control (DTETC)	0x38	R/W	0x70	0x71	415
Reserved	0x39-0x3F	—	0x72-0x7E	0x73-0x7F	—

1. Software access via the **mfdcr** and **mtdcr** instructions use these values for the DCR number. Software writes to these registers via **mtdcr** when **nex_sfwcntl_en** is negated are ignored.

2. The CSC and PCR registers are shown in this table as part of the Nexus programmer's model. They are only present at the top level SoC Nexus controller in a multi-Nexus implementation, not in the Nexus 3 module. The SoC's CSC Register is readable through Nexus, but the PCR is shown for reference only here.
3. The "PCR_INDEX" is a parameter determined by the SoC.

61.4.1 Client Select Control Register (CSC)

The CSC Register determines which Nexus client is under development. This register is present at the top-level SOC Nexus 3 controller to select one of multiple on-chip Nexus 3 units.

Offset: Refer to *Table 1171* Access: Read-only

	0	1	2	3	4	5	6	7
R	0				CS			
W								
Reset	0	0	0	0	0	0	0	0

Figure 1217. Client Select Control register (CSC)

Table 1172. CSC Register field descriptions

Field	Description
4:7 CS	Client Select Control xxxx Nexus client (SoC level)

61.4.2 Port Configuration Register - reference only (PCR)

The Port Configuration Register (PCR) controls the basic port functions for all Nexus modules in a multi-Nexus environment. This includes clock control and auxiliary port width. All bits in this register are writable only once after system reset.

Note: This register is "reference only" and should not be used for any configuration (Instead SoC module NPC.PCR register is used for the configuration). Its OPC, MCK_EN and MCK_DIV bits are present but have no impact.

Offset: Refer to *Table 1171* Access: Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	OPC	0	MCK_EN	MCK_DIV			0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

Figure 1218. Port Configuration Register (PCR)

Table 1173. PCR field descriptions

Field	Description
0 OPC	Output Port Mode Control (SoC Level) 0 Reduced Port Mode configuration (min# **nex_mdo[n:0]** pins defined by SOC) 1 Full Port Mode configuration (max# **nex_mdo[n:0]** pins defined by SOC)
2 MCK_EN	MCKO Clock Enable (SoC Level) 0 **nex_mcko** is disabled 1 **nex_mcko** is enabled
3:5 MCK_DIV	MCKO Clock Divide Ratio (see note below) (SoC Level) 000 **nex_mcko** is 1x processor clock freq. 001 **nex_mcko** is 1/2x processor clock freq. 010 Reserved (default to 1/2x processor clock freq.) 011 **nex_mcko** is 1/4x processor clock freq. 100 Reserved (default to 1/2x processor clock freq.) 101 Reserved (default to 1/2x processor clock freq.) 110 Reserved (default to 1/2x processor clock freq.) 111 **nex_mcko** is 1/8x processor clock freq.

Note: The CSC and PCR Registers exist in a separate module at the SoC level in a multi-Nexus environment. If the core Nexus 3 module is the only Nexus module, these registers are not implemented and the Nexus 3 defined Development Control Register 1 (DC1) is used to control the SoC-level Nexus port functionality.

61.4.3 Nexus Development Control Register 1 (DC1)

Nexus Development Control Register 1 is used to control the basic development features of the Nexus 3 module. Development Control Register 1 is shown in *Figure 1219* and its fields are described in *Table 1174*.

Note: Its *OPC and MCK_DIV bits are present but have no impact.*

Offset: Refer to *Table 1171* Access: Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	OPC	MCK_DIV		0	PTM	0	0	OTS	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	POTD	TSEN		EOC		EIC		0	0	TM					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1219. Development Control register 1 (DC1)

Table 1174. DC1 Register field descritions

Field	Description
0 OPC	Output Port Mode Control 0 Reduced Port Mode configuration (min# **nex_mdo[n:0]** pins defined 1 Full Port Mode configuration (max# **nex_mdo[n:0]** pins defined
1:2 MCK_DIV	MCKO Clock Divide Ratio (see note below) 00 **nex_mcko** is 1x processor clock freq. 01 **nex_mcko** is 1/2x processor clock freq. 10 **nex_mcko** is 1/4x processor clock freq. 11 **nex_mcko** is 1/8x processor clock freq.
4 PTM	Program Trace Method 0 Program Trace uses traditional Branch Messages 1 Program Trace uses Branch History Messages
7 OTS	Ownership Trace PID Select 0 PID0 data is transmitted within Ownership Trace Messages 1 Nexus PID Register (NPIDR) data is transmitted within Ownership Trace Messages
17 POTD	Periodic Ownership Trace Disable 0 Periodic Ownership Trace message events are enabled 1 Periodic Ownership Trace message events are disabled
18:19 TSEN	Timestamp Enable (not implemented, write to 00) 00 Timestamp is disabled
20:21 EOC	EVTO Control 00 **nex_evto_b** upon occurrence of Watchpoints (configured in DC2 and DC3) 01 **nex_evto_b** upon entry into Debug Mode 1x Reserved
22:23 EIC	EVTI Control 00 **nex_evti_b** is used for synchronization (Program Trace Sync msg is generated) 01 **nex_evti_b** is used for Debug request 10 **nex_evti_b** is disabled 1x Reserved
26:31 TM	Trace Mode(1) 000000 All Trace Disabled xxxxx1 Ownership Trace enabled xxxx1x Data Trace enabled xxx1xx Program Trace enabled xx1xxx Watchpoint Trace enabled x1xxxx Reserved 1xxxxx Data Acquisition Trace enabled

1. This field may be updated by hardware in response to watchpoint triggering or external hardware trigger inputs if not already enable for tracing by a previous direct write to DC1. Direct writes to this field to enable one or more trace types take precedence over hardware updates. Refer to *Section 61.4.7* for more information on watchpoint triggering. Refer to *Section 61.16* for more information on external hardware triggering.

Note: The Output Port Mode Control bit (OPC) and MCKO Clock Divide Ratio bits (MCK_DIV) MUST ONLY be modified during system reset or debug mode to insure correct output port and output clock functionality. It is also recommended that all other bits of the DC1 also only be modified in one of these two modes.

61.4.4 Nexus Development Control Registers 2-3 (DC2-3)

Nexus Development Control Registers 2 and 3 are used to control output signaling on the Nexus 3 module. A table of watchpoints can be found in the Core (e200z420n3) Core Debug Support chapter.

Development Control Register 2 is shown in *Figure 1220* and its fields are described in *Table 1175*.

Offset: Refer to *Table 1171* Access: Read/Write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	EWC															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1220. Development Control register 2 (DC2)

Table 1175. DC2 Register field descriptions

Field	Description
16:31 EWC	EVTO Watchpoint Configuration(1) 0000000000000000 No Watchpoints #0-15 trigger **nex_evto_b** xxxxxxxxxxxxxxx1 Watchpoint #0 triggers **nex_evto_b** xxxxxxxxxxxxxx1x Watchpoint #1 triggers **nex_evto_b** xxxxxxxxxxxxx1xx Watchpoint #2 triggers **nex_evto_b** xxxxxxxxxxxx1xxx Watchpoint #3 triggers **nex_evto_b** xxxxxxxxxxx1xxxx Watchpoint #4 triggers **nex_evto_b** xxxxxxxxxx1xxxxx Watchpoint #5 triggers **nex_evto_b** xxxxxxxxx1xxxxxx Watchpoint #6 triggers **nex_evto_b** xxxxxxxx1xxxxxxx Watchpoint #7 triggers **nex_evto_b** xxxxxxx1xxxxxxxx Watchpoint #8 triggers **nex_evto_b** xxxxxx1xxxxxxxxx Watchpoint #9 triggers **nex_evto_b** xxxxx1xxxxxxxxxx Watchpoint #10 triggers **nex_evto_b** xxxx1xxxxxxxxxxx Watchpoint #11 triggers **nex_evto_b** xxx1xxxxxxxxxxxx Watchpoint #12 triggers **nex_evto_b** xx1xxxxxxxxxxxxx Watchpoint #13 triggers **nex_evto_b** x1xxxxxxxxxxxxxx Watchpoint #14 triggers **nex_evto_b** 1xxxxxxxxxxxxxxx Watchpoint #15 triggers **nex_evto_b**

1. EOC bits in DC1 must be programmed to trigger $\overline{\text{EVTO}}$ on Watchpoint occurrence for EWC bits to have any effect.

Development Control Register 3 is shown in *Figure 1221* and its fields are described in *Table 1176*.

Offset: Refer to *Table 1171* Access: Read/Write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	EWC													
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1221. Development Control register 3 (DC3)

Table 1176. DC3 Register field descriptions

Field	Description
18:31 EWC	EVTO Watchpoint Configuration(1) 00000000000000 No Watchpoints #16-#29 trigger **nex_evto_b** xxxxxxxxxxxxxxx1 Watchpoint #16 triggers **nex_evto_b** xxxxxxxxxxxx1x Watchpoint #17 triggers **nex_evto_b** xxxxxxxxxxx1xx Watchpoint #18 triggers **nex_evto_b** xxxxxxxxxx1xxx Watchpoint #19 triggers **nex_evto_b** xxxxxxxxx1xxxx Watchpoint #20 triggers **nex_evto_b** xxxxxxxx1xxxxx Watchpoint #21 triggers **nex_evto_b** xxxxxxx1xxxxxx Watchpoint #22 triggers **nex_evto_b** xxxxxx1xxxxxxx Watchpoint #23 triggers **nex_evto_b** xxxxx1xxxxxxxx Watchpoint #24 triggers **nex_evto_b** xxxx1xxxxxxxxx Watchpoint #25 triggers **nex_evto_b** xxx1xxxxxxxxxx Watchpoint #26 triggers **nex_evto_b** xx1xxxxxxxxxxx Watchpoint #27 triggers **nex_evto_b** x1xxxxxxxxxxxx Watchpoint #28 triggers **nex_evto_b** 1xxxxxxxxxxxxx Watchpoint #29 triggers **nex_evto_b**

1. EOC bits in DC1 must be programmed to trigger $\overline{\text{EVTO}}$ on Watchpoint occurrence for EWC bits to have any effect.

61.4.5 Nexus Development Control Register 4 (DC4)

Nexus Development Control Register 4 is used to control mark selection for Program and Data Trace Messaging, as well as masking of events which initiate Program Correlation Messages on the Nexus 3 module.

Development Control Register 4 is shown in *Figure 1222* and its fields are described in *Table 1177*.

Offset: Refer to *Table 1171* Access: Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	PTMARK	DTMARK	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	EVCDM															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1222. Development Control register 4 (DC4)

Table 1177. DC4 Register Field description

Field	Description
0 PTMARK	Program Trace Mark 0 Ignore MSR_{PMM} for masking program trace messages 1 Mask program trace messages when MSR_{PMM}='0', unmask program trace messages when MSR_{PMM}='1'
1 DTMARK	Data Trace Mark 0 Ignore MSR_{PMM} for masking data trace messages 1 Mask data trace messages when MSR_{PMM}='0', unmask data trace messages when MSR_{PMM}='1'
16:31 EVCDM	Event Code (EVCODE) Mask(1) 0000000000000000 No EVCODEs masked for Program Correlation Messages xxxxxxxxxxxxxxx1 EVCODE #0 is masked for Program Correlation Messages xxxxxxxxxxxxxx1x EVCODE #1 is masked for Program Correlation Messages xxxxxxxxxxxxx1xx EVCODE #2 is masked for Program Correlation Messages xxxxxxxxxxxx1xxx EVCODE #3 is masked for Program Correlation Messages xxxxxxxxxxx1xxxx EVCODE #4 is masked for Program Correlation Messages xxxxxxxxxx1xxxxx EVCODE #5 is masked for Program Correlation Messages xxxxxxxxx1xxxxxx EVCODE #6 is masked for Program Correlation Messages xxxxxxxx1xxxxxxx EVCODE #7 is masked for Program Correlation Messages xxxxxxx1xxxxxxxx EVCODE #8 is masked for Program Correlation Messages xxxxxx1xxxxxxxxx EVCODE #9 is masked for Program Correlation Messages xxxxx1xxxxxxxxxx EVCODE #10 is masked for Program Correlation Messages xxxx1xxxxxxxxxxx EVCODE #11 is masked for Program Correlation Messages xxx1xxxxxxxxxxxx EVCODE #12 is masked for Program Correlation Messages xx1xxxxxxxxxxxxx EVCODE #13 is masked for Program Correlation Messages x1xxxxxxxxxxxxxx EVCODE #14 is masked for Program Correlation Messages 1xxxxxxxxxxxxxxx EVCODE #15 is masked for Program Correlation Messages

1. Refer to *Table 1169* for implemented EVCODEs.

61.4.6 Development Status Register (DS)

The Development Status Register is used to report system debug status. When Debug Mode is entered or exited, or an SoC- or core-defined Low Power Mode is entered (see note below), a Debug Status Message is transmitted with DS[31:24]. The external tool can read this register at any time.

Offset: Refer to *Table 1171* Access: Read-only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DBG	LPS			LPC		0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

Figure 1223. Development Status Register (DS)

Table 1178. DS Register field descriptions

Field	Description
0 DBG	CPU Debug Mode Status 0 CPU not in Debug mode 1 CPU in Debug mode (**jd_debug_b** signal asserted)
1:3 LPS	System Low Power Mode Status 000 Normal (Run) mode xx1 DOZE mode (**p_doze** signal asserted) x1x NAP mode (**p_nap** signal asserted) 1xx SLEEP mode (**p_sleep** signal asserted)
4:5 LPC	CPU Low Power Mode Status 00 Normal (Run) mode 01 CPU in Halted state (**p_halted** signal asserted) 10 CPU in Stopped state (**p_stopped** signal asserted) 11 CPU in Waiting state (**p_waiting** signal asserted)

61.4.7 Watchpoint Trigger Registers (WT, PTSTC, PTETC, DTSTC, DTETC)

The Watchpoint Trigger Registers allows the watchpoints defined within the Nexus1 logic to trigger actions. These watchpoints can control Program and/or Data Trace enable and disable. The control bits can be used to produce a related “window” for triggering Trace Messages. Watchpoint trigger register WT is used to control triggering by a single selected watchpoint. The Program Trace Start Trigger Control (PTSTC), Program Trace End Trigger Control (PTETC), Data Trace Start Trigger Control (DTSTC), and Data Trace End Trigger Control (DTETC) are used for extended trigger controls for the respective function. If multiple watchpoints are desired for triggering, or a watchpoint beyond watchpoint #13 is required, then one or more of the extended watchpoint trigger registers may be used. A field

encoding of 4’b1111 in one of the WT register fields enables the corresponding extended trigger register. For all other WT field encodings, the corresponding extended trigger register is disabled and the contents are ignored. Note that direct writes to enable program trace and/or data trace in DC1 will override these controls, and trace will remain enabled until another direct write to DC1 to disable program and/or data trace occurs.

When a start trigger is detected, the designated trace features become enabled, and the corresponding enable bits of the DC1 register are set. Whenever a stop trigger is detected, the designated trace features become disabled, and the corresponding enable bits of the DC1 register are cleared. If the same trigger condition is used for both start and stop triggering, then the designated trace features will toggle between being enabled and disabled at each occurrence of the trigger condition. Similarly, if start and stop triggers for a trace feature occur simultaneously, then the designated trace feature will toggle between enabled and disabled depending on the enable state at the time of the trigger events. For example, if tracing is enabled, and a start and stop trigger occur simultaneously, then tracing will be disabled. Direct writes of the DC1 register take precedence over any trace feature enable state that is derived from watchpoint triggering. A table of watchpoints can be found in the Core (e200z420n3) Core Debug Support chapter.

Offset: Refer to *Table 1171* Access: Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	PTS				PTE				DTS				DTE			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

Figure 1224. Watchpoint Trigger register (WT)

Table 1179 details the Watchpoint Trigger register fields.

Table 1179. WT Register Field descriptions

Field	Description
0:3 PTS	Program Trace Start Control 0000 Trigger disabled 0001 Use Watchpoint #0 0010 Use Watchpoint #1 ... 1101 Use Watchpoint #12 1110 Use Watchpoint #13 1111 Use control settings in the PTSTC register
4:7 PTE	Program Trace End Control 0000 Trigger disabled 0001 Use Watchpoint #0 0010 Use Watchpoint #1 ... 1101 Use Watchpoint #12 1110 Use Watchpoint #13 1111 Use control settings in the PTETC register
8:11 DTS	Data Trace Start Control 0000 Trigger disabled 0001 Use Watchpoint #0 0010 Use Watchpoint #1 ... 1101 Use Watchpoint #12 1110 Use Watchpoint #13 1111 Use control settings in the DTSTC register
12:15 DTE	Data Trace End Control 0000 Trigger disabled 0001 Use Watchpoint #0 0010 Use Watchpoint #1 ... 1101 Use Watchpoint #12 1110 Use Watchpoint #13 1111 Use control settings in the DTETC register

For extended Program Trace start trigger control, the PTSTC register is used.

Offset: Refer to *Table 1171* Access: Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	PTST[0:15]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	PTST[16:31]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1225. Program Trace Start Trigger Control register (PTSTC)

Table 1180 details the PTSTC register fields.

Table 1180. PTSTC Register field descriptions

Field	Description
0:31 PTST	Program Trace Start Trigger Control 00000000000000000000000000000000 Trigger disabled xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1 Use Watchpoint #0 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1x Use Watchpoint #1 xxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xx Use Watchpoint #2 xxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxx Use Watchpoint #3 xxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxx Use Watchpoint #4 xxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxx Use Watchpoint #5 xxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxx Use Watchpoint #6 xxxxxxxxxxxxxxxxxxxxxxxx1xxxxxxx Use Watchpoint #7 xxxxxxxxxxxxxxxxxxxxxxx1xxxxxxxx Use Watchpoint #8 xxxxxxxxxxxxxxxxxxxxxx1xxxxxxxxx Use Watchpoint #9 xxxxxxxxxxxxxxxxxxxxx1xxxxxxxxxx Use Watchpoint #10 xxxxxxxxxxxxxxxxxxxx1xxxxxxxxxxx Use Watchpoint #11 xxxxxxxxxxxxxxxxxxx1xxxxxxxxxxxx Use Watchpoint #12 xxxxxxxxxxxxxxxxxx1xxxxxxxxxxxxx Use Watchpoint #13 xxxxxxxxxxxxxxxxx1xxxxxxxxxxxxxx Use Watchpoint #14 xxxxxxxxxxxxxxxx1xxxxxxxxxxxxxxx Use Watchpoint #15 xxxxxxxxxxxxxxx1xxxxxxxxxxxxxxxx Use Watchpoint #16 xxxxxxxxxxxxxx1xxxxxxxxxxxxxxxxx Use Watchpoint #17 xxxxxxxxxxxxx1xxxxxxxxxxxxxxxxxx Use Watchpoint #18 xxxxxxxxxxxx1xxxxxxxxxxxxxxxxxxx Use Watchpoint #19 xxxxxxxxxxx1xxxxxxxxxxxxxxxxxxxx Use Watchpoint #20 xxxxxxxxxx1xxxxxxxxxxxxxxxxxxxxx Use Watchpoint #21 xxxxxxxxx1xxxxxxxxxxxxxxxxxxxxxx Use Watchpoint #22 xxxxxxxx1xxxxxxxxxxxxxxxxxxxxxxx Use Watchpoint #23 xxxxxxx1xxxxxxxxxxxxxxxxxxxxxxxx Use Watchpoint #24 xxxxxx1xxxxxxxxxxxxxxxxxxxxxxxxx Use Watchpoint #25 xxxxx1xxxxxxxxxxxxxxxxxxxxxxxxxx Use Watchpoint #26 xxxx1xxxxxxxxxxxxxxxxxxxxxxxxxxx Use Watchpoint #27 xxx1xxxxxxxxxxxxxxxxxxxxxxxxxxxx Use Watchpoint #28 xx1xxxxxxxxxxxxxxxxxxxxxxxxxxxxx Use Watchpoint #29 x1xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx Use Watchpoint #30 1xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx Use Watchpoint #31

For extended Program Trace end trigger control, the PTETC register is used.

Offset: Refer to *Table 1171* Access: Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	PTET[0:15]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	PTET[16:31]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1226. Program Trace End Trigger Control register (PTETC)

Table 1181 details the PTETC register fields.

Table 1181. PTETC Register field descriptions

Field	Description
0:31 PTET	Program Trace End Trigger Control 00000000000000000000000000000000 Trigger disabled xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1 Use Watchpoint #0 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1x Use Watchpoint #1 xxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xx Use Watchpoint #2 xxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxx Use Watchpoint #3 xxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxx Use Watchpoint #4 xxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxx Use Watchpoint #5 xxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxx Use Watchpoint #6 xxxxxxxxxxxxxxxxxxxxxxxx1xxxxxxx Use Watchpoint #7 xxxxxxxxxxxxxxxxxxxxxxx1xxxxxxxx Use Watchpoint #8 xxxxxxxxxxxxxxxxxxxxxx1xxxxxxxxx Use Watchpoint #9 xxxxxxxxxxxxxxxxxxxxx1xxxxxxxxxx Use Watchpoint #10 xxxxxxxxxxxxxxxxxxxx1xxxxxxxxxxx Use Watchpoint #11 xxxxxxxxxxxxxxxxxxx1xxxxxxxxxxxx Use Watchpoint #12 xxxxxxxxxxxxxxxxxx1xxxxxxxxxxxxx Use Watchpoint #13 xxxxxxxxxxxxxxxxx1xxxxxxxxxxxxxx Use Watchpoint #14 xxxxxxxxxxxxxxxx1xxxxxxxxxxxxxxx Use Watchpoint #15 xxxxxxxxxxxxxxx1xxxxxxxxxxxxxxxx Use Watchpoint #16 xxxxxxxxxxxxxx1xxxxxxxxxxxxxxxxx Use Watchpoint #17 xxxxxxxxxxxxx1xxxxxxxxxxxxxxxxxx Use Watchpoint #18 xxxxxxxxxxxx1xxxxxxxxxxxxxxxxxx Use Watchpoint #19 xxxxxxxxxxx1xxxxxxxxxxxxxxxxxxx Use Watchpoint #20 xxxxxxxxxx1xxxxxxxxxxxxxxxxxxx Use Watchpoint #21 xxxxxxxxx1xxxxxxxxxxxxxxxxxxxx Use Watchpoint #22 xxxxxxxx1xxxxxxxxxxxxxxxxxxxxx Use Watchpoint #23 xxxxxxx1xxxxxxxxxxxxxxxxxxxxxx Use Watchpoint #24 xxxxxx1xxxxxxxxxxxxxxxxxxxxxxx Use Watchpoint #25 xxxxx1xxxxxxxxxxxxxxxxxxxxxxxx Use Watchpoint #26 xxxx1xxxxxxxxxxxxxxxxxxxxxxxxx Use Watchpoint #27 xxx1xxxxxxxxxxxxxxxxxxxxxxxxxx Use Watchpoint #28 xx1xxxxxxxxxxxxxxxxxxxxxxxxxxx Use Watchpoint #29 x1xxxxxxxxxxxxxxxxxxxxxxxxxxxx Use Watchpoint #30 1xxxxxxxxxxxxxxxxxxxxxxxxxxxxx Use Watchpoint #31

For extended Data Trace start trigger control, the DTSTC register is used.

Offset: Refer to *Table 1171* Access: Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	DTST[2:15]													
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	DTST[16:31]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1227. Data Trace Start Trigger Control register (DTSTC)

The following table details the DTSTC register fields.

Table 1182. DTSTC Register field descriptions

Field	Description
2:31 DTST	Data Trace Start Trigger Control 00000000000000000000000000000000 Trigger disabled xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1 Use Watchpoint #0 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1x Use Watchpoint #1 xxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xx Use Watchpoint #2 xxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxx Use Watchpoint #3 xxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxx Use Watchpoint #4 xxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxx Use Watchpoint #5 xxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxx Use Watchpoint #6 xxxxxxxxxxxxxxxxxxxxxxxx1xxxxxxx Use Watchpoint #7 xxxxxxxxxxxxxxxxxxxxxxx1xxxxxxxx Use Watchpoint #8 xxxxxxxxxxxxxxxxxxxxxx1xxxxxxxxx Use Watchpoint #9 xxxxxxxxxxxxxxxxxxxxx1xxxxxxxxxx Use Watchpoint #10 xxxxxxxxxxxxxxxxxxxx1xxxxxxxxxxx Use Watchpoint #11 xxxxxxxxxxxxxxxxxxx1xxxxxxxxxxxx Use Watchpoint #12 xxxxxxxxxxxxxxxxxx1xxxxxxxxxxxxx Use Watchpoint #13 xxxxxxxxxxxxxxxxx1xxxxxxxxxxxxxx Use Watchpoint #14 xxxxxxxxxxxxxxxx1xxxxxxxxxxxxxxx Use Watchpoint #15 xxxxxxxxxxxxxxx1xxxxxxxxxxxxxxxx Use Watchpoint #16 xxxxxxxxxxxxxx1xxxxxxxxxxxxxxxxx Use Watchpoint #17 xxxxxxxxxxxxx1xxxxxxxxxxxxxxxxxx Use Watchpoint #18 xxxxxxxxxxxx1xxxxxxxxxxxxxxxxxxx Use Watchpoint #19 xxxxxxxxxxx1xxxxxxxxxxxxxxxxxxxx Use Watchpoint #20 xxxxxxxxxx1xxxxxxxxxxxxxxxxxxxxx Use Watchpoint #21 xxxxxxxxx1xxxxxxxxxxxxxxxxxxxxxx Use Watchpoint #22 xxxxxxxx1xxxxxxxxxxxxxxxxxxxxxxx Use Watchpoint #23 xxxxxxx1xxxxxxxxxxxxxxxxxxxxxxxx Use Watchpoint #24 xxxxxx1xxxxxxxxxxxxxxxxxxxxxxxxx Use Watchpoint #25 xxxxx1xxxxxxxxxxxxxxxxxxxxxxxxxx Use Watchpoint #26 xxxx1xxxxxxxxxxxxxxxxxxxxxxxxxxx Use Watchpoint #27 xxx1xxxxxxxxxxxxxxxxxxxxxxxxxxxx Use Watchpoint #28 xx1xxxxxxxxxxxxxxxxxxxxxxxxxxxxx Use Watchpoint #29 x1xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx Use Watchpoint #30 1xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx Use Watchpoint #31

For extended Data Trace end trigger control, the DTETC register is used.

Offset: Refer to *Table 1171* Access: Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	DTET[2:15]													
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	DTET[16:31]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1228. Data Trace End Trigger Control register (DTETC)

The following table details the DTETC register fields.

Table 1183. DTETC Register fields descriptions

Field	Description
2:31 DTET	Data Trace End Trigger Control 00000000000000000000000000000000 Trigger disabled xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1 Use Watchpoint #0 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1x Use Watchpoint #1 xxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xx Use Watchpoint #2 xxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxx Use Watchpoint #3 xxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxx Use Watchpoint #4 xxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxx Use Watchpoint #5 xxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxx Use Watchpoint #6 xxxxxxxxxxxxxxxxxxxxxxxx1xxxxxxx Use Watchpoint #7 xxxxxxxxxxxxxxxxxxxxxxx1xxxxxxxx Use Watchpoint #8 xxxxxxxxxxxxxxxxxxxxxx1xxxxxxxxx Use Watchpoint #9 xxxxxxxxxxxxxxxxxxxxx1xxxxxxxxxx Use Watchpoint #10 xxxxxxxxxxxxxxxxxxxx1xxxxxxxxxxx Use Watchpoint #11 xxxxxxxxxxxxxxxxxxx1xxxxxxxxxxxx Use Watchpoint #12 xxxxxxxxxxxxxxxxxx1xxxxxxxxxxxxx Use Watchpoint #13 xxxxxxxxxxxxxxxxx1xxxxxxxxxxxxxx Use Watchpoint #14 xxxxxxxxxxxxxxxx1xxxxxxxxxxxxxxx Use Watchpoint #15 xxxxxxxxxxxxxxx1xxxxxxxxxxxxxxxx Use Watchpoint #16 xxxxxxxxxxxxxx1xxxxxxxxxxxxxxxxx Use Watchpoint #17 xxxxxxxxxxxxx1xxxxxxxxxxxxxxxxxx Use Watchpoint #18 xxxxxxxxxxxx1xxxxxxxxxxxxxxxxxxx Use Watchpoint #19 xxxxxxxxxxx1xxxxxxxxxxxxxxxxxxxx Use Watchpoint #20 xxxxxxxxxx1xxxxxxxxxxxxxxxxxxxxx Use Watchpoint #21 xxxxxxxxx1xxxxxxxxxxxxxxxxxxxxxx Use Watchpoint #22 xxxxxxxx1xxxxxxxxxxxxxxxxxxxxxxx Use Watchpoint #23 xxxxxxx1xxxxxxxxxxxxxxxxxxxxxxxx Use Watchpoint #24 xxxxxx1xxxxxxxxxxxxxxxxxxxxxxxxx Use Watchpoint #25 xxxxx1xxxxxxxxxxxxxxxxxxxxxxxxxx Use Watchpoint #26 xxxx1xxxxxxxxxxxxxxxxxxxxxxxxxxx Use Watchpoint #27 xxx1xxxxxxxxxxxxxxxxxxxxxxxxxxxx Use Watchpoint #28 xx1xxxxxxxxxxxxxxxxxxxxxxxxxxxxx Use Watchpoint #29 x1xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx Use Watchpoint #30 1xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx Use Watchpoint #31

61.4.8 Nexus Watchpoint Mask Register (WMSK)

The Nexus Watchpoint Mask register controls which watchpoint events are enabled to produce Watchpoint Trace Messages ($DC1_{TM}$ must also be programmed to generate Watchpoint Trace Messages).

Offset: Refer to *Table 1171* Access: Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	WEM[0:15]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	WEM[16:31]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1229. Watchpoint Mask register (WMSK)

Table 1184 details the Watchpoint Trigger register fields.

Table 1184. WMSK Register field descriptions

Field	Description
0:31 WEM	Watchpoint Enable for Messaging 00000000000000000000000000000000 No Watchpoints enabled for Watchpoint Trace Messaging xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1 Watchpoint #0 enabled for WTM xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1x Watchpoint #1 enabled for WTM xxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xx Watchpoint #2 enabled for WTM xxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxx Watchpoint #3 enabled for WTM xxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxx Watchpoint #4 enabled for WTM xxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxx Watchpoint #5 enabled for WTM xxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxx Watchpoint #6 enabled for WTM xxxxxxxxxxxxxxxxxxxxxxxx1xxxxxxx Watchpoint #7 enabled for WTM xxxxxxxxxxxxxxxxxxxxxxx1xxxxxxxx Watchpoint #8 enabled for WTM xxxxxxxxxxxxxxxxxxxxxx1xxxxxxxxx Watchpoint #9 enabled for WTM xxxxxxxxxxxxxxxxxxxxx1xxxxxxxxxx Watchpoint #10 enabled for WTM xxxxxxxxxxxxxxxxxxxx1xxxxxxxxxxx Watchpoint #11 enabled for WTM xxxxxxxxxxxxxxxxxxx1xxxxxxxxxxxx Watchpoint #12 enabled for WTM xxxxxxxxxxxxxxxxxx1xxxxxxxxxxxxx Watchpoint #13 enabled for WTM xxxxxxxxxxxxxxxxx1xxxxxxxxxxxxxx Watchpoint #14 enabled for WTM xxxxxxxxxxxxxxxx1xxxxxxxxxxxxxxx Watchpoint #15 enabled for WTM xxxxxxxxxxxxxxx1xxxxxxxxxxxxxxxx Watchpoint #16 enabled for WTM xxxxxxxxxxxxxx1xxxxxxxxxxxxxxxxx Watchpoint #17 enabled for WTM xxxxxxxxxxxxx1xxxxxxxxxxxxxxxxxx Watchpoint #18 enabled for WTM xxxxxxxxxxxx1xxxxxxxxxxxxxxxxxx Watchpoint #19 enabled for WTM xxxxxxxxxxx1xxxxxxxxxxxxxxxxxxx Watchpoint #20 enabled for WTM xxxxxxxxxx1xxxxxxxxxxxxxxxxxxx Watchpoint #21 enabled for WTM xxxxxxxxx1xxxxxxxxxxxxxxxxxxx Watchpoint #22 enabled for WTM xxxxxxxx1xxxxxxxxxxxxxxxxxxx Watchpoint #23 enabled for WTM xxxxxxx1xxxxxxxxxxxxxxxxxxx Watchpoint #24 enabled for WTM xxxxxx1xxxxxxxxxxxxxxxxxxx Watchpoint #25 enabled for WTM xxxxx1xxxxxxxxxxxxxxxxxxx Watchpoint #26 enabled for WTM xxxx1xxxxxxxxxxxxxxxxxxx Watchpoint #27 enabled for WTM xxx1xxxxxxxxxxxxxxxxxxx Watchpoint #28 enabled for WTM xx1xxxxxxxxxxxxxxxxxxx Watchpoint #29 enabled for WTM x1xxxxxxxxxxxxxxxxxxx Watchpoint #30 enabled for WTM 1xxxxxxxxxxxxxxxxxxx Watchpoint #31 enabled for WTM

61.4.9 Nexus Overrun Control Register (OVCR)

The Nexus Overrun Control register controls Nexus behavior as the internal message queues fill up. Response options include suppressing selected message types, or stalling processor instruction execution.

Offset: Refer to *Table 1171* Access: Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	SPTHOLD		0	0	0	0	0	0	SPEN					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	STTHOLD		0	0	0	0	0	0	0	0	0	0	0	STEN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1230. Nexus Overrun Control Register (OVCR)

Table 1185. OVCR field descriptions

Name	Description
2:3 SPTHOLD	Suppression Threshold 00 Suppression threshold is when message queues are 1/4 full 01 Suppression threshold is when message queues are 1/2 full 10 Suppression threshold is when message queues are 3/4 full 11 Reserved
10:15 SPEN	Suppression Enable 000000 Suppression is disabled xxxxx1 Ownership Trace message suppression is enabled xxxx1x Data Trace message suppression is enabled xxx1xx Program Trace message suppression is enabled xx1xxx Watchpoint Trace message suppression is enabled x1xxxx Reserved 1xxxxx Data Acquisition message suppression is enabled
18:19 STTHOLD	Stall Threshold 00 Stall threshold is when message queues are 1/4 full 01 Stall threshold is when message queues are 1/2 full 10 Stall threshold is when message queues are 3/4 full 11 Reserved
31 STEN	Stall Enable 0 Stalling is disabled 1 Stalling is enabled

61.4.10 Data Trace Control Register (DTC)

The Data Trace Control Register controls whether DTM Messages are restricted to reads, writes, or both for a user programmable address range. In addition, control is provided to restrict data trace message generation to only those load or store accesses that are not stack-related, in order to minimize required data trace message bandwidth.

There are four Data Trace channels controlled by the DTC for the Nexus 3 module. Channels can be programmed to trace data accesses or instruction accesses, but not independently.

Offset: Refer to *Table 1171* Access: Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	RWT1		RWT2		RWT3		RWT4		0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	STDC1	STDC2	STDC3	STDC4	0	0	0	0	RC1	RC2	RC3	RC4	DI	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1231. Data Trace Control register (DTC)

The following table details the Data Trace Control register fields.

Table 1186. DTC Register field descriptions

Field	Description
0:1 RWT1	Read/Write Trace 1 00 No trace enabled X1 Enable Data Read Trace 1X Enable Data Write Trace
2:3 RWT2	Read/Write Trace 2 00 No trace enabled X1 Enable Data Read Trace 1X Enable Data Write Trace
4:5 RWT3	Read/Write Trace 3 00 No trace enabled X1 Enable Data Read Trace 1X Enable Data Write Trace
6:7 RWT4	Read/Write Trace 4 00 No trace enabled x1 Enable Data Read Trace 1x Enable Data Write Trace
16 STDC1	Stack Trace Disable Control 1 0 Tracing of stack accesses are not disabled for range 1 1 Tracing of stack accesses are disabled for range 1; data accesses using GPR R1 in effective address computations are not traced and do not generate data range watchpoint outputs
17 STDC2	Stack Trace Disable Control 2 0 Tracing of stack accesses are not disabled for range 2 1 Tracing of stack accesses are disabled for range 2; data accesses using GPR R1 in effective address computations are not traced and do not generate data range watchpoint outputs

Table 1186. DTC Register field descriptions (continued)

Field	Description
18 STDC3	Stack Trace Disable Control 3 0 Tracing of stack accesses are not disabled for range 3 1 Tracing of stack accesses are disabled for range 3; data accesses using GPR R1 in effective address computations are not traced and do not generate data range watchpoint outputs
19 STDC4	Stack Trace Disable Control 4 0 Tracing of stack accesses are not disabled for range 4 1 Tracing of stack accesses are disabled for range 4; data accesses using GPR R1 in effective address computations are not traced and do not generate data range watchpoint outputs
24 RC1	Range Control 1 0 Condition trace on address within range 1 Condition trace on address outside of range
25 RC2	Range Control 2 0 Condition trace on address within range 1 Condition trace on address outside of range
26 RC3	Range Control 3 0 Condition trace on address within range 1 Condition trace on address outside of range
27 RC4	Range Control 4 0 Condition trace on address within range 1 Condition trace on address outside of range
28 DI	Data Access / Instruction Access Trace 0 Condition trace on data accesses 1 Condition trace on instruction accesses Note: the support for monitoring instruction accesses is not guaranteed to be present in all versions of the Nexus 3 module. Note: the setting of the DI bit also affects the information provided on the data trace port outputs (See the “Data Trace Port Signals” section in the Core (e200z420n3) Core Debug Support chapter.)

61.4.11 Data Trace Start Address Register n (DTSAn)

The Data Trace Start Address Registers define the start addresses for each trace channel. *Figure 1232* shows the format of the DTSAn registers where the counter number, n = 1 to 4.

Offset: Refer to Table 1171 Access: Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	Data Trace Start Address[0:15]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	Data Trace Start Address[16:31]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1232. Data Trace Start Address register n (DTSAn)

61.4.12 Data Trace End Address Register n (DTEAn)

The Data Trace End Address Registers define the end addresses for each trace channel. *Figure 1233* shows the format of the DTEAn registers where the counter number, n = 1 to 4.

Offset: Refer to Table 1171 Access: Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	Data Trace End Address[0:15]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	Data Trace End Address[16:31]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1233. Data Trace End Address register n (DTEAn)

Table 1187 illustrates the range that will be selected for Data Trace for various cases of DTSA being less than, greater than, or equal to DTEA.

Table 1187. Data Trace: Address Range Options

Programmed Values	Range Control Bit Value	Range Selected
DTSA < DTEA	0	DTSA -> <- DTEA
DTSA < DTEA	1	<- DTSA DTEA ->
DTSA > DTEA	N/A	Invalid Range - no trace
DTSA = DTEA	N/A	Invalid Range - no trace

Note: DTSA must be less than DTEA in order to guarantee correct Data Write/Read Traces. Data Trace ranges are inclusive of the DTSA and DTEA addresses for Range Control settings indicating "within range," and are exclusive of the DTSA and DTEA addresses for Range Control settings indicating "outside of range."

Accesses which meet the range and access type qualifiers will cause assertion of a watchpoint output for Ranges 1, 2, and 3. There are three dedicated watchpoint outputs, one for each DTSA/DTEA sets 1, 2, and 3. Range 4 does not provide a watchpoint output. Note that when DTC_{DI}=1, all instruction fetches and prefetches (including discarded prefetches) are monitored, and thus theses range watchpoints differ from the IACx watchpoint outputs, which are not asserted for instructions which are not executed (that is when the instruction prefetch is discarded).

61.4.13 Read/Write Access Control/Status (RWCS)

The Read Write Access Control/Status Register provides control for Read/Write Access. Read/Write access provides DMA-like access to memory-mapped resources on the AHB System bus either while the processor is halted, or during runtime. Control is provided over access type, size, count, and certain bus attributes. The RWCS Register also provides Read/Write Access Status information per *Table 1189*.

Offset: Refer to *Table 1171* Access: Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	AC	RW	SZ			MAP			PR		ATTR				0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	CNT														ERR	DV
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1234. Read/Write Access Control/Status register (RWCS)

Table 1188. RWCS register field descriptions

Field	Description
0 AC	Access Control 0 End access/ Access has completed 1 Start access
1 RW	Read/Write Select 0 Read access 1 Write access
2:4 SZ	Word Size 000 8-bit (byte) 001 16-bit (half-word) 010 32-bit (word) 011 64-bit (doubleword, requires two passes through RWD) 100 Reserved (default to word) 101 Reserved (default to word) 110 Reserved (default to word) 111 Reserved (default to word)

Table 1188. RWCS register field descriptions (continued)

Field	Description
5:7 MAP	MAP Select 000 Primary memory map 001 Reserved 010 Reserved ... 110 Reserved 111 Reserved
8:9 PR(1)	Read/Write Access Priority 00 Reserved (default to highest priority) 01 Reserved (default to highest priority) 10 Reserved (default to highest priority) 11 Highest access priority
10:13 ATTR	Access Attributes 0xxx Reserved 1xxx Reserved x0xx **p_d_hprot[4]** driven to 0 for accesses x1xx **p_d_hprot[4]** driven to 1 for accesses xx0x **p_d_hprot[3]** driven to 0 for accesses xx1x **p_d_hprot[3]** driven to 1 for accesses xxx0 **p_d_hprot[2]** driven to 0 for accesses xxx1 **p_d_hprot[2]** driven to 1 for accesses
16:29 CNT	Access Control Count 0xXXXX Number of accesses of word size SZ
30 ERR	Read/Write Access Error (see *Table 1189*)
31 DV	Read/Write Access Data Valid (see *Table 1189*)

1. The priority functionality is not currently implemented.

Table 1189. Read/Write Access Status Bit Encoding

Read Action	Write Action	ERR	DV
Read Access has not completed	Write Access completed without error	0	0
Read Access error has occurred	Write Access error has occurred	1	0
Read Access completed without error	Write Access has not completed	0	1
Not Allowed	Not allowed	1	1

61.4.14 Read/Write Access Data (RWD)

The Read/Write Access Data Register (RWD) provides the data to/from system bus memory-mapped locations when initiating a read or a write access.

Offset: Refer to *Table 1171* Access: Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	Read/Write Data[0:15]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	Read/Write Data[16:31]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1235. Read/Write Access Data register (RWD)

Read/Write accesses to the AHB require that the debug firmware properly retrieve/place the data in the RWD. *Table 1190* shows the proper placement of data into the RWD. Note that doubleword transfers require two passes through RWD.

Table 1190. RWD data placement for Transfers

Transfer Size and byte offset	RWA(2:0)	RWCS[SZ]	RWD(1)			
			31:24	23:16	15:8	7:0
Byte	x x x	0 0 0	—	—	—	X
Half	x x 0	0 0 1	—	—	X	X
Word	x 0 0	0 1 0	X	X	X	X
Doubleword						
first RWD pass (low order data)	0 0 0	0 1 1	X	X	X	X
second RWD pass (high order data)			X	X	X	X

1. “X” indicates byte lanes with valid data
“—” indicates byte lanes which will contain unused data.

Table 1191 shows the mapping of RWD bytes to byte lanes of the AHB read and write data buses.

Table 1191. RWD byte lane mapping

Transfer Size and byte offset	RWA(2:0)	RWD(1)			
		31:24	23:16	15:8	7:0
Byte @000	0 0 0	—	—	—	AHB[7:0]
Byte @001	0 0 1	—	—	—	AHB[15:8]
Byte @010	0 1 0	—	—	—	AHB[23:16]
Byte @011	0 1 1	—	—	—	AHB[31:24]
Byte @100	1 0 0	—	—	—	AHB[39:32]
Byte @101	1 0 1	—	—	—	AHB[47:40]

Table 1191. RWD byte lane mapping (continued)

Transfer Size and byte offset	RWA(2:0)	RWD(1)			
		31:24	23:16	15:8	7:0
Byte @110	1 1 0	—	—	—	AHB[55:48]
Byte @111	1 1 1	—	—	—	AHB[63:56]
Half @000	0 0 0	—	—	AHB[15:8]	AHB[7:0]
Half @010	0 1 0	—	—	AHB[31:24]	AHB[23:16]
Half @100	1 0 0	—	—	AHB[47:40]	AHB[39:32]
Half @110	1 1 0	—	—	AHB[63:56]	AHB[55:48]
Word @000	0 0 0	AHB[31:24]	AHB[23:16]	AHB[15:8]	AHB[7:0]
Word @100	1 0 0	AHB[63:56]	AHB[55:48]	AHB[47:40]	AHB[39:32]
Doubleword @000 first RWD pass second RWD pass	0 0 0	AHB[31:24] AHB[63:56]	AHB[23:16] AHB[55:48]	AHB[15:8] AHB[47:40]	AHB[7:0] AHB[39:32]

1. "-" indicates byte lanes which will contain unused data.

61.4.15 Read/Write Access Address (RWA)

The Read/Write Access Address Register provides the system bus address to be accessed when initiating a read or a write access.

Offset: Refer to *Table 1171* Access: Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	Read/Write Address[0:15]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	Read/Write Address[16:31]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1236. Read/Write Access Address register (RWA)

61.5 Nexus 3 Register Access via JTAG/OnCE

Access to Nexus 3 register resources is enabled by loading a single instruction ("*NEXUS3-ACCESS*") into the JTAG Instruction Register (IR) (OnCE OCMD register). For the Nexus 3 block, the OCMD value is 0b0001111100.

Once the NEXUS3-ACCESS instruction has been loaded, the JTAG/OnCE port allows tool/target communications with all Nexus 3 registers according to the register map in *Table 1171*.

Reading/writing of a Nexus 3 register then requires two (2) passes through the Data-Scan (DR) path of the JTAG state machine. (See *Section 61.23*.)

1. The first pass through the DR selects the Nexus 3 register to be accessed by providing an index (see *Table 1171*), and the direction (read/write). This is achieved by loading an 8-bit value into the JTAG Data Register (DR). This register has the following format:

Figure 1237. DR register format

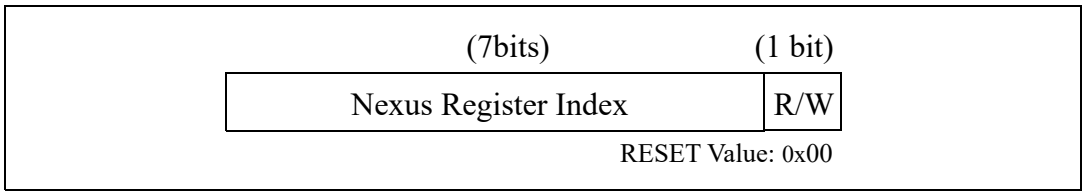


Table 1192. Nexus register index

Nexus Register Index	Selected from values in *Table 1171*
Read/Write (R/W):	0 Read 1 Write

2. The second pass through the DR then shifts the data in or out of the JTAG port, LSB first.
   a) During a read access, data is latched from the selected Nexus register when the JTAG state machine passes through the “Capture-DR” state.
   b) During a write access, data is latched into the selected Nexus register when the JTAG state machine passes through the “Update-DR” state.

61.6 Nexus 3 Register Access via Software

Access to most Nexus 3 register resources by software is supported by mapping the Nexus 3 registers to privileged DCRs (device control registers) which are accessible via the **mfdcr** and **mtdcr** instructions. It is intended that these access only occur when no possible conflicts with hardware-based accesses are possible. A conflict between hardware and software accesses will produce undefined results. *Table 1171* shows the mappings of Nexus 3 registers to DCR numbers. Software writes to Nexus 3 register resources are blocked when the **nex_sfwcntl_en** input signal is negated, without signaling an exception. Note that the Nexus 3 Read/Write Access functionality is not available to software, since software can use load and store instructions to access memory.

61.7 Nexus Message Fields

Nexus messages are comprised of fields. Each field contains a distinct piece of information within a message, and each message contains multiple fields. Messages are transferred in packets over the Auxiliary Output protocol. A packet is a collection of fields. A packet may contain any number of fixed length fields, but may contain at most one variable length field. The variable length field must be the last field in a packet. The following subsections describe a subset of the message field types.

61.7.1 TCODE Field

The TCODE field is a 6-bit fixed length field that identifies the type of message and its format. The field encodings are assigned by IEEE-ISTO 5001.

61.7.2 Source ID Field (SRC)

Each Nexus module in a device is identified by a unique Client Source Identification Number. The number assigned to each Nexus module is determined by the SoC integrator, and is provided on the **nex3_ext_src_id[0:3]** input signals. Multi-threaded processors may assign additional source ID information to indicate which thread a message is associated with. The Nexus 3 module implements a 4-bit fixed length Source ID field consisting of a Client Source ID.

61.7.3 Relative Address Field (U-ADDR)

The non-sync forms of the Program and Data Trace messages include addresses which are relative to the address that was transmitted in the previous Program or Data Trace message respectively. The relative address format is compliant with IEEE-ISTO 5001 and is designed to reduce the number of bits transmitted for address fields.

The relative address is generated by XORing the new address with the previous, and then using only the results up to the most significant ‘1’. To recreate the original address, the relative address is XORed with the previously decoded address.

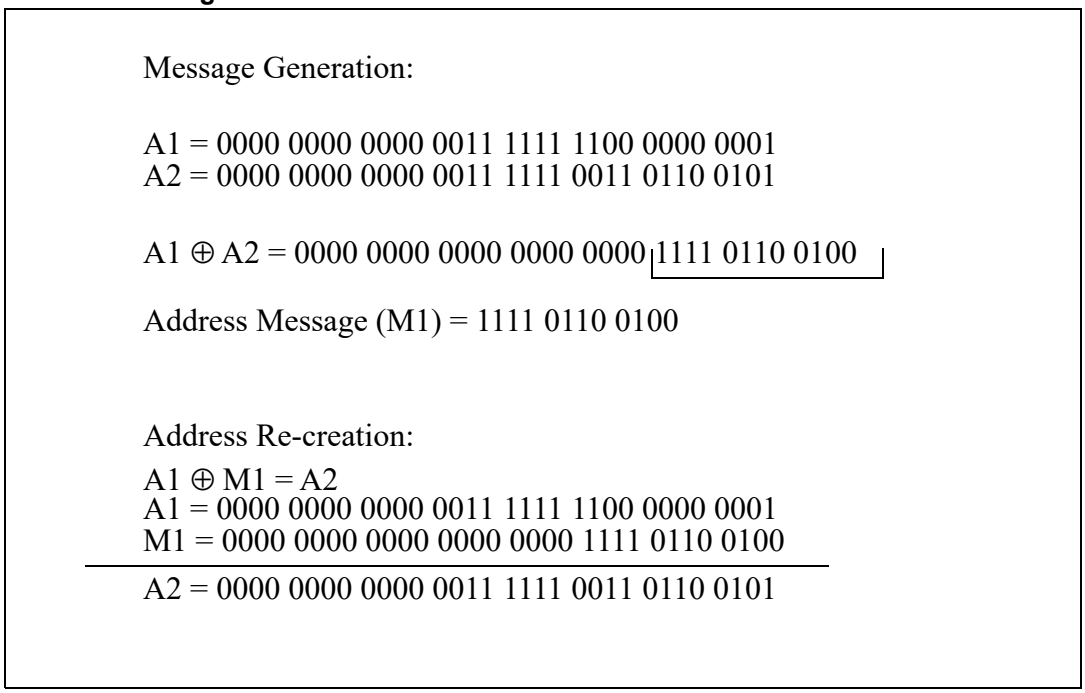
The relative address of a Program Trace message is calculated with respect to the previous Program Trace message, regardless of any address information that may have been sent in any other trace messages in the interim between the two Program Trace messages.

The relative address of a Data Trace message is calculated with respect to the previous Data Trace message, regardless of any address information that may have been sent in any other trace messages in the interim between the two Data Trace messages.

Program trace messages also provide the execution mode status in the least significant bit of the **reconstructed** address field. A value of ‘0’ indicates that preceding instruction count and history information should be interpreted in a non-VLE context. A value of ‘1’ indicates that the preceding instruction count and history information should be interpreted in a VLE context.

Previous Address (A1) =0x0003FC01, New Address (A2) = 0x0003F365

Figure 1238. Relative Address Generation and Re-creation



61.7.4 Full Address Field (F-ADDR)

Program Trace synchronization messages provide the full address associated with the trace event (leading zeroes may be truncated) with the intent of providing a reference point for development tools to operate from when reconstructing relative addresses. Synchronization messages are generated at significant mode switches and are also generated periodically to ensure that development tools are guaranteed to have a reference address given a sufficiently large sample of trace messages. Program trace messages also provide the execution mode status in the least significant bit of the reconstructed address field. A value of ‘0’ indicates that preceding instruction count and history information should be interpreted in a non-VLE context. A value of ‘1’ indicates that the preceding instruction count and history information should be interpreted in a VLE context.

61.8 Nexus Message Queues

The Nexus 3 module implements internal message queues capable of storing up to three messages per cycle into a small initial queue which then fills a larger queue at up to two messages per cycle. Messages that enter the queues are transmitted in the order in which they are received.

If more than three messages attempt to enter the queue in the same cycle, the highest priority messages are stored and the remaining message(s) will be dropped due to a collision. Collision events are expected to be rare.

The Overrun Control register (OVCR) controls the Nexus behavior as the message queue fills. The Nexus block may be programmed to:

- Allow the queue to overflow, drain the contents, queue an overrun error message and resume tracing.
- Stall the processor when the queue utilization reaches the selected threshold.
- Suppress selected message types when the queue utilization reaches the selected threshold.

61.8.1 Message Queue Overrun

In this mode, the message queue will stop accepting messages when an overrun condition is detected. The contents of the queues will be allowed to drain until empty. Incoming messages are discarded until the queue is emptied. Once empty, an overrun error message is enqueued which contains information about the types of messages that were discarded due to the overrun condition.

61.8.2 CPU Stall

In this mode, processor instruction issue is stalled when the queue utilization reaches the selected threshold. The processor is stalled long enough drop one threshold level below the level which triggered the stall. For example, if stalling the processor is triggered at 1/4 full, the stall will stay in effect until the queue utilization drops to empty. There may be significant skid from the time that the stall request is made until the processor is able to stop completing instructions. This skid should be taken into consideration when programming the threshold. Refer to *Section 61.4.9* for complete programming options.

61.8.3 Message Suppression

In this mode, the message queue will disable selected message types when the queue utilization reaches the selected threshold. This allows lower bandwidth tracing to continue and possibly avoid an overrun condition. If an overrun condition occurs despite this message suppression, the queue will respond according to the behavior described in *Section 61.8.1*. Suppressed message types are reported in the error message if they occur after overrun in addition to other discarded message types. Once triggered, message suppression will remain in effect until queue utilization drops to the threshold below the level selected to trigger suppression.

61.8.4 Nexus Message Priority

Nexus messages may be lost due to contention with other message types under the following circumstances:

- More than three messages are generated in the same cycle

Table 1193 lists the various message types and their relative priority from highest to lowest.

Up to three message requests can be queued into the message buffer in a given cycle. If more than three message requests exist in a given cycle, the three highest priority message classes are queued into the message buffer. The remaining messages which did not successfully queue into the message buffer in that cycle will generate subsequent responses as detailed in *Table 1193*.

The CPU is capable of completing two instructions per cycle. If multiple trace messages need to be queued at the same time, they will be queued with the following priority:

Instruction 0 (oldest instruction) (WPM ->DQM -> PCM_{PIDMSG} -> OTM -> BTM -> DTM)-> Instruction1 (newer instruction) (WPM -> DQM -> OTM -> BTM -> DTM). Up to three messages may be simultaneously queued. Note that for the cycle following a dropped PTM, non-periodic OTM, or DQM message, only two other messages may be queued in addition to the dropped Error Message.

Watchpoint Messages from instructions which complete at the same time or events that occur during the same cycle will be combined.

Table 1193. Message Type Priority and Message Dropped Responses

Message Type	Message	Priority	Message Dropped Response
Error	Error	0 (highest)	N/A(1)
WP(Watchpoint Trace)	WPM (Watchpoint Message)	1	N/A(1)
DQ(Data Acquisition)	DQM (Data Acquisition Message)	2	DQM Error Message
Program Trace (PID MSG)	PCM: PID/NPIDR update (Program Correlation Message)	2	OTM Error Message
OT(Ownership)	OTM: PID/NPIDR update (Ownership Trace Message)	2	OTM Error Message(2)
Program Trace	BTM (Branch Trace Message)	2	BTM Error Message Sync upgrade next BTM
	Sync (Program Sync Message)	3	Sync upgrade next BTM
	RFM (Resource Full for Instruction counter or history buffer)	4	BTM Error Message Sync upgrade next BTM
	DS (Debug Status Message)	5	Sync upgrade next BTM
	PCM (Program Correlation Message)	6	BTM Error Message Sync upgrade next BTM
DT (Data Trace)	DTM (Data Trace Message)	7	Sync upgrade next DTM
OT (Ownership)	OTM - Periodic update (Ownership Trace Message)	8 (lowest)	none

1. Error and Watchpoint messages are not dropped due to collisions, due to their priority.
2. Message will always be dropped if program trace is enabled, and program correlation messages for PID messages are not masked (Event Code = 0101). No error message is sent for this case since the PID value is contained in the higher priority message.

61.8.5 Data Acquisition Message Priority Loss Response

If a Data Acquisition Message (DQM) loses arbitration due to contention with higher priority messages, an error message will be generated to indicate that a DQM has been lost due to contention.

61.8.6 Ownership Trace Message Priority Loss Response

If an Ownership Trace message (OTM) due to software updates to the Process ID state loses arbitration due to contention with higher priority messages other than a program correlation message with EVCODE = 0101 (PID update), an error message will be generated to indicate that a OTM has been lost due to contention. If the pending OTM is a periodic update, the event is dropped without generating an error message.

61.8.7 Program Trace Message Priority Loss Response

If a Program Trace message (PTM) loses arbitration due to contention with higher priority messages, and the discarded PTM is a Program Correlation message, a Resource Full message for instruction count or history buffer, or a Branch Trace message, then an Error message is generated to indicate that branch trace information has been lost, and the next Branch Trace message will be upgraded to a sync-type message.

If the discarded PTM is a Program Correlation message with PID information (EVCODE=0101), the Error message will indicate a dropped OTM and a dropped Program Trace (Error code = xxxx11xx).

61.8.8 Program Trace Sync Message Priority Loss Response

If a Program Trace Sync message (SYNC) loses arbitration due to contention with higher priority messages, the Sync event is discarded, and the next Branch Trace message will be upgraded to a sync-type message.

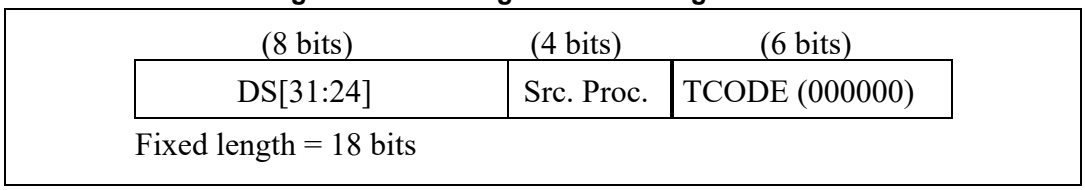
61.8.9 Data Trace Message Priority Loss Response

If a Data Trace message (DTM) loses arbitration due to contention with higher priority messages, the DTM event is discarded, and the next DTM will be upgraded to a sync-type message.

61.9 Debug Status Messages

Debug Status Messages report low power mode and debug status. Debug Status Messages are enabled when Nexus 3 is enabled. Entering/exiting Debug Mode as well as entering, exiting, or changing Low Power Mode(s) will trigger a Debug Status Message, indicating the value of the most significant byte in the Development Status register. Debug status information is sent out in the following format:

Figure 1239. Debug Status Message Format



61.10 Error Messages

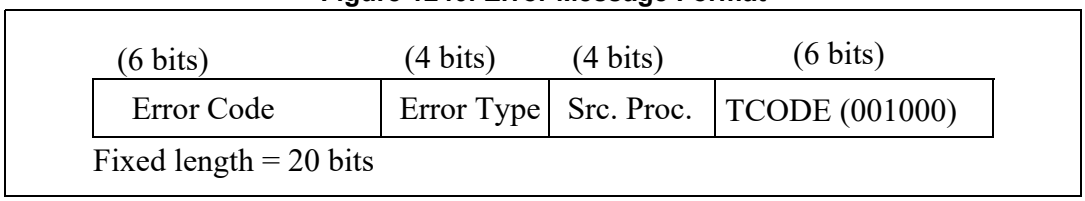
Error messages are enabled whenever the debug logic is enabled. There are two conditions that will produce an error message, each receiving a separate error type designation:

- A message is discarded due to contention with other (higher priority) message types. These errors will have an Error Type value of 1.
- The message queue overruns. After the queue is drained, an error message is enqueued with an error code that indicates what types of messages were discarded during the interim. These errors will have an Error Type value of 0.

Note: The OVCR Register can be used in order to alleviate potential overrun situations.

Error information is messaged out in the following format (also see *Table 1166* and *Table 1167*):

Figure 1240. Error Message Format



61.11 Ownership Trace

This section details the ownership trace features of the Nexus 3 module.

61.11.1 Overview

Ownership trace provides a macroscopic view, such as task flow reconstruction, when debugging software written in a high level (or object-oriented) language. It offers the highest level of abstraction for tracking operating system software execution. This is especially useful when the developer is not interested in debugging at lower levels.

61.11.2 Ownership Trace Messaging (OTM)

Ownership trace information is messaged via the auxiliary port using an Ownership Trace Message (OTM). Core (e200z420n3) processors contain a <asisitalic>PowerISA 2.06 defined “Process ID” register within the CPU. It is updated by the operating system software to provide task/process ID information. The contents of this register are replicated on the pins of the processor and connected to Nexus. The Process ID (PID) register value can be accessed using the **mfspr**/**mtspr** instructions.

Note: The CPU includes a Process ID register (PID0), thus the Nexus UBA functionality is not implemented.

In addition, to decouple trace information from the PID0 register and to provide a full independent 32-bit process ID for debug use, the Nexus PID Register (NPIDR) may be used instead of PID0 to provide OTM process ID information. It is updated by the operating system software to provide task/process ID information. The Nexus Process ID (NPIDR) register value can be accessed using the **mfspr**/**mtspr** instructions. This register is accessible in user or supervisor mode.

Offset: Refer to SPR - 517 Access: Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	Nexus 3 Process ID[0:15]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	Nexus 3 Process ID[16:31]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1241. Nexus 3 Process ID Register (NPIDR)

Note: *Note: OS updates to NPIDR should be performed in addition to normal PID0 updates when a process switch occurs, in order to properly generate OTM messages with new process ID information when NPIDR is selected for OTM use.*

The process ID source (PID0 or NPIDR) is selected by the setting of the $DC1_{OTS}$ control bit.

There are two conditions which will cause an Ownership Trace Message when Ownership Trace is enabled:

- When the PID0 register is written to by the processor and $DC1_{OTS}$ indicates PID0 should be used, or when the NPIDR register is written to by the processor and $DC1_{OTS}$ indicates NPIDR should be used, the data is latched within Nexus, and is messaged out via the auxiliary port, allowing development tools to trace ownership flow. However, if Program Trace is enabled, and program correlation messages for PID updates are not masked (Event Code = 0101), then an OTM will not be generated for an update to the selected process ID register, since the program correlation message will provide this process ID update information.
- Periodically, at least once every 256 messages, the most recent state of the selected process ID register is messaged out. The resulting Ownership Trace message will indicate in the PID Index subfield that PID0/NPIDR status is being reported and the most recent value of the PID0/NPIDR register will be conveyed in the Process ID value subfield. These periodic Ownership Trace message events can be disabled by setting $DC1_{POTD}$.

Ownership trace information is messaged out in the following format:

Figure 1242. Ownership Trace Message Format

(1-32 bits)	(4 bits)	(4 bits)	(6 bits)
Process ID	PID Index (0000)	Src. Proc.	TCODE (000010)

Variable length = 15-46 bits

61.12 Program Trace

This section details the program trace mechanism supported by Nexus3 for the e200z420n3 processor. Program trace is implemented via Branch Trace Messaging (BTM) as per the

IEEE-ISTO 5001 standard definition. Branch Trace Messaging for Core (e200z420n3) processors is accomplished by snooping the internal address bus (between the CPU and Cache), attribute signals, and CPU Status (**p_mode[0:3]**, **p_pstat_pipe{0,1}[0:5]**).

61.12.1 Branch Trace Messaging Types

Traditional Branch Trace Messaging facilitates program trace by providing the following types of information:

- Messaging for taken direct branches includes how many sequential instructions were executed since the last taken branch or exception, including the taken direct branch. Branch instructions are included in the count of sequential instructions.
- Messaging for taken indirect branches and exceptions includes how many sequential instructions were executed since the last taken branch or exception and the unique portion of the branch target address or exception vector address. Branch instructions are included in the count of sequential instructions. For taken indirect branches which trigger generation of a message, the branch is also included in the count.

Branch History Messaging facilitates program trace by providing the following information.

- Messaging for taken indirect branches and exceptions includes a) how many sequential instructions (I-CNT) were executed since the last predicate instruction, taken/not taken direct branch, taken/not-taken indirect branch, or exception, b) the unique portion of the branch target address or exception vector address, and c) a branch/predicate instruction history field. Each bit in the history field represents a direct branch or predicated instruction where a value of one (1) indicates taken, and a value of zero (0) indicates not taken. Not-taken indirect branches will generate a history bit with a value of zero (0). Instructions that generate history bits are not included in instruction counts. For taken indirect branches which trigger generation of this message type, the branch is included in the count, but not in the history field.

61.12.1.1 Indirect Branch Message Instructions

Table 1194 shows the types of instructions and events which cause Indirect Branch Messages or Branch History Messages to be encoded.

Table 1194. Indirect Branch Message Sources

Source of Indirect Branch Message	Instructions / Detail
Taken branch relative to a register value	**bcctr**, **bcctrl**, **bclr**, **bclrl, se_bctr, se_bctrl, se_blr, se_blrl**
System Call / Trap exceptions taken	**se_sc, tw**
Return from interrupts / exceptions	**se_rfi, se_rfci, se_rfdi**
Exit from reset with Program Trace Enabled	Indirect branch with Sync, target address is initial instruction, count=1

61.12.1.2 Direct Branch Message Instructions

Table 1195 shows the types of instructions which will cause Direct Branch Messages or will toggle a bit in the instruction history buffer to be messaged out in a Resource Full Message or Branch History Message.

Table 1195. Direct Branch Message Sources

Source of Direct Branch Message	Instructions
Taken direct branch instructions Instruction Synchronize	**b**, **ba**, **bl**, **bla**, **bc**, **bca**, **bcl**, **bcla, se_b. se_bc, se_bl, e_b, e_bc, e_bl, e_bcl, se_isync**

61.12.1.3 BTM using Branch History Messages

Traditional BTM Messaging can accurately track the number of sequential instructions between branches, but cannot accurately indicate which instructions were conditionally executed, and which were not.

Branch History Messaging solves this problem by providing a predicated instruction history field in each Indirect Branch Message. Each bit in the history represents a predicated instruction or direct branch, or a not-taken indirect branch. A value of one (1) indicates the conditional instruction was executed or the direct branch was taken. A value of zero (0) indicates the conditional instruction was not executed or the branch was not taken.

Branch History Messages solve predicated instruction tracking and save bandwidth since only indirect branches cause messages to be queued.

61.12.1.4 BTM using Traditional Program Trace Messages

Based on the PTM bit in the DC1 Register, Program Tracing can utilize either Branch History Messages (PTM=1) or traditional Direct/Indirect Branch Messages (PTM=0).

Branch History will save bandwidth and keep consistency between methods of Program Trace, yet may lose temporal order between BTM messages and other types of messages. Since direct branches are not messaged, but are instead included in the history field of the Indirect Branch History Message, other types of messages may enter the FIFO between Branch History Messages. The development tool cannot determine the ordering of “events” that occurred with respect to direct branches simply by the order in which messages are sent out.

Traditional BTM messages maintain their temporal ordering because each event that can cause a message to be queued will enter the FIFO in the order it occurred and will be messaged out maintaining that order.

61.12.2 BTM Message Formats

The Nexus 3 block supports three types of traditional BTM Messages - Direct, Indirect, and Synchronization Messages. It supports two types of branch history BTM Messages - Indirect Branch History, and Indirect Branch History with Synchronization Messages.

61.12.2.1 Indirect Branch Messages (History)

Indirect branches include all taken branches whose destination is determined at run time, interrupts, and exceptions. If $DC1_{PTM}$ is set to ‘1’, indirect branch information is messaged out in the following format:

Figure 1243. Indirect Branch Message (History) Format

(1-32 bits)	(1–32 bits)	(1–8 bits)	(1 bit)	(4 bits)	(6 bits)
Branch History	Relative Address	Sequence Count	(0)	Source Proc.	TCODE (011100)

Max length = 83 bits; Min length = 14 bits

61.12.2.2 Indirect Branch Messages (Traditional)

If $DC1_{PTM}$ is cleared to '0', indirect branch information is messaged out in the following format:

Figure 1244. Indirect Branch Message Format

(1-32 bits)	(1-8 bits)	(1 bit)	(4 bits)	(6 bits)
Relative Address	Sequence Count	(0)	Source Proc.	TCODE (000100)

Max length = 51 bits; Min length = 13 bits

61.12.2.3 Direct Branch Messages (Traditional)

Direct branches (conditional or unconditional) are all taken branches whose destination is fixed in the instruction opcode. Direct branch information is messaged out in the following format:

Figure 1245. Direct Branch Message Format

(1-8 bits)	(4 bits)	(6 bits)
Sequence Count	Src. Proc.	TCODE (000011)

Max length = 18 bits; Min length = 11bits

Note: When $DC1_{PTM}$ is set, Direct Branch Messages will not be transmitted. Instead, each direct branch, not-taken indirect branch, or predicated instruction will be recorded in the history buffer.

61.12.3 Program Trace Message Fields

The following subsections describe specific fields used for Program Trace messages.

61.12.3.1 Sequential Instruction Count Field (ICNT)

Most of the program trace messages include an instruction count field. For traditional Branch Messages, ICNT represents the number of sequential instructions including non-taken branches since the last Direct/Indirect Branch Messages. Branch instructions which trigger message generation are included in the ICNT.

For Branch History Messages, ICNT represents the number of instructions executed since the last taken/non-taken direct branch, predicate instruction, last taken/not-taken indirect branch, or exception. Branch instructions which trigger message generation are included in the ICNT. Instructions which generate history bits are not included in the ICNT.

The sequential instruction counter overflows after its value reaches 255 and is reset to 0. In addition, the next BTM Message (corresponding to the 256th or later instruction) will be converted to a w/sync type message.

The instruction counter is reset every time the instruction count is transmitted in a message or whenever there is a branch/predicate history event, as well as on exiting from debug mode.

61.12.3.2 Branch/Predicate Instruction History (HIST)

If $DC1_{PTM}$ is set, BTM messaging will use the Branch History format. The branch history (HIST) field in these messages provides a history of branch execution used for reconstructing the program flow. The branch/predicate history buffer stores information about branch and predicate instruction execution. The buffer is implemented as a left-shifting register. The buffer is preloaded with a one (1) which acts as a stop bit (the most significant 1 in the history field is a termination bit for the field). The pre-loaded bit itself is not part of the history, but is transmitted with the packet.

A value of one (1) is shifted into the history buffer for each taken direct branch (program counter relative branch) or predicate instruction whose condition evaluates to true. A value of zero (0) is shifted into the history buffer for each not-taken branch (including indirect branch instructions) or predicate instruction whose condition evaluates to false.

This history buffer information is transmitted as part of an Indirect Branch with History message, as part of a Program Correlation message, or as part of a Resource Full message if the history buffer becomes full. The history buffer is reset every time the history information is transmitted in a message, as well as on exiting from debug mode.

Table 1196. Branch/Predicate History Events

Branch/Predicate History Event	History Bit(s)	Relevant Instructions
Not taken register indirect branches	0	bcctr, bcctrl, bclr, bclrl
Not taken direct branches	0	b, ba, bc, bca, bla, bcla, bl, bcl
Taken direct branches	1	**b, ba, bc, bca, bla, bcla, bl, bcl**(1)

1. If the EVCODE for direct branch function calls is not masked in DC4, taken **bl** and **bcl** instructions will generate Program Correlation Messages and will not be logged in the history buffer.

61.12.3.3 Execution Mode Indication

In order for a development tool to properly interpret instruction count and history information, it must be aware of the execution mode context of that information. VLE instructions will be interpreted differently from non-VLE instructions.

Program trace messages provide the execution mode status in the least significant bit of the **reconstructed** address field. A value of ‘0’ indicates that preceding instruction count and history information should be interpreted in a non-VLE context. A value of ‘1’ indicates that the preceding instruction count and history information should be interpreted in a VLE context.

61.12.4 Resource Full Messages

The Resource Full Message is used in conjunction with Branch Trace and Branch History Messages. The Resource Full Message is generated when either the internal branch/predicate history buffer is full, or if the BTM Instruction sequence counter (I-CNT) overflows. If synchronization is needed at the time this message is generated, the synchronization is delayed until the next Branch Trace Message that is not a Resource Full Message.

For history buffer overflow, the Resource Full Message transmits a Resource Code (RCODE) of 0b0001 and the current contents of the history buffer, including the stop bit, are transmitted in the Resource Data (RDATA) field. This history information can be concatenated by the development tool with the branch/predicate history information from subsequent messages to obtain the complete branch/predicate history between indirect changes of flow.

For instruction counter overflow, the Resource Full Message transmits an RCODE of 0b0000 and a value of 0xFF is transmitted in the RDATA field, indicating that 255 sequential instructions have been executed since the last change of flow or, if program trace is in history mode, since the last instruction which recorded history information.

Figure 1246. Resource Full Message Format

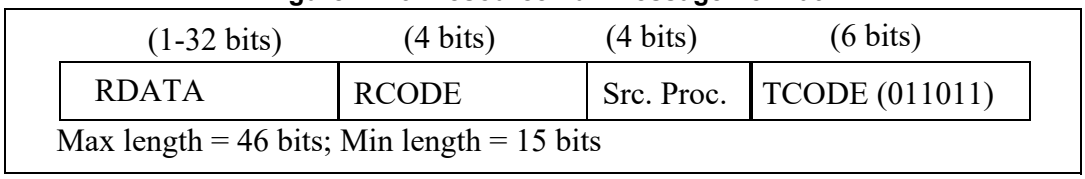


Table 1197 shows the RCODE encodings and RDATA information used for Resource Full messages.

Table 1197. RCODE Encoding

RCODE	Description	RDATA field
0000	Program Trace Instruction counter reached 255 and was reset.	0xFF
0001	Program Trace, Branch / Predicate Instruction History full.	Branch HIstory. This type of packet is terminated by a stop bit set to 1 after the last history bit.

61.12.5 Program Correlation Messages

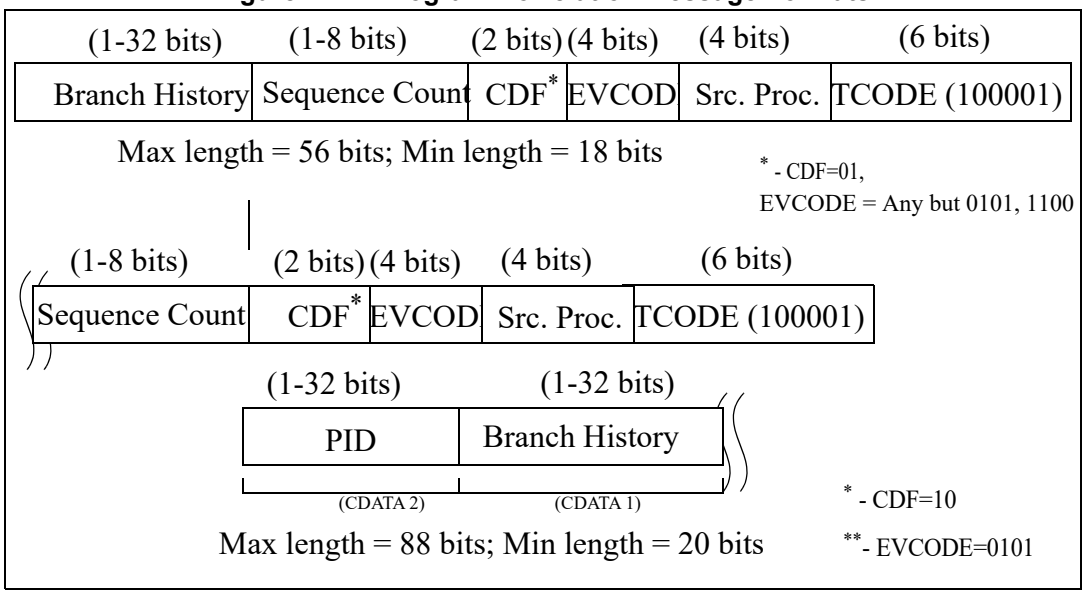
Program Correlation Messages (PCMs) are used to correlate events to the program flow that may or may not be associated with the instruction stream. The following events will result in a PCM when program trace is enabled:

- When the CPU enters debug mode, a PCM is generated. The instruction count and history information provided by the PCM can be used to determine the last sequence of instructions executed prior to debug mode entry.
- When the CPU first enters a low power mode in which instructions are no longer executed, a PCM is generated. The instruction count and history information provided by the PCM can be used to determine the last sequence of instructions executed prior to low power mode entry.
- Whenever program trace is disabled by any means, a PCM is generated. The instruction count and history information provided by the PCM can be used to determine the last sequence of instructions executed prior to disabling program trace.
- When a “Branch and Link” instruction executes (direct branch function call - **bl**/**bcl**/**bla**/**bcla**-type instructions)
- When program trace becomes masked due to MSR_{PMM}=‘0’ and $DC4_{PTMARK}$=‘1’.
- When a write to the process ID register selected by $DC1_{OTS}$ (PID0 or NPIDR) is made via a **mtspr** PID0 or **mtspr** NPIDR.

Refer to *Table 1169* for the event codes that are supported in this implementation. Event code masking is available via the EVCDM field of the DC4 register to allow for control over generation of Program Correlation messages for each event type.

Program Correlation is messaged out in the following formats:

Figure 1247. Program Correlation Message Formats



61.12.5.1 Program Correlation Message Generation for PID Updates

When a (potentially) new value is established in the selected process ID register via a **mtspr** PID0/NPIDR, a Program Correlation message is generated containing the information regarding the new process ID value. This PCM also contains the current history and instruction count. The message is provided so that address translation information can be processed by the development tool in the proper program flow. The **mtspr** PID0/NPIDR is included in the instruction count information. Note that Ownership Trace Messages (other than the periodic OTM) are redundant with the information provided, and may be disabled to avoid unnecessary message bandwidth or collisions.

61.12.6 Program Trace Overflow Error Messages

An Error Message occurs when a new message cannot be queued due to the message queue being full. The FIFO will discard incoming messages until it has completely emptied the queue. Once emptied, an Error Message will be queued. The error encoding will indicate which type(s) of messages attempted to be queued while the FIFO was being emptied.

61.12.7 Program Trace Synchronization Messages

By default, program trace messages will perform XOR compression on the branch target address to produce the address field for the message. This compression is consistent with the specification in IEEE-ISTO 5001.

Under some conditions an uncompressed address is sent to provide development tools with a baseline reference address. A Program Trace Synchronization Message is messaged via

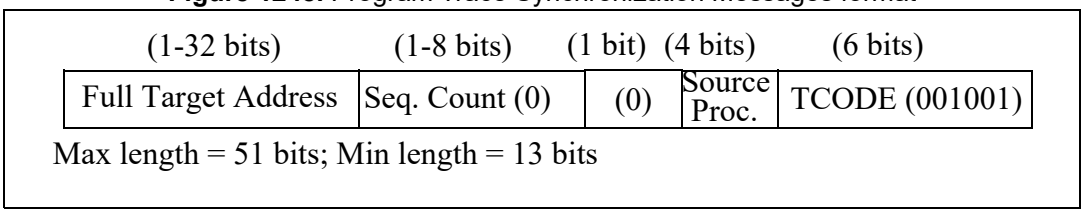
the auxiliary port (provided Program Trace is enabled) for the following conditions (see *Table 1198*):

- Initial Program Trace Message after exit from system reset or whenever program trace is enabled.
- Upon exiting from a CPU Low Power state.
- Upon exiting from Debug Mode.
- Upon occurrence of queue overrun (can be caused by any trace message), provided Program Trace is enabled.
- Upon assertion of the Event In (**nex_evti_b**) pin if the EIC bits within the DC1 Register have enabled this feature.
- When program trace becomes unmasked due to MSR_{PMM}->'1' with $DC4_{PTMARK}$='1'.

Note that the ICNT information for these messages is driven to zero, thus will not always be meaningful for some of these cases.

The format for Program Trace Synchronization Messages is as follow:

Figure 1248. Program Trace Synchronization Messages format



Exception conditions that result in Program Trace Synchronization are summarized in *Table 1198*.

Table 1198. Program Trace Exception Summary

Exception Condition	Exception Handling
System Reset Negation	At the negation of JTAG reset (**j_trst_b**), queue pointers, counters, state machines, and registers within the Nexus 3 module are reset. Upon exiting system reset (**p_reset_b**), if Program Trace is already enabled, a Program Trace Sync Message is sent.
Program Trace Enabled	The first Program Trace Message (after Program Trace has been enabled or re-enabled) is a synchronization message.
Exit from Low Power/Debug	Upon exit from a Low Power mode or Debug mode, a Program Trace Sync Message is sent.
Queue Overrun	An Error Message occurs when a new message cannot be queued due to the message queue being full. The FIFO will discard messages until it has completely emptied the queue. Once emptied, an Error Message will be queued. The error encoding will indicate which type(s) of messages attempted to be queued while the FIFO was being emptied. The next BTM message in the queue will be a Program Trace Sync Message.
Event In	If the Nexus module is enabled, a **nex_evti_b** assertion initiates a Program Trace Sync Message if Program Trace is enabled and the EIC bits of the DC1 Register have enabled this feature.

Note that for cases where program trace sync messages are generated due to program trace being enabled, the address contained in the sync message may either be the address

of the instruction which caused program trace to be enabled, or may be the address of the first instruction of an exception handler which is executed in response to unsuccessful completion of that instruction.

61.12.8 Program Trace Direct/Indirect Branch with Sync Messages

Under some conditions an uncompressed address is sent to provide development tools with a baseline reference address. A Program Trace Synchronization or a Direct/Indirect Branch with Sync Message is messaged via the auxiliary port (provided Program Trace is enabled) for the following conditions (see *Table 1198*):

- Upon direct/indirect branch after a Program Sync Message was lost due to a collision while attempting to enter the message queue.
- Upon direct/indirect branch after the sequential instruction counter has expired indicating 255 instructions have occurred since the last change of flow.
- When the periodic program trace counter has expired indicating 255 *without-sync* Program Trace Messages have occurred since the last *with-sync* message occurred.

Note that the ICNT and History information for the first message will not always be meaningful, since the temporary masking of program trace may result in ambiguous values. Subsequent w/sync messages will not have this issue.

The format for Program Trace Direct/Indirect Branch with Sync Messages is as follows:

Figure 1249. Direct/Indirect Branch with Sync Message Format

(1-32 bits)	(1-8 bits)	(1 bit)	(4 bits)	(6 bits)
Full Target Address	Sequence Count	(0)	Source Proc.	TCODE (001011 or 001100)

Max length = 51 bits; Min length = 13 bits

The format for Program Trace Indirect Branch History with Sync. Messages is as follows:

Figure 1250. Indirect Branch History w/ Sync. Message Format

(1-32 bits)	(1-32 bits)	(1-8 bits)	(1 bit)	(4 bits)	(6 bits)
Branch History	Full Target Address	Sequence Count	(0)	Source Proc.	TCODE (011101)

Max length = 83 bits; Min length = 14 bits

Exception conditions that result in Program Trace Synchronization using a w/Sync message type are summarized in *Table 1199*.

Table 1199. Program Trace Exception Summary for w/Sync BTM Messages

Exception Condition	Exception Handling
Periodic Program Trace Sync.	A forced synchronization occurs periodically after 255 non-sync Program Trace Messages have been queued. A Direct/Indirect Branch w/ Sync. Message is queued. The periodic program trace message counter then resets.
Sequential Instruction Count Overflow	After the sequential instruction counter reaches its maximum count (up to 255 sequential instructions may be executed), a forced synchronization occurs. The sequential counter then resets. A Program Trace Direct/Indirect Branch w/ Sync.Message is queued upon execution of the next branch. A Resource Full Message is Queued on the overflow event. If a branch instruction is the 255th instruction to occur, and causes a Program Trace message to be queued, then no Resource Full Message is queued, and the w/Sync message will be queued for the *next* Program Trace Direct/Indirect Branch Message.
Collision Priority	All Messages have the following priority: Instruction 0 (WPM -> DQM -> PCM_{PIDMSG}- > OTM -> BTM -> DTM)-> Instruction1 (WPM -> DQM -> OTM -> BTM -> DTM), where instruction0 is the oldest instruction. A BTM Message from Instruction1 which attempts to enter the queue at the same time as three higher priority messages from either instruction will be lost. An Error Message will be sent indicating the BTM was lost. The following direct/indirect branch will queue a Direct/Indirect Branch w/ Sync. Message. The count value within this message will reflect the number of sequential instructions executed after the last successful BTM Message was generated. This count will include the branch which did not generate a message due to the collision.

61.12.9 Enabling Program Trace

Program Trace Messaging can be enabled in one of three ways:

- Setting the TM field of the DC1 Register to enable Program Trace
- Using the PTS field of the WT Register to enable Program Trace on Watchpoint hits (watchpoints are configured within the CPU)
- Using the external hardware trace enable input signal (nex_ptm_starttr)
- Filtering of Program Trace messages may be performed using the MSR_{PMM} bit and the setting of $DC4_{PTMARK}$

61.12.10 Program Trace Timing Diagrams (2 MDO / 1 MSEO configuration)

Figure 1251. Program Trace - Indirect Branch Message (Traditional)

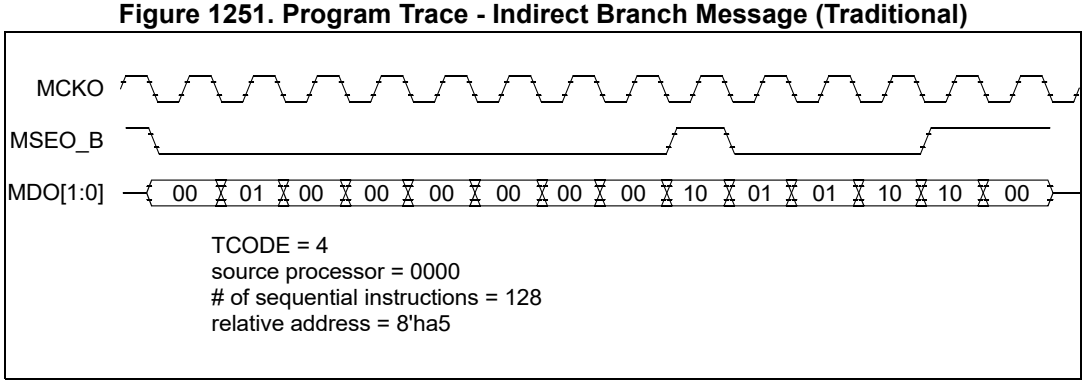


Figure 1252. Program Trace - Indirect Branch Message (History)

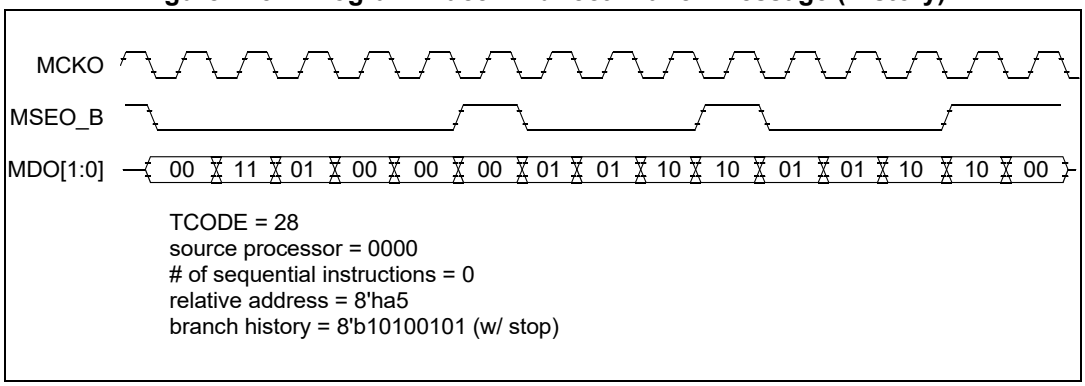


Figure 1253. Program Trace - Direct Branch (Traditional) & Error Messages

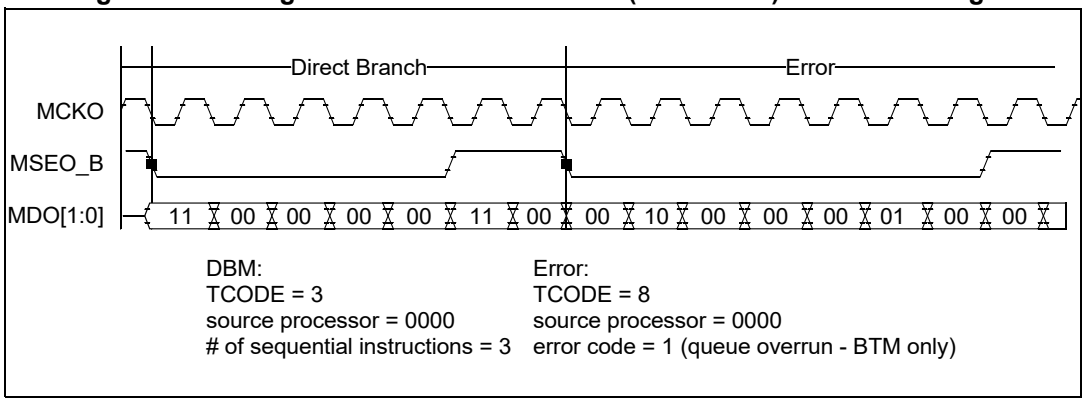
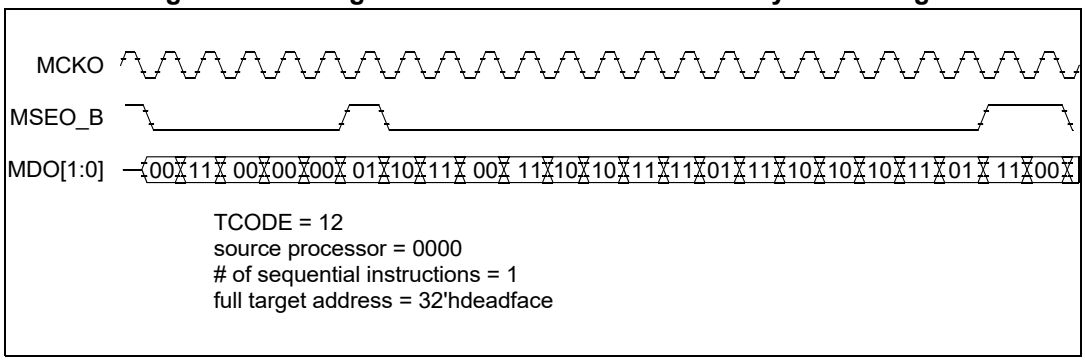


Figure 1254. Program Trace - Indirect Branch w/ Sync. Message



61.13 Data Trace

This section deals with the Data Trace mechanism supported by the Nexus 3 module. Data Trace is implemented via Data Write Messaging (DWM) and Data Read Messaging (DRM), as per the **IEEE-ISTO 5001** standard.

61.13.1 Data Trace Messaging (DTM)

Data Trace Messaging is accomplished by snooping the internal address and data buses, and storing the information for qualifying accesses (based on enabled features and matching target addresses). The Nexus 3 module traces all data access that meet the selected range and attributes.

Note: Data Trace is only performed on the internal data buses. This allows for data visibility for Core (e200z420n3) processors which incorporate a data cache. Only CPU initiated accesses will be traced. No DMA accesses to the AHB system bus will be traced.

Data Trace Messaging can be enabled in one of three ways.

- Setting the TM field of the DC1 Register to enable Data Trace.
- Using the DTS field of the WT Register to enable Data Trace on Watchpoint hits (watchpoints are configured within the Nexus1 module).
- Using the external hardware trace enable signal (**nex_dtm_starttr**)

61.13.2 DTM Message Formats

The Nexus 3 block supports five types of DTM Messages - Data Write, Data Read, Data Write Synchronization, Data Read Synchronization and Error Messages.

61.13.2.1 Data Write Messages

The Data Write Message contains the data write value and the address of the write access, relative to the previous Data Trace Message. Data Write Message information is messaged out in the following format:

Figure 1255. Data Write Message Format

(1-64 bits)	(1-32 bits)	(4 bits)	(1 bit)	(4 bits)	(6 bits)
Data Value(s)*	Relative Address	Data Size	(0)	Src. Proc	TCODE (000101)

Max length = 111 bits; Min length = 17 bits

61.13.2.2 Data Read Messages

The Data Read Message contains the data read value and the address of the read access, relative to the previous Data Trace Message. Data Read Message information is messaged out in the following format:

Figure 1256. Data Read Message Format

(1-64 bits)	(1-32 bits)	(4 bits)	(1 bit)	(4 bits)	(6 bits)
Data Value(s)*	Relative Address	Data Size	(0)	Src. Proc	TCODE (000110)

Max length = 111 bits; Min length = 17 bits

*Note: * Core (e200z420n3)-based CPUs are capable of generating two (2) reads or writes per clock cycle in cases where multiple registers are accessed with a single instruction (lmw/stmw). These will have a double word pair size encoding (**p_tsiz** = 0b000). In these cases, the Nexus 3 module will send one (1) Data Trace Message with the two 32-bit data values as one combined 64-bit value for each message.*

*For Core (e200z420n3)-based CPUs, the doubleword encoding (**p_tsiz** = 0b000) may also indicate a doubleword access and will be sent out as a single Data Trace Message with a single 64-bit data value.*

The debug/development tool will need to distinguish the two cases based on the family of processor.

61.13.2.3 Data Trace Synchronization Messages

A Data Trace Write/Read with Sync. Message is messaged via the auxiliary port (provided Data Trace is enabled) for the following conditions (see *Table 1200*):

- Initial Data Trace Message after exit from system reset or whenever Data Trace is enabled.
- Upon returning from a CPU Low Power state.
- Upon returning from Debug Mode.
- After occurrence of queue overrun (can be caused by any trace message), provided Data Trace is enabled.
- After the periodic data trace counter has expired indicating 255 *without-sync* Data Trace Messages have occurred since the last *with-sync* message occurred.
- Upon assertion of the Event In (**nex_evti_b**) pin, the first Data Trace Message will be a synchronization message if the EIC bits of the DC1 Register have enabled this feature.
- Upon Data Trace Write/Read after the previous DTM Message was lost due to an attempted access to a secure memory location (for SOC's w/ security).
- Upon Data Trace Write/Read after the previous DTM Message was lost due to a collision entering the FIFO between the DTM Message and any two of the following: Watchpoint Message, Ownership Trace Message, or Program Trace Message.

Data Trace Synchronization Messages provide the full address (without leading zeros) and insure that development tools fully synchronize with Data Trace regularly. Synchronization messages provide a reference address for subsequent DTMs, in which only the unique portion of the Data Trace address is transmitted. The format for Data Trace Write/Read with Sync. Messages is as follows:

Figure 1257. Data Write/Read with Sync. Message Format

(1-64 bits)	(1-32 bits)	(4 bits)	(1 bit)	(4 bits)	(6 bits)
Data Value	Full Address	Data Size	(0)	Source Proc.	TCODE (001101 or 001110)

Max length = 111 bits; Min length = 17 bits

Exception conditions that result in Data Trace Synchronization are summarized in *Table 1200*.

Table 1200. Data Trace Exception Summary

Exception Condition	Exception Handling
System Reset Negation	At the negation of JTAG reset (**j_trst_b**), queue pointers, counters, state machines, and registers within the Nexus 3 module are reset. If Data Trace is enabled, the first Data Trace Message is a Data Write/Read w/ Sync. Message.
Data Trace Enabled	The first Data Trace Message (after Data Trace has been enabled) is a synchronization message.
Exit from Low Power/Debug	Upon exit from a Low Power mode or Debug mode the next Data Trace Message will be converted to a Data Write/Read with Sync. Message.

Table 1200. Data Trace Exception Summary (continued)

Exception Condition	Exception Handling
Queue Overrun	An Error Message occurs when a new message cannot be queued due to the message queue being full. The FIFO will discard messages until it has completely emptied the queue. Once emptied, an Error Message will be queued. The error encoding will indicate which type(s) of messages attempted to be queued while the FIFO was being emptied. The next DTM message in the queue will be a Data Write/Read w/ Sync. Message.
Periodic Data Trace Sync.	A forced synchronization occurs periodically after 255 Data Trace Messages have been queued. A Data Write/Read w/ Sync. Message is queued. The periodic data trace message counter then resets.
Event In	If the Nexus module is enabled, a **nex_evti_b** assertion initiates a Data Trace Write/Read w/ Sync. Message upon the next data write/read (if Data Trace is enabled and the EIC bits of the DC1 Register have enabled this feature).
Collision Priority	All Messages have the following priority: Instruction 0 (WPM -> DQM -> PCM_{PIDMSG} -> OTM -> BTM -> DTM)-> Instruction1 (WPM -> DQM -> OTM -> BTM -> DTM), where instruction0 is the oldest instruction. A DTM Message which attempts to enter the queue at the same time as three other higher priority messages will be lost. A subsequent read/write will queue a Data Trace Read/Write w/ Sync. Message.

61.13.3 DTM Operation

61.13.3.1 Data Trace Windowing

Data Write/Read Messages are enabled via the RWT field in the Data Trace Control Register (DTC) for each DTM channel. Data Trace windowing is achieved via the address range defined by the DTEA and DTSA Registers and by the RC field in the DTC register. All CPU initiated read/write accesses which fall inside or outside these address ranges, as programmed, are candidates to be traced.

61.13.3.2 Data Access / Instruction Access Data Tracing

The Nexus3 module is capable of tracing either instruction access data or data access data and can be configured for either type of data trace by setting the DI1 field within the Data Trace Control Register. This setting applies to all DTM channels.

61.13.3.3 Data Trace Filtering

Data Trace filtering is available base on the settings of MSR_{PMM} and $DC4_{DTMARK}$.

61.13.3.4 Bus Cycle Special Cases

Table 1201. Bus Cycle Special Cases

Special Case	Action
Bus cycle aborted	Cycle ignored
Bus cycle with data error ($\overline{TEA}$)(1)	Data Trace Message discarded
Bus cycle completed without error(1)	Cycle captured & transmitted

Table 1201. Bus Cycle Special Cases (continued)

Special Case	Action
AHB bus cycle initiated by Nexus 3	Cycle ignored
Bus cycle is an instruction fetch	Cycle selectively ignored based on DTC_{DI} setting
Bus cycle accesses misaligned data (across 64-bit boundary) - both 1st & 2nd transactions within data trace range	1st & 2nd cycle captured & a single or a pair of DTM(s) is (are) transmitted (see Note)
Bus cycle accesses misaligned data (across 64-bit boundary) - 1st transaction within data trace range; 2nd transaction out of data trace range	1st & 2nd cycle captured & a single or a pair of DTM(s) is (are) transmitted (see Note)
Bus cycle accesses misaligned data (across 64-bit boundary) - 1st transaction within data trace range; 2nd transaction (regardless of within range or not) receives a bus error	Data Trace Message discarded
Bus cycle accesses misaligned data (across 64-bit boundary) - 1st transaction out of data trace range; 2nd transaction within data trace range	1st & 2nd cycle captured & a single or a pair of DTM(s) is (are) transmitted (see Note)
Bus cycle accesses misaligned data (across 64-bit boundary) - 1st transaction out of data trace range; 2nd transaction within range, receives a bus error	Data Trace Message discarded

1. Buffering of stores in the CPU store buffer may generate a DTM prior to the actual memory access, regardless of an error termination condition from memory.

Note: For misaligned accesses (crossing 64-bit boundary), the access is broken into two accesses by the CPU. If either access is within the data trace range, a single DTM will be sent with a size encoding indicating the size of the original access (that is word), and the address indicating the original misaligned accesses, unless the misaligned access wraps into the doubleword at address 0. In this case, since the two portions of the misaligned access are not contiguous, two DTMs will be sent, one for each portion. The size encodings and the addresses of the DTMs will indicate the accessed bytes of data. The data trace port handles these cases in the same manner. (See the Data Trace Port section in the Core (e200z420n3) Core Debug Support chapter.)

A store to the cache's store buffer within the data trace range may initiate a DTM message prior to completion of the actual memory access.

61.13.4 Data Trace Timing Diagrams(8 MDO / 2 MSEO configuration)

Figure 1258. Data Trace - Data Write Message

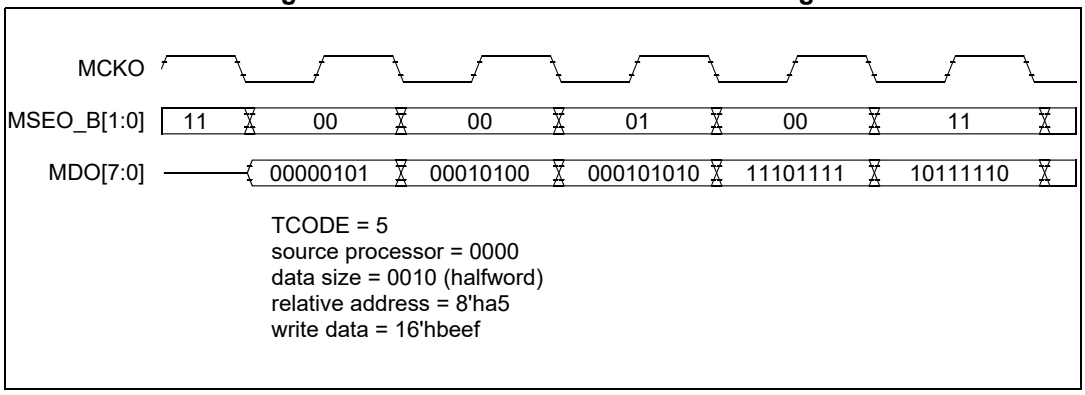
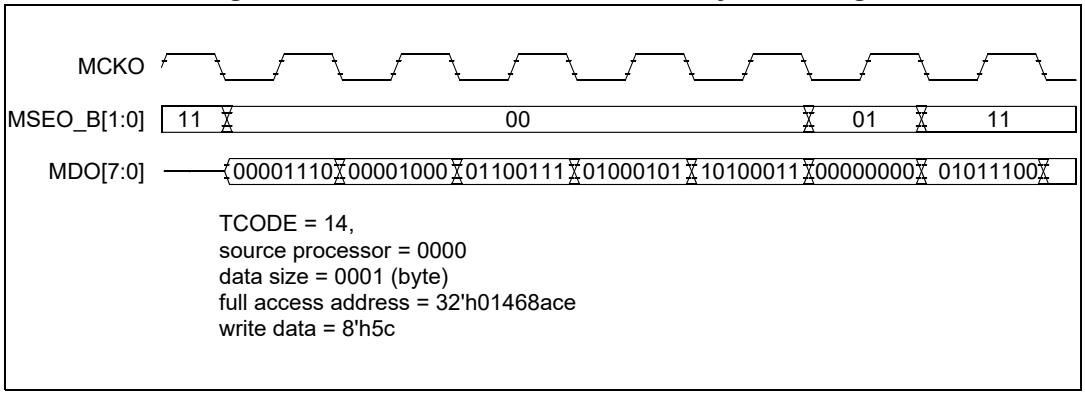


Figure 1259. Data Trace - Data Read w/ Sync Message



61.14 Data Acquisition Messaging

This section details the Data Acquisition mechanisms supported by the Nexus 3 module. Data Acquisition Trace is implemented using Data Acquisition Trace Messages in accordance with IEEE-ISTO 5001 definitions. The control mechanism to export the data is different from the recommendations of the standard, however.

Data Acquisition Trace provides a convenient and flexible mechanism for the debugger to observe the architectural state of the machine through software instrumentation.

61.14.1 Data Acquisition ID Tag Field

The DQTAG Tag field (DQTAG) is a 8-bit value specifying control or attribute information for the data included in the Data Acquisition Message. DQTAG is sampled from $DEVENT_{DQTAG}$ when a write to DDAM is performed via **mtspr** operations. The usage of the DQTAG is left to the discretion of the development tool to be used in whatever manner is deemed appropriate for the application.

61.14.2 Data Acquisition Data Field

The Data Acquisition Data field (DQDATA) is the data captured from the DDAM write operation via **mtspr** operations. Leading zeros are omitted from the message.

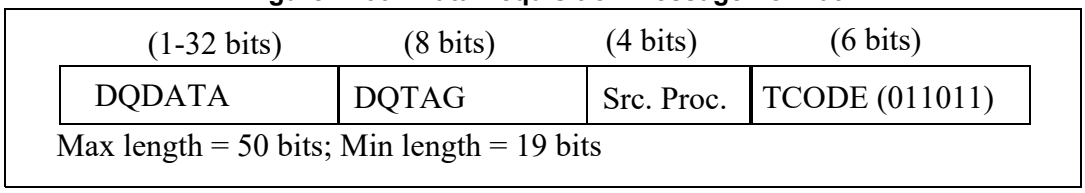
61.14.3 Data Acquisition Trace event

For DQM, a dedicated SPR has been allocated (DDAM). It is expected that the general use case is to instrument the software and use **mtspr** operations to generate Data Acquisition Messages.

There is no explicit error response for failed accesses as a result of contention between an internal and external debugger. Software may be blocked or given ownership of DDAM and the DQTAG field of the DEVENT register via control in EDBRAC0 while in External Debug Mode. Hardware always has access to these registers. Refer to the “External Debug Resource Allocation Control Register (EDBRAC0)” section in the Core (e200z420n3) Core Debug Support chapter for more detail on EDBRAC0.

Reads from the Data Acquisition channel do not generate a Data Acquisition event and will return zeroes for the read data.

Figure 1260. .Data Acquisition Message Format



61.15 Watchpoint Trace Messaging

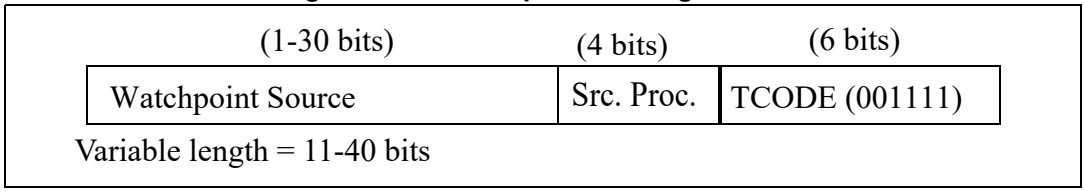
Enabling Watchpoint Messaging is done by setting the Watchpoint Trace Enable bit in the DC1 Register. Setting the individual Watchpoint sources is supported through the Nexus1 module and the Performance Monitor unit. The Nexus1 module is capable of setting multiple types of watchpoints. Please refer to the Core (e200z420n3) Core Debug Support chapter for details on watchpoint initialization.

When watchpoints occur due to one or more asserted watchpoint event signals and Watchpoint Trace Messaging is enabled, a Watchpoint Trace message will be sent to the message queue to be messaged out. This message includes the watchpoint number indicating which watchpoint(s) caused the message. If more than one enabled watchpoint occurs in a single cycle, only one Watchpoint Trace message is generated and multiple bits of the watchpoint hit field will be set. The settings of the $WMSK_{WEM}$ field control which watchpoints are enabled to generate watchpoint trace messages.

The occurrence of any of the defined watchpoints can also be programmed to assert the Event Out (**nex_evto_b**) pin for one (1) period of the output clock (**nex_mcko**) based on settings in the DC2 and DC3 registers. See *Table 1205* for details on **nex_evto_b**.

Watchpoint information is messaged out in the following format:

Figure 1261. Watchpoint Message Format



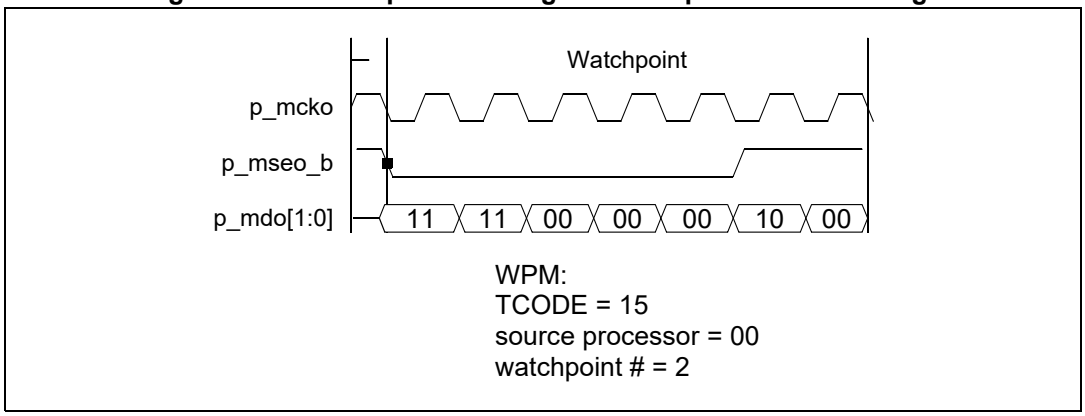
The Watchpoint Source message field will contain a ‘1’ for each asserted watchpoint. Leading zeros are truncated.

Table 1202. Watchpoint Source Encoding

Watchpoint Source (1:30 bits)	Watchpoint Description
00000000000000000000000000000000	No Watchpoints enabled for Watchpoint Trace Messaging
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1	Watchpoint #0 enabled for WTM
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1x	Watchpoint #1 enabled for WTM
xxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xx	Watchpoint #2 enabled for WTM
xxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxx	Watchpoint #3 enabled for WTM
xxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxx	Watchpoint #4 enabled for WTM
xxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxx	Watchpoint #5 enabled for WTM
xxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxx	Watchpoint #6 enabled for WTM
xxxxxxxxxxxxxxxxxxxxxxxx1xxxxxxx	Watchpoint #7 enabled for WTM
xxxxxxxxxxxxxxxxxxxxxxx1xxxxxxxx	Watchpoint #8 enabled for WTM
xxxxxxxxxxxxxxxxxxxxxx1xxxxxxxxx	Watchpoint #9 enabled for WTM
xxxxxxxxxxxxxxxxxxxxx1xxxxxxxxxx	Watchpoint #10 enabled for WTM
xxxxxxxxxxxxxxxxxxxx1xxxxxxxxxxx	Watchpoint #11 enabled for WTM
xxxxxxxxxxxxxxxxxxx1xxxxxxxxxxxx	Watchpoint #12 enabled for WTM
xxxxxxxxxxxxxxxxxx1xxxxxxxxxxxxx	Watchpoint #13 enabled for WTM
xxxxxxxxxxxxxxxxx1xxxxxxxxxxxxxx	Watchpoint #14 enabled for WTM
xxxxxxxxxxxxxxxx1xxxxxxxxxxxxxxx	Watchpoint #15 enabled for WTM
xxxxxxxxxxxxxxx1xxxxxxxxxxxxxxxx	Watchpoint #16 enabled for WTM
xxxxxxxxxxxxxx1xxxxxxxxxxxxxxxxx	Watchpoint #17 enabled for WTM
xxxxxxxxxxxxx1xxxxxxxxxxxxxxxxxx	Watchpoint #18 enabled for WTM
xxxxxxxxxxxx1xxxxxxxxxxxxxxxxxxx	Watchpoint #19 enabled for WTM
xxxxxxxxxxx1xxxxxxxxxxxxxxxxxxxx	Watchpoint #20 enabled for WTM
xxxxxxxxxx1xxxxxxxxxxxxxxxxxxxxx	Watchpoint #21 enabled for WTM
xxxxxxxxx1xxxxxxxxxxxxxxxxxxxxxx	Watchpoint #22 enabled for WTM
xxxxxxxx1xxxxxxxxxxxxxxxxxxxxxxx	Watchpoint #23 enabled for WTM
xxxxxxx1xxxxxxxxxxxxxxxxxxxxxxxx	Watchpoint #24 enabled for WTM
xxxxxx1xxxxxxxxxxxxxxxxxxxxxxxxx	Watchpoint #25 enabled for WTM
xxxxx1xxxxxxxxxxxxxxxxxxxxxxxxxx	Watchpoint #26 enabled for WTM
xxxx1xxxxxxxxxxxxxxxxxxxxxxxxxxx	Watchpoint #27 enabled for WTM
xxx1xxxxxxxxxxxxxxxxxxxxxxxxxxxx	Watchpoint #28 enabled for WTM
xx1xxxxxxxxxxxxxxxxxxxxxxxxxxxxx	Watchpoint #29 enabled for WTM
x1xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx	
1xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx	

61.15.1 Watchpoint Timing Diagram (2 MDO / 1 MSEO configuration)

Figure 1262. Watchpoint Message & Watchpoint Error Message



61.16 External Hardware Trigger Controls

The Nexus 3 module provides a set of hardware trigger inputs to allow for external hardware to also independently control enabling/disabling of Program Trace, Data Trace, Ownership Trace, and Watchpoint Trace messaging. These signals can be used to produce a “window” for trace messaging for each type.

When a start trigger condition is detected for one of the trace types, and the corresponding trace type has not already been enabled by a software write to the DC1 register, the designated trace features become enabled, and the corresponding enable bit(s) of the DC1 register is set. When a stop trigger condition is detected and the corresponding trace type has not already been enabled by a software write to the DC1 register, the designated trace feature becomes disabled, and the corresponding enable bit of the DC1 register is cleared. If a simultaneous trigger condition is received for both start and stop triggering, then the designated trace feature will toggle between enabled and disabled depending on the enable state at the time of the trigger events. For example, if tracing is enabled, and a start and stop trigger condition occur simultaneously, then tracing will be disabled. Direct writes of the DC1 register take precedence over any trace feature control action that is derived from external hardware triggering, and if the corresponding trace feature has been enabled by a direct write to DC1, then the corresponding hardware trigger signals will have no effect. *Table 1203* details the external hardware trigger signals.

Table 1203. External Hardware Trigger Signals

Name	Description
nex_dtm_starttr	Data Trace Messaging start trigger input. A transition from 0->1 on this input signals a start trace condition.
nex_dtm_stoptr	Data Trace Messaging stop trigger input. A transition from 0->1 on this input signals a stop trace condition.
nex_otm_starttr	Ownership Trace Messaging start trigger input. A transition from 0->1 on this input signals a start trace condition.
nex_otm_stoptr	Ownership Trace Messaging stop trigger input. A transition from 0->1 on this input signals a stop trace condition.

Table 1203. External Hardware Trigger Signals (continued)

Name	Description
nex_ptm_starttr	Program Trace Messaging start trigger input. A transition from 0->1 on this input signals a start trace condition.
nex_ptm_stoptr	Program Trace Messaging stop trigger input. A transition from 0->1 on this input signals a stop trace condition.
nex_wtm_starttr	Watchpoint Trace Messaging start trigger input. A transition from 0->1 on this input signals a start trace condition.
nex_wtm_stoptr	Watchpoint Trace Messaging stop trigger input. A transition from 0->1 on this input signals a stop trace condition.

These signals are active-high inputs and are transition sensitive. A 0->1 transition on the signal indicates the corresponding condition is requested. These signals are not synchronized, and must meet setup and hold requirements relative to **nex_clk**.

61.17 Nexus 3 Read/Write Access to Memory-Mapped Resources

The Read/Write access feature allows access to memory-mapped resources via the JTAG/OnCE port. The Read/Write mechanism supports single as well as block reads and writes to AHB resources.

The Nexus 3 module is capable of accessing resources on the system bus (AHB). Memory-mapped registers and other non-cached memory can be accessed via the standard memory map settings.

All accesses are setup and initiated by the Read/Write Access Control/Status Register (RWCS), as well as the Read/Write Access Address (RWA) and Read/Write Access Data Registers (RWD). Nexus 3 read/write accesses are run as privileged data non-cacheable accesses by default, and drive the **p_d_hprot[5:0]** bus access attributes to 6‘b000011 accordingly. The $RWCS_{ATTR}$ field is provided to allow a portion of these default values to be modified when performing read or write accesses using the Nexus 3 Read/Write access mechanism.

Using the Read/Write Access Registers (RWCS/RWA/RWD), memory mapped AHB resources can be accessed through Nexus 3. The following subsections describe the steps which are required to access memory-mapped resources.

Note: Read/Write Access can only access memory mapped resources when system reset is de-asserted and clocks are running.

Misaligned accesses are NOT supported in the Nexus 3 module.

Uncorrectable ECC errors on Nexus 3 read access will result in the RWD register being updated with the raw data received.

61.17.1 Single Write Access

1. Initialize the Read/Write Access Address Register (RWA) through the access method outlined in *Section 61.5*. Configure as follows:
   – Write Address -> 32h'xxxxxxxx (write address)
2. Initialize the Read/Write Access Control/Status Register (RWCS) through the access method outlined in *Section 61.5*. Configure the bits as follows:
   – Access Control (AC) -> 1b'1 (to indicate start access)
   – Map Select (MAP) -> 3b'000 (primary memory map)
   – Access Priority (PR) -> 2b'11 (highest priority)
   – Read/Write (RW) -> 1b'1 (write access)
   – Word Size (SZ) -> 3b'0xx (32-bit, 16-bit, 8-bit)

- Access Count (CNT) -> 14h'0000 or 14h'0001(single access)

Note: Access Count (CNT) of 14'h0000 or 14'h0001 will perform a single access.

3. Initialize the Read/Write Access Data Register (RWD) through the access method outlined in *Section 61.5*. Configure as follows:
   – Write Data -> 32h'xxxxxxxx (write data)
4. The Nexus block will then arbitrate for the AHB system bus and transfer the data value from the data buffer RWD Register to the memory mapped address in the Read/Write Access Address Register (RWA). When the access has completed, Nexus clears the AC and DV bits in the RWCS Register. If the access has completed without error, (ERR=1'b0), Nexus asserts the **nex_rdy_b** pin (see *Table 1205* for details) and clears the ERR bit in the RWCS Register. Otherwise, if the access completes with an error, the **nex_err_b** pin will be asserted and the ERR bit will be set to indicate an error has occurred, and the **nex_rdy_b** pin will not be asserted. Once DV has been cleared, this indicates that the device is ready for the next access, or that an error has occurred.

*Note: Only the **nex_rdy_b** and **nex_err_b** pins as well as the AC, DV, and ERR bits within the RWCS provide Read/Write Access status to the external development tool.*

61.17.2 Block Write Access

1. For a block write access, follow Steps 1, 2, and 3 outlined in *Section 61.17.1* to initialize the registers, but using a value greater than one (14'h0001) for the CNT field in the RWCS Register.
2. The Nexus block will then arbitrate for the AHB system bus and transfer the first data value from the RWD Register to the memory mapped address in the Read/Write Access Address Register (RWA). When the transfer has completed without error, the address from the RWA Register is incremented to the next word size (specified in the SZ field), the number from the CNT field is decremented, and the **nex_rdy_b** pin is asserted. If the access has completed without error, Nexus clears the ERR and DV bits in the RWCS Register, otherwise the ERR bit will be set and the DV bit will be cleared to indicate an error has occurred, the **nex_err_b** pin is asserted, the block transfer is aborted, and the AC bit in the RWCS register is cleared. Once DV has been cleared, this indicates that the device is ready for the next access in the block transfer, or an error has occurred.
3. If AC has not been cleared due to an error, repeat Step 3 in *Section 61.17.1* until the internal CNT value is zero (0). When this occurs, the AC bit within the RWCS will be cleared to indicate the end of the block write access.

Note: The actual RWA value as well as the CNT field within the RWCS are not changed when executing a block write access. The original values can be read by the external development tool at any time.

61.17.3 Single Read Access

1. Initialize the Read/Write Access Address Register (RWA) through the access method outlined in *Section 61.5*. Configure as follows:
   – Read Address -> 32h'xxxxxxxx (read address)
2. Initialize the Read/Write Access Control/Status Register (RWCS) through the access method outlined in *Section 61.5*. Configure the bits as follows:
   – Access Control (AC) -> 1b'1 (to indicate start access)
   – Map Select (MAP) -> 3b'000 (primary memory map)
   – Access Priority (PR) -> 2b'11 (highest priority)
   – Read/Write (RW) -> 1b'0 (read access)
   – Word Size (SZ) -> 3b'0xx (32-bit, 16-bit, 8-bit)
   – Access Count (CNT) -> 14h'0000 or 14h'0001(single access)

Note: Access Count (CNT) of 14'h0000 or 14'h0001 will perform a single access.

3. The Nexus block will then arbitrate for the AHB system bus and the read data will be transferred from the AHB to the RWD Register. When the access has completed without error, Nexus clears the ERR bit and sets the DV bit in the RWCS Register and asserts the **nex_rdy_b** pin (see *Table 1205* for detail on **nex_rdy_b**). Otherwise, if the access has completed with an error, the **nex_err_b** pin will be asserted, the ERR bit will be set and the DV bit will be cleared to indicate an error has occurred, and the **nex_rdy_b** pin will not be asserted. Once ERR or DV has been set, this indicates that the device is ready for the next access, or that an error has occurred. The AC bit will be cleared in either case.
4. The data can then be read from the Read/Write Access Data Register (RWD) through the access method outlined in *Section 61.5*.

*Note: Only the **nex_rdy_b** and **nex_err_b** pins as well as the AC, DV and ERR bits within the RWCS provide Read/Write Access status to the external development tool.*

61.17.4 Block Read Access

1. For a block read access, follow Steps 1 and 2 outlined in *Section 61.17.3* to initialize the registers, but using a value greater than one (14'h0001) for the CNT field in the RWCS Register.
2. The Nexus block will then arbitrate for the AHB system bus and the read data will be transferred from the AHB to the RWD Register. When the access has completed without error, Nexus clears the ERR bit and sets the DV bit in the RWCS Register and asserts the **nex_rdy_b** pin (see *Table 1205* for detail on **nex_rdy_b**). Otherwise, if the access has completed with an error, the **nex_err_b** pin will be asserted, the ERR bit will be set and the DV bit will be cleared to indicate an error has occurred, and the **nex_rdy_b** pin will not be asserted. Once ERR or DV has been set, this indicates that the device is ready for the next access, or that an error has occurred. The AC bit will be cleared in either case.
3. When the transfer has completed without error, the address from the RWA Register is incremented to the next word size (specified in the SZ field), the number from the CNT

field is decremented, Nexus clears the ERR bit and sets the DV bit in the RWCS Register, and asserts the **nex_rdy_b** pin (see *Table 1205* for detail on **nex_rdy_b**). Otherwise, if the access has completed with an error, the **nex_err_b** pin will be asserted, the ERR bit will be set and the DV bit will be cleared to indicate an error has occurred, the AC bit is cleared and the block transfer is aborted, and the **nex_rdy_b** pin will not be asserted. Once ERR or DV has been set, this indicates that the device is ready for the next access, or that an error has occurred. Once DV has been set, this indicates that the device is ready for the next access in the block transfer, or if ERR is set (AC will be cleared), the block transfer has been aborted.

4. The data can then be read from the Read/Write Access Data Register (RWD) through the access method outlined in *Section 61.5*.
5. If AC has not been cleared due to an error, repeat Steps 3 and 4 in *Section 61.17.3* until the CNT value is zero (0). When this occurs, the AC bit within the RWCS is cleared to indicate the end of the block read access.

Note: The data values must be shifted out 32-bits at a time LSB first (that is doubleword read = two word reads from the RWD).

The actual RWA value as well as the CNT field within the RWCS are not changed when executing a block read access. The original values can be read by the external development tool at any time.

61.17.5 Error Handling

The Nexus 3 module handles various error conditions as follows:

61.17.5.1 AHB Read/Write Error

All address and data errors that occur on read/write accesses to the AHB system bus will return a transfer error encoding on the **p_hresp[1:0]** signals. If this occurs:

1. The access is terminated without re-trying (AC bit is cleared)
2. The ERR bit in the RWCS Register is set
3. The Error Message is sent (TCODE = 8) indicating Read/Write Error

61.17.5.2 Access Termination

The following cases are defined for sequences of the Read/Write protocol that differ from those described in the above sections.

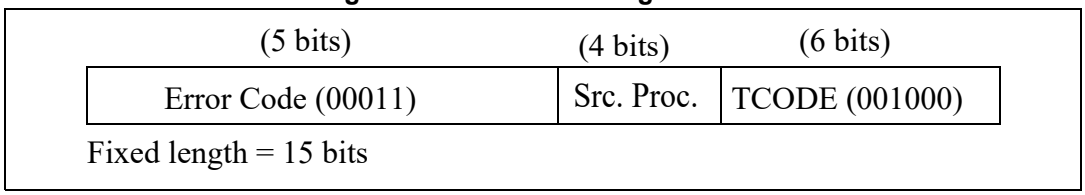
1. If the AC bit in the RWCS Register is set to start Read/Write accesses and invalid values are loaded into the RWD and/or RWA, then an AHB access error may occur. This is handled as described above.
2. If a block access is in progress (all cycles not completed), and the RWCS Register is written, then the original block access is terminated at the boundary of the nearest completed access.
   a) If the RWCS is written with the AC bit set, the next Read/Write access will begin and the RWD can be written to/ read from.
   b) If the RWCS is written with the AC bit cleared, the Read/Write access is terminated at the nearest completed access. This method can be used to break (early terminate) block accesses.

61.17.6 Read/Write Access Error Message

The Read/Write Access Error Message is sent out when an AHB system bus access error (read or write) has occurred.

Error information is messaged out in the following format:

Figure 1263. Error Message Format



61.18 Nexus 3 Pin Interface

This section details information regarding the Nexus 3 pins and pin protocol.

The Nexus 3 pin interface provides the function of transmitting messages from the messages queues to the external tools. It is also responsible for handshaking with the message queues.

61.18.1 Pins Implemented

The Nexus 3 module implements an auxiliary port consisting of one (1) **nex_evti_b** and one (1) **nex_mseo_b** or two (2) **nex_mseo_b[1:0]**. It also implements a configurable number of **nex_mdo[n:0]** pins, (1) **nex_rdy_b** pin, (1) **nex_err_b** pin, (1) **nex_evto_b** pin, and one (1) clock output pin (**nex_mcko**), as well as additional configuration pins described in *Table 1205*. The output pins are synchronized to the Nexus 3 output clock (**nex_mcko**).

All Nexus 3 input functionality may be controlled through the JTAG/OnCE port in compliance with IEEE 1149.1 (see *Section 61.5* for details). The JTAG pins are incorporated as I/O to the processor, and are further described in the “JTAG/OnCE Pins” section of the Core (e200z420n3) Core Debug Support chapter.

Table 1204. JTAG Pins for Nexus 3

JTAG Pins	Input/ Output	Description of JTAG Pins (included in Nexus 1)
j_tdo	O	The Test Data Output (**j_tdo**) pin is the serial output for test instructions and data. **j_tdo** is three-stateable and is actively driven in the “Shift-IR” and “Shift-DR” controller states. **j_tdo** changes on the falling edge of **j_tclk**.
j_tdi	I	The Test Data Input (**j_tdi**) pin receives serial test instruction and data. TDI is sampled on the rising edge of **j_tclk**.
j_tms	I	The Test Mode Select (**j_tms**) input pin is used to sequence the OnCE controller state machine. **j_tms** is sampled on the rising edge of **j_tclk**.
j_tclk	I	The Test Clock (**j_tclk**) input pin is used to synchronize the test logic, and control register access through the JTAG/OnCE port.
j_trst_b	I	The Test Reset (**j_trst_b**) input pin is used to asynchronously initialize the JTAG/OnCE controller.

The auxiliary pins are used to send and receive messages and are described in *Table 1205*.

Table 1205. Nexus 3 Auxiliary Pins

Auxiliary Pins	Input/ Output	Description of Auxiliary Pins
nex_mcko	O	Message Clock Out (**nex_mcko**) is a free running output clock to development tools for timing of **nex_mdo[n:0]** & **nex_mseo_b[1:0]** pin functions. **nex_mcko** is programmable through the DC1 Register.
nex_mdo[n:0]	O	Message Data Out (**nex_mdo[n:0])** are output pins used for OTM, BTM, and DTM. External latching of **nex_mdo[n:0]** shall occur on the rising edge of the Nexus3 clock (**nex_mcko**).
nex_mseo_b[1:0]	O	Message Start/End Out (**nex_mseo_b[1:0]**) are output pins which indicate when a message on the **nex_mdo[n:0]** pins has started, when a variable length packet has ended, and when the message has ended. External latching of **nex_mseo_b[1:0]** shall occur on the rising edge of the Nexus3 clock (**nex_mcko**). One or two pin MSEO functionality is determined at integration time per SOC implementation
nex_rdy_b	O	Ready (**nex_rdy_b**) is an output pin used to indicate to the external tool that the Nexus block is ready for the next Read/Write Access. If Nexus is enabled, this signal is asserted upon successful (without error) completion of an AHB system bus transfer (Nexus read or write) & is held asserted until the JTAG/OnCE state machine reaches the “Capture_DR” state. Upon exit from system reset or if Nexus is disabled, **nex_rdy_b** remains de-asserted
nex_err_b	O	Error (**nex_err_b**) is an output pin used to indicate to the external tool that the Nexus block is ready for the next Read/Write Access and an error has occurred on the previous access. If Nexus is enabled, this signal is asserted upon unsuccessful (with error) completion of an AHB system bus transfer (Nexus read or write) & is held asserted until the JTAG/OnCE state machine reaches the “Capture_DR” state. Upon exit from system reset or if Nexus is disabled, **nex_err_b** remains de-asserted
nex_evto_b	O	Event Out (**nex_evto_b**)is an output which, when asserted, indicates one of two events has occurred based on the EOC bits in the DC1 Register. **nex_evto_b** is held asserted for one (1) cycle of **nex_mcko**: 1) one (or more) watchpoints has occurred (from Nexus1) & EOC = 2’b00 2) debug mode was entered (jd_debug_b asserted from Nexus1) & EOC = 2’b01
nex_evti_b	I	Event In (**nex_evti_b**) is an input which, when asserted, will initiate one of two events based on the EIC bits in the DC1 Register (if the Nexus module is enabled at reset): 1) Program Trace & Data Trace synchronization messages (provided Program Trace & Data Trace are enabled & EIC = 2’b00). 2) Debug request to Nexus1 module (provided EIC = 2’b01 and this feature is implemented).

Table 1205. Nexus 3 Auxiliary Pins (continued)

Auxiliary Pins	Input/ Output	Description of Auxiliary Pins
nex_ext_src_id[0 :3]	I	nex_ext_src_id[0:3] is used to provide the SRC field value used in each message. These pins are tied to a predetermined value at SoC integration time
nex_sfwcntl_en]	I	nex_sfwcntl_en is used to allow software to control the module resources via the DCR register mappings. SoC logic should drive this signal appropriately in a semi-static manner based on the presence of an external hardware debugger and appropriate security precautions.

The Nexus auxiliary port arbitration pins are used when the Nexus 3 module is implemented in a multi-Nexus SoC which shares a single auxiliary output port. The arbitration is controlled by an SoC level Nexus Auora Router (NAR). Refer to *Section 61.20* for detail on Nexus port arbitration.

Table 1206. Nexus Port Arbitration Signals

Nexus Port Arbitration Pins	Input/ Output	Description of Arbitration Pins
nex_aux_req[1:0]	O	Nexus Auxiliary Request (**nex_aux_req[1:0]**) output signals indicate to an SoC level Nexus arbiter a request for access to the shared Nexus auxiliary port in a multi-Nexus implementation. The priority encodings are determined by how many messages are currently in the message queues (See *Table 1208*.)
nex_aux_busy	O	Nexus Auxiliary Busy (**nex_aux_busy**) is an output signal to an SoC level Nexus arbiter indicating that the Nexus 3 module is currently transmitting its message after being granted the Nexus auxiliary port.
nar_aux_grant	I	Nexus Auxiliary Grant (**nar_aux_grant**) is an input from the SoC level NAR that the auxiliary port has been granted to the Nexus 3 module to transmit its message.
ext_multi_nex_sel	I	Multi-Nexus Select (**ext_multi_nex_sel**) is a static signal indicating that the Nexus 3 module is implemented within a multi-Nexus environment. If set, port control and arbitration is controlled by the SoC level arbitration module (NAR).

61.18.2 Pin Protocol

The protocol for the processor transmitting messages via the auxiliary pins is accomplished with the MSEO pin function outlined in *Table 1207*. Both single and dual pin cases are shown.

nex_mseo_b[1:0] is used to signal the end of variable-length packets, and not fixed length packets. **nex_mseo_b[1:0]** is sampled on the rising edge of the Nexus 3 clock (**nex_mcko**).

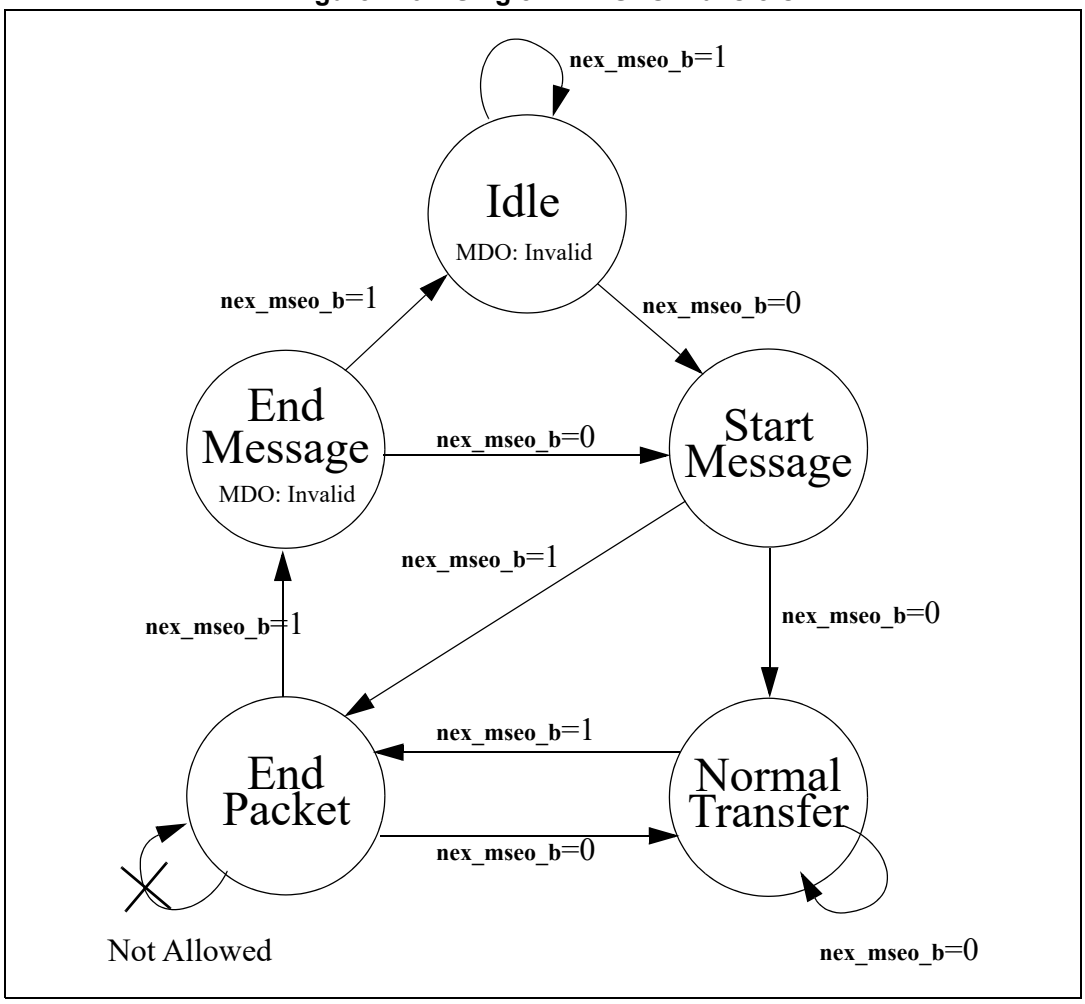
Single pin MSEO is not supported on the SPC574Sx.

Table 1207. MSEO Pin(s) Protocol

nex_mseo_b Function	Single nex_mseo_b data (serial)	Dual nex_mseo_b[1:0] data
Start of message	1-1-0	11-00
End of message	0-1-1-(more 1's)	00 (or 01)-11-(more 1's)
End of variable length packet	0-1-0	00-01
Message transmission	0's	00's
Idle (no message)	1's	11's

Figure 1264 illustrates the state diagram for single pin MSEO transfers.

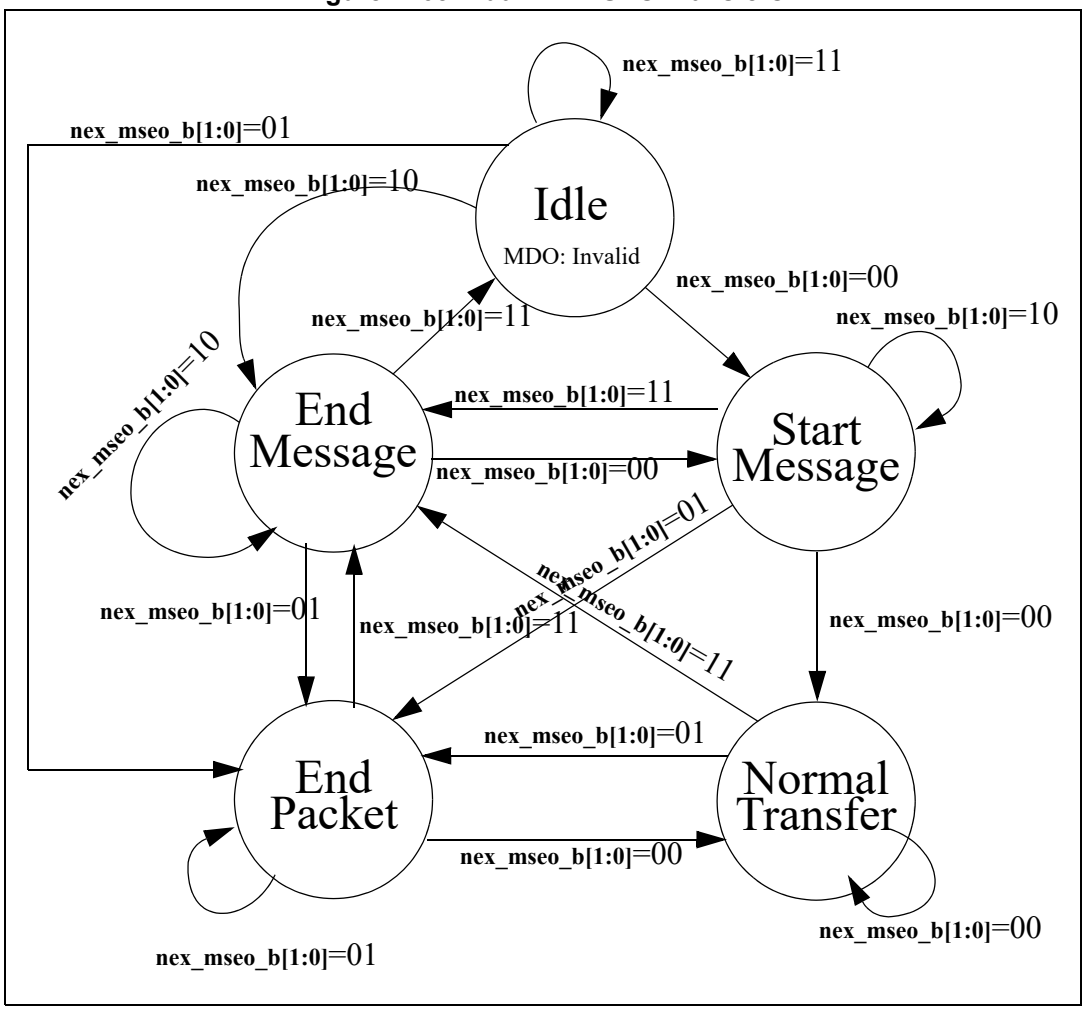
Figure 1264. Single Pin MSEO Transfers



Note that the "End Message" state does not contain valid data on the **nex_mdo[n:0]** pins. Also, It is not possible to have two consecutive "End Packet" messages. This implies the minimum packet size for a variable length packet is 2x the number of **nex_mdo[n:0]** pins. This ensures that a false end of message state is not entered by emitting two consecutive '1's on the **nex_mseo_b** pin before the actual end of message.

Figure 1265 illustrates the state diagram for dual pin MSEO transfers.

Figure 1265. Dual Pin MSEO Transfers



The dual pin MSEO option is more robust that the single pin option. Termination of the current message may immediately be followed by the start of the next message on the consecutive clocks. An extra clock to end the message is not necessary as with the one MSEO pin option. The dual pin option also allows for consecutive “End Packet” states. This can be an advantage when small, variable sized packets are transferred.

Note: The “End Message” state may also indicate the end of a variable-length packet as well as the end of the message when using the dual pin option.

61.19 Rules for Output Messages

Class 3 compliant embedded processors must provide messages via the auxiliary port in a consistent manner as described below:

- A variable-sized packet within a message must end on a port boundary.
- A variable-sized packet may start within a port boundary only when following a fixed length packet. (If two variable-sized packets end and start on the same clock, it is

impossible to know which bit is from the last packet and which bit is from the next packet.)

- Whenever a variable-length packet is sized such that it does not end on a port boundary, it is necessary to extend and zero fill the remaining bits after the highest-order bit so that it can end on a port boundary. For example, if the **nex_mdo[n:0]** port is 2 bits wide, and the unique portion of an indirect address TCODE is 5 bits, then the remaining 1 bit of **nex_mdo[n:0]** must be packed with a 0.

61.20 Auxiliary Port Arbitration

In a multi-Nexus environment, the Nexus 3 module must arbitrate for the shared Nexus port at the SoC level.The request scheme is implemented as a 2-bit request with various levels of priority. The priority levels are defined in *Table 1208* below. The Nexus 3 module will receive a 1-bit grant signal (**nar_aux_grant**) from the SoC level arbiter. When a grant is received, the Nexus 3 module will begin transmitting its message following the protocol outlined in *Section 61.18.2*. The Nexus 3 module will maintain control of the port, by asserting the **nex_aux_busy** signal, until the MSEO state machine reaches the “End Message” state.

Table 1208. MDO Request Encodings

Request Level	MDO Request Encoding (nex_aux_req[1:0])	Condition of Queue
No Request	00	No message to send
Low Priority	01	Message queue less than 1/2 full
—	10	Reserved
High Priority	11	Message queue 1/2 full or more

61.21 Examples

The following are examples of Program Trace and Data Trace Messages.

Table 1209 illustrates an example Indirect Branch Message with 2 MDO / 1MSEO configuration. *Table 1210* illustrates the same example with an 8 MDO / 2 MSEO configuration.

Note that T0 and S0 are the least significant bits where:

- Tx = TCODE number (fixed)
- Sx = Source processor (fixed)
- MAP = (always set to 0)
- Ix = Number of instructions (variable)
- Ax = Unique portion of the address (variable)

Note that during clock 13, the **nex_mdo[n:0]** pins are ignored in the single MSEO case.

Table 1209. Indirect Branch Message Example (2 MDO / 1 MSEO)

Clock	nex_mdo[1:0]		nex_mseo_b	State
0	X	X	1	Idle (or end of last message)
1	T1	T0	0	Start Message
2	T3	T2	0	Normal Transfer
3	T5	T4	0	Normal Transfer
4	S1	S0	0	Normal Transfer
5	S3	S2	0	Normal Transfer
6	I0	MAP	0	Normal Transfer
7	I2	I1	0	Normal Transfer
8	I4	I3	1	End Packet
9	A1	A0	0	Normal Transfer
10	A3	A2	0	Normal Transfer
11	A5	A4	0	Normal Transfer
12	A7	A6	1	End Packet
13	0	0	1	End Message
14	T1	T0	0	Start Message

Table 1210. Indirect Branch Message Example (8 MDO / 2 MSEO)

Clock	nex_mdo[7:0]								nex_mseo_b[1:0]		State
0	X	X	X	X	X	X	X	X	1	1	Idle (or end of last message)
1	S1	S0	T5	T4	T3	T2	T1	T0	0	0	Start Message
2	I4	I3	I2	I1	I0	MAP	S3	S2	0	1	End Packet
3	A7	A6	A5	A4	A3	A2	A1	A0	1	1	End Packet/End Message
4	S1	S0	T5	T4	T3	T2	T1	T0	0	0	Start Message

Table 1211 and *Table 1212* illustrate examples of Direct Branch Messages: one with 2 MDO / 1 MSEO, and one with 8 MDO / 2 MSEO.

Note that T0 and I0 are the least significant bits where:

- Tx = TCODE number (fixed)
- Sx = Source processor (fixed)
- Ix = Number of Instructions (variable)

Table 1211. Direct Branch Message Example (2 MDO / 1 MSEO)

Clock	nex_mdo[1:0]		nex_mseo_b	State
0	X	X	1	Idle (or end of last message)
1	T1	T0	0	Start Message
2	T3	T2	0	Normal Transfer
3	T5	T4	0	Normal Transfer
4	S1	S0	0	Normal Transfer
5	S3	S2	0	Normal Transfer
6	I1	I0	1	End Packet
7	0	0	1	End Message

Table 1212. Direct Branch Message Example (8 MDO / 2 MSEO)

Clock	nex_mdo[7:0]								nex_mseo_b[1:0]		State
0	X	X	X	X	X	X	X	X	1	1	Idle (or end of last message)
1	S1	S0	T5	T4	T3	T2	T1	T0	0	0	Start Message
2	0	0	0	0	I1	I0	S3	S2	1	1	End Packet/End Message
3	S1	S0	T5	T4	T3	T2	T1	T0	0	0	Start Message

Table 1213 illustrates an example Data Write Message with 8 MDO / 1 MSEO configuration, and *Table 1214* illustrates the same DWM with 8 MDO / 2 MSEO configuration

Note that T0, A0, D0 are the least significant bits where:

- Tx = TCODE number (fixed)
- Sx = Source processor (fixed)
- MAP = (always set to 0)
- Zx = Data size (fixed)
- Ax = Unique portion of the address (variable)
- Dx = Write data (variable - 8, 16 or 32-bit)

Table 1213. Data Write Message Example (8 MDO / 1 MSEO)

Clock	nex_mdo[7:0]								nex_mseo_b	State
0	X	X	X	X	X	X	X	X	1	Idle (or end of last message)
1	S1	S0	T5	T4	T3	T2	T1	T0	0	Start Message
2	A0	Z3	Z2	Z1	Z0	0	S3	S2	1	End Packet
3	D7	D6	D5	D4	D3	D2	D1	D0	0	Normal Transfer
4	0	0	0	0	0	0	0	0	1	End Packet
5	0	0	0	0	0	0	0	0	1	End Message

Table 1214. Data Write Message Example (8 MDO / 2 MSEO)

Clock	nex_mdo[7:0]								nex_mseo_b[1:0]		State
0	X	X	X	X	X	X	X	X	1	1	Idle (or end of last message)
1	S1	S0	T5	T4	T3	T2	T1	T0	0	0	Start Message
2	A0	Z3	Z2	Z1	Z0	0	S3	S2	0	1	End Packet
3	D7	D6	D5	D4	D3	D2	D1	D0	1	1	End Packet/ End Message

61.22 Electrical Characteristics

For all electrical characteristics related to core processor and Nexus 3 operation, please refer to the SPC574Sx datasheet.

61.23 IEEE 1149.1 (JTAG) RD/WR Sequences

This section contains example JTAG/OnCE sequences used to access resources.

61.23.1 JTAG Sequence for Accessing Internal Nexus Registers

Table 1215. Accessing Internal Nexus 3 Registers via JTAG/OnCE

Step #	TMS Pin	Description
1	1	IDLE -> SELECT-DR_SCAN
2	0	SELECT-DR_SCAN -> CAPTURE-DR (Nexus Command Register value loaded in shifter)
3	0	CAPTURE-DR -> SHIFT-DR
4	0	(7) TCK clocks issued to shift in direction (rd/wr) bit and first 6 bits of Nexus reg. addr.
5	1	SHIFT-DR -> EXIT1-DR (7th bit of Nexus reg. shifted in)
6	1	EXIT1-DR -> UPDATE-DR (Nexus shifter is transferred to Nexus Command Register)
7	1	UPDATE-DR -> SELECT-DR_SCAN
8	0	SELECT-DR_SCAN -> CAPTURE-DR (Register value is transferred to Nexus shifter)
9	0	CAPTURE-DR -> SHIFT-DR
10	0	(31) TCK clocks issued to transfer register value to TDO pin while shifting in TDI value
11	1	SHIFT-DR -> EXIT1-DR (MSB of value is shifted in/out of shifter)
12	1	EXIT1-DR -> UPDATE -DR (if access is write, shifter is transferred to register)
13	0	UPDATE-DR -> RUN-TEST/IDLE (transfer complete - Nexus controller to Reg. Select state)

61.23.2 JTAG Sequence for Read Access of Memory-Mapped Resources

Table 1216. Accessing Memory-Mapped Resources (Reads)

Step #	TCLK clocks	Description
1	13	Nexus Command = write to Read/Write Access Address Register (RWA)
2	37	Write RWA (initialize starting read address: data input on TDI)
3	13	Nexus Command = write to Read/Write Control/Status Register (RWCS)
4	37	Write RWCS (initialize read access mode and CNT value: data input on TDI)
5	—	Wait for falling edge of **nex_rdy_b** or **nex_err_b** pin
6	13	Nexus Command = read Read/Write Access Data Register (RWD)
7	37	Read RWD (data output on TDO)
8	—	If CNT > 0, go back to Step #5

61.23.3 JTAG Sequence for Write Access of Memory-Mapped Resources

Table 1217. Accessing Memory-Mapped Resources (Writes)

Step #	TCLK clocks	Description
1	13	Nexus Command = write to Read/Write Access Control/Status Register (RWCS)
2	37	Write RWCS (initialize write access mode and CNT value - data input on TDI)
3	13	Nexus Command = write to Read/Write Address Register (RWA)
4	37	Write RWA (initialize starting write address - data input on TDI)
5	13	Nexus Command = read Read/Write Access Data Register (RWD)
6	37	Write RWD (data output on TDO)
7	—	Wait for falling edge of **nex_rdy_b** or **nex_err_b** pin
8	—	If CNT > 0, go back to Step #5

62 Nexus Crossbar Multi-Master Client (NXMC)

62.1 Introduction

The Nexus Crossbar Multi-master Client (NXMC) module provides real-time trace capabilities in compliance with the IEEE-ISTO Nexus 5001-2012 standard. This module provides development support capabilities for devices without requiring address and data pins for internal visibility. A portion of the pin interface is also compliant with the IEEE 1149.1 JTAG standard. The IEEE-ISTO 5001 standard defines an extensible auxiliary port which is used in conjunction with the JTAG port.

Many bus masters such as DMA AHB, and CPU Data Bus AHB start accesses to internal address locations via the system bus. The NXMC module monitors the system bus and provides real-time trace information to debug or development tools.This also provides the capability of sending user defined messages for the peripherals attached to it.

62.1.1 Features

The NXMC module is compliant with the IEEE-ISTO 5001-2012 standard. The following features are implemented:

- Data trace via Data Write Messaging (DWM) and Data Read Messaging (DRM). This provides the capability for the development tool to trace reads or writes, or both, to (selected) internal memory resources.
- Multi-master tracing capability for multiple master accesses
- Watchpoint messaging based on internal/external triggers, via the auxiliary pins
- Watchpoint trigger events out based on internal/external triggers
- Watchpoint trigger enable of data trace messaging
- Start or stop of Tracing (Data Trace, Watchpoint Trace) based on external triggers
- Nexus Auxiliary port for higher data throughput
  - One $\overline{\text{EVTO}}$ (Watchpoint Event) pin
  - One $\overline{\text{EVTI}}$ (Event In) pin
- Registers for data trace, watchpoint generation, and watchpoint trigger
- All features controllable and configurable via the JTAG port

62.2 External signal description

This section details information regarding the standard ports and protocol. The NXMC pin interface provides the function of transmitting messages from the messages queues to NPC. It is also responsible for handshaking with the message queues.

The NXMC module implements the following signals:

- One $\overline{\text{EVTI}}$
- One $\overline{\text{EVTO}}$

These pins are described in *Table 1219*.

All NXMC input functionality is controlled through the JTAG port in compliance with IEEE 1149.1 (see *Section 62.5.1: IEEE 1149.1 (JTAG) Test Access Port* for details). A separate JTAG TAP controller is instantiated within the NXMC. *Table 1218* describes the JTAG pins.

Table 1218. JTAG pins for NXMC

JTAG port	Input/ output	Description
TDO	O	The Test Data Output (TDO) pin is the serial output for test instructions and data. TDO is three-stateable and is actively driven in the shift-IR and shift-DR controller states. TDO changes on the falling edge of TCK.
TDI	I	The Test Data Input (TDI) pin receives serial test instruction and data. TDI is sampled on the rising edge of TCK and has an internal pull-up resistor.
TMS	I	The Test Mode Select (TMS) input pin is used to sequence the JTAG test controllers' state machine. TMS is sampled on the rising edge of TCK and has an internal pull-up resistor.
TCK	I	The Test Clock (TCK) input pin is used to synchronize the test logic, and control register access through the JTAG port.
TRST	I	The Test Reset ($\overline{\text{TRST}}$) input pin is used to asynchronously initialize the JTAG controller. The $\overline{\text{TRST}}$ pin has an internal pull-down resistor.

Table 1219. NXMC auxiliary pins

Auxiliary port	Input/ output	Description
EVTO	O	Event Out ($\overline{\text{EVTO}}$) is an output which, when asserted, indicates one of two events has occurred based on the EOC bits in the DC Register: – one of the internal or external watchpoints has occurred and EOC = 2'b00 – system-level debug mode was entered (ipg_debug) and EOC = 2'b01 $\overline{\text{EVTO}}$ is held asserted for one cycle.
EVTI	I	Event In ($\overline{\text{EVTI}}$) is an input which initiates synchronization messages. If the Nexus module is enabled at reset, (see *Section 62.5.2: Enabling NXMC operation*), assertion initiates data trace synchronization messages (provided data trace is enabled).

62.3 Supported messages and TCODEs

The NXMC pins allow for flexible transfer operations via public messages. A TCODE defines the transfer format, the number or size of the packets, or both, to be transferred, and the purpose of each packet. The IEEE-ISTO 5001-2012 standard defines a set of public messages. The NXMC block supports the public TCODEs listed in *Table 1220*. The SRC and MASTER field values are described in the device-specific chapter that describes how the modules are configured.

Table 1220. Supported public TCODEs

Message name	Min packet size (bits)	Max packet size (bits)	Packet name	Packet type	Packet description
Data Trace - Data Write Message	6	6	TCODE	fixed	TCODE number = 58
	4	4	SRC	fixed	Source processor identifier (multiple Nexus configuration)
	4	4	MASTER	fixed	Indicates which master (ID) initiated the data access
	3	3	DSZ	fixed	Data size (See *Table 1221*)
	1	32	U-ADDR	variable	Unique portion of the data write address
	8	64	DATA	fixed	Data write value (size fixed based on DSZ field)
	1	26	TSTAMP	variable	Timestamp (Optional)
Data Trace - Data Read Message	6	6	TCODE	fixed	TCODE number = 59
	4	4	SRC	fixed	Source processor identifier (multiple Nexus configuration)
	4	4	MASTER	fixed	Indicates which master (ID) initiated the data access
	3	3	DSZ	fixed	Data size (See *Table 1221*)
	1	32	U-ADDR	variable	Unique portion of the data read address
	8	64	DATA	fixed	Data read value (size fixed based on DSZ field)
	1	26	TSTAMP	variable	Timestamp (Optional)
Error Message	6	6	TCODE	fixed	TCODE number = 8
	4	4	SRC	fixed	Source processor identifier (multiple Nexus configuration)
	4	4	ETYPE	fixed	Error type (See *Table 1222*)
	14	14	ECODE	fixed	Error code (See *Table 1223*)
	1	26	TSTAMP	variable	Timestamp (Optional)
Data Trace - Data Write Message with Sync	6	6	TCODE	fixed	TCODE number = 60
	4	4	SRC	fixed	Source processor identifier (multiple Nexus configuration)
	4	4	MASTER	fixed	Indicates which master (ID) initiated the data access
	3	3	DSZ	fixed	Data size (See *Table 1221*)
	1	32	F-ADDR	variable	Full access address (leading zero (0) truncated)
	8	64	DATA	fixed	Data write value (size fixed based on DSZ field)
	1	26	TSTAMP	variable	Timestamp (Optional)

Table 1220. Supported public TCODEs (continued)

Message name	Min packet size (bits)	Max packet size (bits)	Packet name	Packet type	Packet description
Data Trace - Data Read Message with Sync	6	6	TCODE	fixed	TCODE number = 61
	4	4	SRC	fixed	Source processor identifier (multiple Nexus configuration)
	4	4	MASTER	fixed	Indicates which master (ID) initiated the data access
	3	3	DSZ	fixed	Data size (See *Table 1221*)
	1	32	F-ADDR	variable	Full access address (leading zero (0) truncated)
	8	64	DATA	fixed	Data read value (size fixed based on DSZ field)
	1	26	TSTAMP	variable	Timestamp (Optional)
Watchpoint Message	6	6	TCODE	fixed	TCODE number = 15
	4	4	SRC	fixed	Source processor identifier (multiple Nexus configuration)
	1	8	WPHIT	variable	Watchpoint source(s) number (See *Table 1235*)
	1	26	TSTAMP	variable	Timestamp (Optional)

Table 1221. Data trace size (DSZ) encodings (TCODE = 5,6,13,14)

DSZ	Transfer size
001	8-bit
010	16-bit
011	32-bit
100	64-bit
000, 101-111	Reserved

Table 1222. Error Type (ETYPE) encodings (applicable to all error messages)

ETYPE	Description
0000	Queue overrun caused message (one or more) to be lost
0001	Contention with higher priority message caused message(s) to be lost
0010	Reserved
0011	Reserved
0100	Reserved
0101	Invalid access opcode (Nexus register unimplemented)
0110–1111	Reserved

Table 1223. Error Code (ECODE) encodings (applicable based on error source)

ECODE		Description[1]
13 to 8	7 to 0	
Source (SRC)		
000000	XXXXXXX1	Watchpoint Message(s) lost
000000	XXXXXX1X	Data Trace Message(s) lost
000000	XXXXX1XX	Reserved
000000	XXXX1XXX	Reserved
000000	XXX1XXXX	Reserved
000000	XX1XXXXX	Reserved
000000	X1XXXXXX	Reserved
000000	1XXXXXXX	Reserved
Source (SRC) n = peripheral index		
CKSRCn	XXXXXXX1	Reserved
CKSRCn	XXXXXX1X	Reserved
CKSRCn	XXXXX1XX	Reserved
CKSRCn	XXXX1XXX	Reserved
CKSRCn	XXX1XXXX	Reserved
CKSRCn	XX1XXXXX	Reserved
CKSRCn	X1XXXXXX	Reserved
CKSRCn	1XXXXXXX	Reserved

1. The SRC field values are described in the device-specific chapter that describes how the modules are configured.

62.4 Register description

This section describes the NXMC register definition. Nexus registers are accessed using the JTAG port in compliance with IEEE 1149.1. See *Section 62.5.1: IEEE 1149.1 (JTAG) Test Access Port* for details. The complete list of registers is shown in *Table 1224*.

Table 1224. NXMC registers

Nexus register	Nexus access opcode	Read/ write	Read address	Write address
Development Control 1 (DC1)	0x2	R/W	0x04	0x05
Development Control 2 (DC2)	0x3	R/W	0x06	0x07
Watchpoint Trigger (WT)	0xB	R/W	0x16	0x17
Data Trace Control (DTC)	0xD	R/W	0x1A	0x1B
Data Trace Start Address1 (DTSA1)	0xE	R/W	0x1C	0x1D
Data Trace Start Address2 (DTSA2)	0xF	R/W	0x1E	0x1F

Table 1224. NXMC registers (continued)

Nexus register	Nexus access opcode	Read/ write	Read address	Write address
Data Trace End Address1 (DTEA1)	0x12	R/W	0x24	0x25
Data Trace End Address2 (DTEA2)	0x13	R/W	0x26	0x27
Breakpoint/Watchpoint Control Register1 (BWC1)	0x16	R/W	0x2C	0x2D
Breakpoint/Watchpoint Control Register2 (BWC2)	0x17	R/W	0x2E	0x2F
Breakpoint/Watchpoint Address Register1 (BWA1)	0x1E	R/W	0x3C	0x3D
Breakpoint/Watchpoint Address Register2 (BWA2)	0x1F	R/W	0x3E	0x3F

62.4.1 Register definitions

Detailed register definitions for the current NXMC implementation are described in the following sections.

62.4.1.1 Development control register 1 (DC1)

The Development Control Registers control the basic development features of the NXMC module. *Figure 1266* shows the format of the DC1.

Address: refer to *Table 1224* Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0(1)	0(1)	0(1)	EOC		0	0	WEN	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	EIC		TM		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. This bit is not implemented and it's value has no impact on the system

Figure 1266. Development control register 1 (DC1)

The DC1 is described in *Table 1225*.

Table 1225. DC1 register field description

Field	Description
3:4 EOC	$\overline{\text{EVTO}}$ Control. 00 $\overline{\text{EVTO}}$ upon occurrence of watchpoint (internal or external) 01 $\overline{\text{EVTO}}$ upon entry into system-level Debug Mode 1x Reserved
7 WEN	Watchpoint Trace Enable. 0 Watchpoint Messaging disabled 1 Watchpoint Messaging enabled

Table 1225. DC1 register field description (continued)

Field	Description
27:28 EIC	$\overline{\text{EVTI}}$ Control 00 $\overline{\text{EVTI}}$ for synchronization (Data Trace) 01 Reserved 10 $\overline{\text{EVTI}}$ disabled 11 Reserved
29:31 TM	Trace Mode 000 No Trace 1xx Reserved x1x Data Trace enabled xx1 Reserved

62.4.1.2 Development control register 2 (DC2)

The Development Control Registers control the basic development features of the NXMC module. *Figure 1267* shows the format of the DC2.

Address: refer to *Table 1224* Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	EWC								0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0(1)
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. This bit is not implemented and it's value has no impact on the system

Figure 1267. Development control register 2 (DC2)

The DC2 is described in *Table 1226*.

Table 1226. DC2 register field description

Field	Description
0:7 EWC	$\overline{\text{EVTO}}$ Watchpoint Configuration(1). 00000000 No Watchpoints trigger $\overline{\text{EVTO}}$ 1XXXXXXX External Watchpoint #1 triggers $\overline{\text{EVTO}}$ X1XXXXXX External Watchpoint #2 triggers $\overline{\text{EVTO}}$ XX1XXXXX Reserved(2) XXX1XXXX Reserved(2) XXXX1XXX Internal Watchpoint #1 triggers $\overline{\text{EVTO}}$ XXXXX1XX Internal Watchpoint #2 triggers $\overline{\text{EVTO}}$ XXXXXX1X Reserved(2) XXXXXXX1 Reserved(2)

1. The EOC bits in DC1 must be programmed to trigger EVTO on watchpoint occurrence for the EWC bits to have any effect.
2. Writing bits 2,3,6,7 has no impact and read value will be 0 for these bits.

62.4.1.3 Watchpoint trigger register (WT)

The Watchpoint Trigger Register allows the watchpoints defined either internally to the NXMC module or by an external module to trigger actions. These watchpoints can control data trace enable and disable. The WT bits can be used to produce an address related window for triggering trace messages. *Figure 1268* shows the format of the WT register.

Address: refer to *Table 1224* Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	DTS			DTE			0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

Figure 1268. Watchpoint trigger register (WT)

The WT register is described in *Table 1227*.

Table 1227. WT register field description

Field	Description(1)
6;8 DTS	Data Trace Start Control. 000 Trigger disabled 001 Use External Watchpoint #1 010 Use External Watchpoint #2 011 Reserved 100 Reserved 101 Use Internal Watchpoint #1 (BWA1 Register) 110 Use Internal Watchpoint #2 (BWA2 Register) 111 Reserved
9;11 DTE	Data Trace End Control. 000Trigger disabled 001 Use External Watchpoint #1 010 Use External Watchpoint #2 011 Reserved 100 Reserved 101 Use Internal Watchpoint #1 (BWA1 Register) 110 Use Internal Watchpoint #2 (BWA2 Register) 111 Reserved

1. The WT bits only enable data trace if the TM bits in the Development Control Register (DC) have not already been set to enable data trace.

62.4.1.4 Data trace control register (DTC)

The Data Trace Control Register controls whether DTM messages are restricted to reads, writes or both for a user programmable address range. There are two data trace channels

controlled by the DTC for the NXMC module. *Figure 1269* shows the format of the DTC register.

Address: refer to *Table 1224* Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	RWT1		RWT2		0	0	0	MME	0	0			MID			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	RC1	RC2	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1269. Data trace control (DTC) register

The DTC register is described in *Table 1228*.

Table 1228. DTC field description

Field	Description
0:1 RWT1	Read/Write Trace 1. 00 No trace messages generated x1 Enable data read trace 1x Enable data write trace
2:3 RWT2	Read/Write Trace 2. 00 No trace messages generated x1 Enable data read trace 1x Enable data write trace
7 MME	Multi Master Tracing enable. 0 Tracing enabled only for Master with ID = MID (DTC[19:16]) 1 Tracing enabled for all the Masters
12:15 MID	ID of the system bus Master to be traced(1)
24 RC1	Range Control 1. 0 Condition trace on address within range (endpoints inclusive) 1 Condition trace on address outside of range (endpoints exclusive)
25 RC2	Range Control 2. 0 Condition trace on address within range (endpoints inclusive) 1 Condition trace on address outside of range (endpoints exclusive)

1. The MASTER field values are described in the device-specific chapter that describes how the modules are configured.

62.4.1.5 Data trace start address registers 1 and 2 (DTSA1 and DTSA2)

The Data Trace Start Address Registers define the start addresses for each trace channel. *Figure 1270* shows the format of the DTSA1 and DTSA2 registers.

Address: refer to *Table 1224* Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	DATA TRACE START ADDRESS															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	DATA TRACE START ADDRESS															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1270. Data trace start address registers 1 and 2 (DTSA1 and DTSA2)

62.4.1.6 Data trace end address registers 1 and 2 (DTEA1 and DTEA2)

The Data Trace End Address Registers define the end addresses for each Trace channel. *Figure 1271* shows the format of the DTEA1 and DTEA2 registers.

Address: refer to *Table 1224* Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	DATA TRACE END ADDRESS															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	DATA TRACE END ADDRESS															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1271. Data trace end address registers 1 and 2 (DTEA1 and DTEA2)

Table 1229 illustrates the range that are selected for data trace for various cases of DTSA being less than, greater than, or equal to DTEA.

Table 1229. Data trace address range options

Programmed values	Range control bit value	Range selected
DTSA ≤ DTEA	0	DTSA -> <- DTEA
DTSA ≤ DTEA	1	<- DTSA DTEA ->
DTSA > DTEA	N/A	Invalid range – no trace

Note: DTSA must be less than (or equal to) DTEA in order to guarantee correct data write/read traces.

When the range control bit is 0 (internal range), accesses to DTSA and DTEA addresses are traced. When the range control bit is 1 (external range), accesses to DTSA and DTEA are not traced.

62.4.1.7 Breakpoint/watchpoint control register 1 (BWC1)

Breakpoint/watchpoint control register 1 controls the attributes for generation of NXMC watchpoint#1. *Figure 1272* shows the format of the BWC1 register.

Address: refer to *Table 1224* Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	BWE1		BRW1		0	0	0	0	0	0	0	0	0	0	BWR1	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	BWT1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1272. Breakpoint/watchpoint control register 1(BWC1)

The BWC1 register is described in *Table 1230*.

Table 1230. BWC1 field description

Field	Description
0:1 BWE1	Breakpoint/Watchpoint #1 Enable 00 Internal Nexus Watchpoint #1 disabled 01 Reserved 10 Reserved 11 Internal Nexus Watchpoint #1 enabled
2:3 BRW1	Breakpoint/Watchpoint #1 Read/Write Select 00 Watchpoint #1 hit on read accesses 01 Watchpoint #1 hit on write accesses 10 Watchpoint #1 on read or write accesses 11 Reserved
14:15 BWR1	Breakpoint/Watchpoint #1 Register Compare 00 No Register Compare (same as BWC1[31:30] = 2'b00) 01 Reserved 10 Compare with BWA1 value 11 Reserved
16 BWT1	Breakpoint/Watchpoint #1 Type 0 Reserved 1 Watchpoint #1 on data accesses

62.4.1.8 Breakpoint/watchpoint control register 2 (BWC2)

Breakpoint/Watchpoint Control Register2 controls attributes for generation of NXMC watchpoint#2. *Figure 1273* shows the format of the BWC2 register.

Address: refer to *Table 1224* Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	BWE2		BRW2		0	0	0	0	0	0	0	0	0	0	BWR2	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	BWT2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1273. Breakpoint/watchpoint control register 2(BWC2)

The BWC2 register is described in *Table 1231*.

Table 1231. BWC2 field description

Field	Description
0:1 BWE2	Breakpoint/Watchpoint #2 Enable 00 Internal Nexus Watchpoint #2 disabled 01 Reserved 10 Reserved 11 Internal Nexus Watchpoint #2 enabled
2:3 BRW2	Breakpoint/Watchpoint #2 Read/Write Select 00 Watchpoint #2 hit on read accesses 01 Watchpoint #2 hit on write accesses 10 Watchpoint #2 on read or write accesses 11 Reserved
14:15 BWR2	Breakpoint/Watchpoint #2 Register Compare. 00 No Register Compare (same as BWC2[31:30] = 2b00) 01 Reserved 10 Compare with BWA2 value 11 Reserved
16 BWT2	Breakpoint/Watchpoint #2 Type 0 Reserved 1 Watchpoint #2 on data accesses

62.4.1.9 Breakpoint/watchpoint address registers 1 and 2 (BWA1 and BWA2)

The Breakpoint/Watchpoint Address Registers are compared with system bus addresses in order to generate internal watchpoints. *Figure 1274* shows the format of the BWA1 and BWA2 registers.

Address: refer to *Table 1224* Access: User read/write

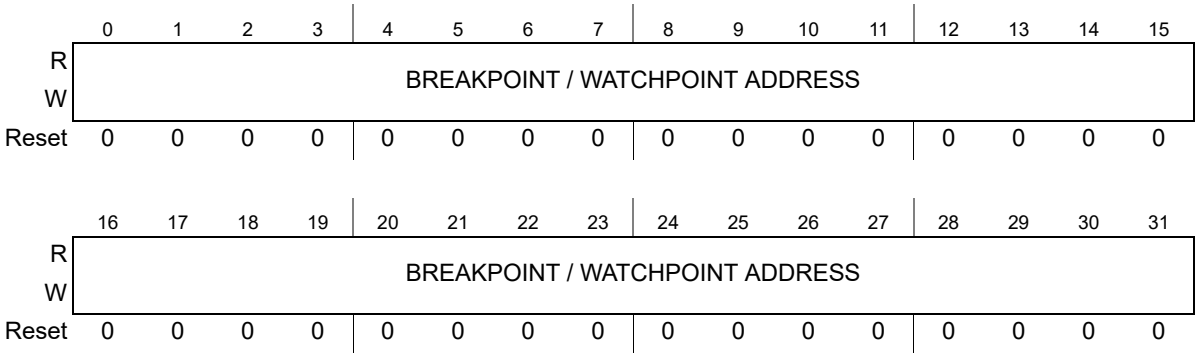


Figure 1274. Breakpoint /watchpoint address registers 1 and 2 (BWA1 and BWA2)

62.4.1.10 Unimplemented registers

Unimplemented registers are those with client select and index value combinations other than those listed in *Table 1224*. For unimplemented registers, the NXMC module drives TDO to zero during the SHIFT-DR state. It also transmits an error message with the invalid access opcode encoding (ETYPE = 0101).

62.5 Programming considerations (RESET)

If NXMC register configuration is to occur during system reset (as opposed to debug mode), all NXMC configuration should be completed between the negation of $\overline{\text{TRST}}$ and system reset de-assertion, after the JTAG ID register has been read by the tool after negation of $\overline{\text{TRST}}$ and should be programmed before deassertion of system reset.

62.5.1 IEEE 1149.1 (JTAG) Test Access Port

The NXMC module uses the IEEE 1149.1 TAP controller for accessing Nexus resources. The JTAG signals themselves are shared by all TAP controllers on the device. The NXMC module implements a 4-bit Instruction Register (IR). The valid instructions and method for register access are outlined in *Section 62.5.4: NXMC register access via JTAG*.

62.5.2 Enabling NXMC operation

The Nexus module is enabled by loading a single instruction into the JTAG Instruction Register (IR). Once enabled, the module is ready to accept control input via the JTAG pins.

The Nexus module is disabled when the JTAG state machine reaches the Test-Logic-Reset state. This state can be reached by the assertion of the $\overline{\text{TRST}}$ pin or by cycling through the state machine using the TMS pin. The Nexus module is also disabled if a Power-on-Reset (POR) event occurs.

If the NXMC module is disabled, no trace output is provided. Nexus registers are not available for reads or writes.

62.5.3 Enabling NXMC TAP controller

Assertion of a Power-on-Reset signal or assertion of the $\overline{\text{TRST}}$ pin resets all TAP controllers. Upon exit from the Test-Logic-Reset state, the IR value is loaded with the JTAG

ID. When the NXMC TAP is accessed, this information helps the development tool obtain information about the Nexus module it is accessing, such as version, sequence, feature set.

62.5.4 NXMC register access via JTAG

Access to Nexus register resources is enabled by loading a single instruction into the JTAG Instruction Register (IR). Once the JTAG NEXUS-ACCESS instruction has been loaded, the JTAG port allows tool/target communications with all Nexus registers according to the map in *Table 1224*.

Reading/writing of a Nexus register then requires two passes through the Data-Scan (DR) path of the JTAG state machine.

1. The first pass through the DR selects the Nexus register to be accessed by providing an index (see *Table 1224*), and the direction (read/write). This is achieved by loading an 8-bit value into the JTAG Data Register (DR). This register has the format shown in *Figure 1275* and *Table 1232*.

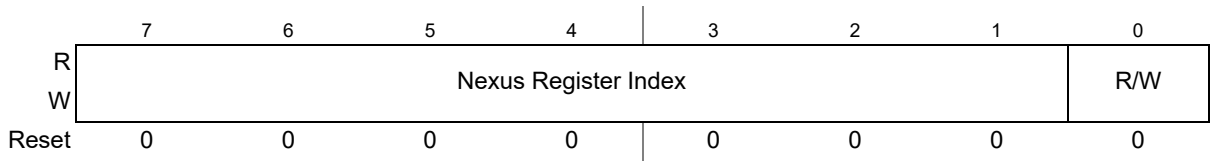


Figure 1275. JTAG DR for Nexus register access

Table 1232. JTAG DR field description for Nexus register access

Field	Description
Nexus Register Index	Selected from values in *Table 1224*
Read/Write (R/W)	0 Read 1 Write

2. The second pass through the DR then shifts the data in or out of the JTAG port, LSB first.
   a) During a read access, data is latched from the selected Nexus register when the JTAG state machine passes through the Capture-DR state.
   b) During a write access, data is latched into the selected Nexus register when the JTAG state machine passes through the Update-DR state.

62.6 Functional description

62.6.1 Data trace

This section describes the data trace mechanism supported by the NXMC module. Data trace is implemented via Data Write Messaging (DWM) and Data Read Messaging (DRM), as per the IEEE-ISTO 5001-2012 standard.

62.6.1.1 Data trace messaging (DTM)

NXMC data trace messaging is accomplished by snooping the system bus and storing the information for qualifying accesses (based on enabled features and matching target

addresses). The NXMC module traces all data access that meet the selected range and attributes.

62.6.1.2 DTM message formats

The NXMC block supports five types of DTM messages:

- Data write
- Data read
- Data write synchronization
- Data read synchronization
- Error messages

62.6.1.2.1 Data write and data read messages

The data write and data read messages contain the data write/read value and the address of the write/read access, relative to the previous data trace message. The DATA field is considered fixed based on the DSZ encodings to either 8, 16, 32, 64. Data write message and data read message information is messaged out in the format shown in *Figure 1276*. The SRC field values are described in the device-specific chapter that describes how the modules are configured.

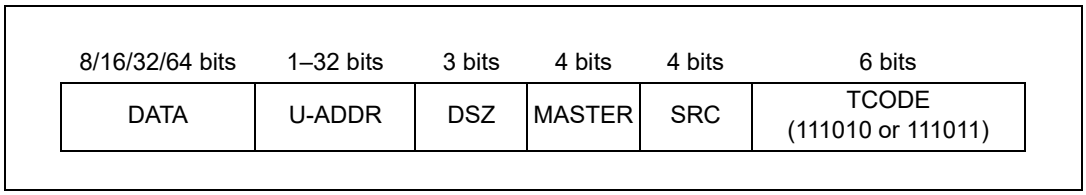


Figure 1276. DTM message format for DATA width = 64 bits

62.6.1.2.2 Data trace with non-contiguous byte strobes asserted

The v6 extensions to AMBA 2.0 allow system bus transfers with bytes non-asserted byte strobes in between asserted byte strobes. The NXMC does not support transfers with non-contiguous byte strobes asserted. In such a condition, the invalid bytes in the DATA packet are driven to the value of 0xFF.

62.6.1.2.3 DTM overflow error messages

An error message occurs when a new message cannot be queued due to the message queue being full. The FIFO discards incoming messages until it has completely emptied the queue. Once emptied, an error message is queued. The error encoding indicates which type(s) of messages attempted to be queued while the FIFO was being emptied.

If only a data trace message attempts to enter the queue while it is being emptied, the error message incorporates the data trace only error encoding (ECODE = 00000000000010). If a watchpoint also attempts to be queued while the FIFO is being emptied, then the error message incorporates error encoding (ECODE = 00000000000011).

Error information is messaged out in the format shown in *Figure 1277*. The SRC field values are described in the device-specific chapter that describes how the modules are configured.

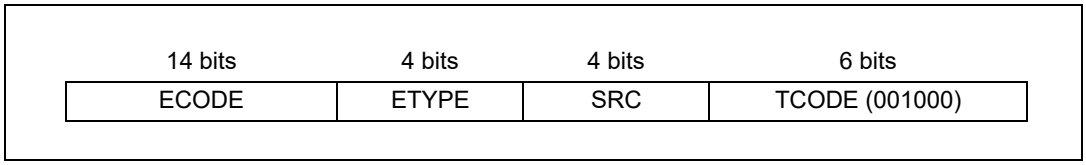


Figure 1277. DTM error message format (fixed length = 48 bits)

62.6.1.2.3.1 Data trace synchronization messages

A data trace write/read with synchronization message is messaged via the auxiliary port (provided data trace is enabled) for the following conditions (see *Table 1233*):

- Initial data trace message upon exit from system reset or whenever data trace is enabled is a synchronization message
- Upon returning from a low power state, the first data trace message is a synchronization message
- Upon returning from debug mode, the first data trace message is a synchronization message
- After occurrence of queue overrun (can be caused by any trace message), the first data trace message is a synchronization message
- After the periodic data trace counter has expired indicating 255 without-sync data trace messages have occurred since the last with-sync message occurred
- Upon assertion of the event in (EVTI) pin, the first data trace message is a synchronization message if the EIC bits of the DC Register have enabled this feature
- Upon data trace write/read after the previous DTM Message was lost due to a collision while entering the FIFO between the DTM message and any of the following: error message, or watchpoint message or peripheral message.This is a very rare occurrence and possible only in case of continuous burst on the system bus as well as watchpoint events which is unrealistic

Data trace synchronization messages provide the full address (without leading zeros) and ensure that development tools fully synchronize with data trace regularly. Synchronization messages provide a reference address for subsequent DTMs, in which only the unique portion of the data trace address is transmitted. The format for data trace write/read with synchronization messages is shown in *Figure 1278*. The SRC and MASTER field values are described in the device-specific chapter that describes how the modules are configured.

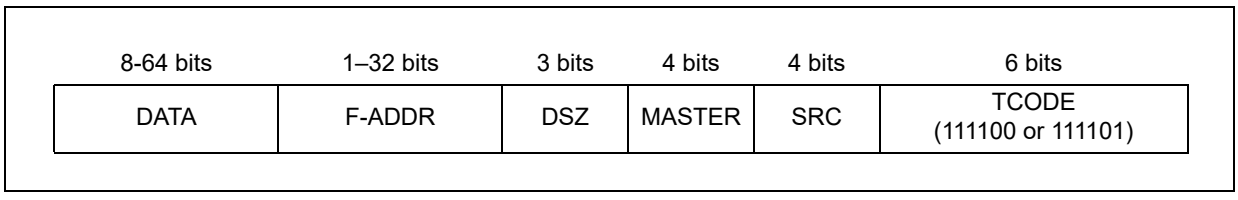


Figure 1278. Data write/read with synchronization message format

Exception conditions that result in data trace synchronization are summarized in *Table 1233*.

Table 1233. Data trace exception summary

Exception condition	Exception handling
System Reset Negation	At the negation of JTAG reset ($\overline{\text{TRST}}$), queue pointers, counters, state machines, and registers within the NXMC module are reset. If data trace is enabled, the first data trace message is a data write/read with synchronization message.
Data Trace Enabled	The first data trace message (after data trace has been enabled) is a synchronization message.
Exit from Low Power/Debug	Upon exit from a low power mode or debug mode the next data trace message is converted to a data write/read with synchronization message.
Queue Overrun	An error message occurs when a new message cannot be queued due to the message queue being full. The FIFO discards messages until it has completely emptied the queue. Once emptied, an error message is queued. The error encoding indicates which type(s) of messages attempted to be queued while the FIFO was being emptied. The next DTM message in the queue is a data write/read with synchronization message.
Periodic Data Trace Synchronization	A forced synchronization occurs periodically after 255 data trace messages have been queued. A data write/read with synchronization message is queued. The periodic data trace message counter then resets.
Event In	If the Nexus module is enabled, an $\overline{\text{EVTI}}$ assertion initiates a data write/read with synchronization message upon the next data write/read (if data trace is enabled and the EIC bits of the DC Register have enabled this feature).
Collision Priority	All messages have the following priority: Error > WPM > ICTM > DTM. Although a very rare occurrence, during simultaneous burst lower priority messages can be lost due to burst of high priority messages. Proper error messages are generated and in case of DTM lost a subsequent read/write queues a data write/read with synchronization message.

62.6.1.3 DTM operation

62.6.1.3.1 Enabling data trace messaging

Data trace messaging can be enabled in one of three ways:

- Setting the TM field of the DC Register via JTAG to enable data trace (DC[1]).
- Using the DTS field of the WT Register to enable data trace on watchpoint hits (either watchpoints configured within the NXMC module or external watchpoint inputs controlled by the device).
- By toggling the external start control, which in turn sets the TM field of the DC register to enable data trace. (The TM bit clears and hence the data tracing stops on the toggle of external stop control. In case of conflict between JTAG write and external start/stop control, JTAG write takes precedence.

Note: When enabling the DTM, either via JTAG or external start/stop control, it is important to make sure that all the other controls (that is DTC, DTSA/E1/2 and WT register fields) are correctly configured to ensure correct DTM functionality.

62.6.1.3.2 DTM queueing

The NXMC implements a programmable depth queue for queuing all messages. Messages that enter the queue are transmitted via the auxiliary pins in the order in which they are queued.

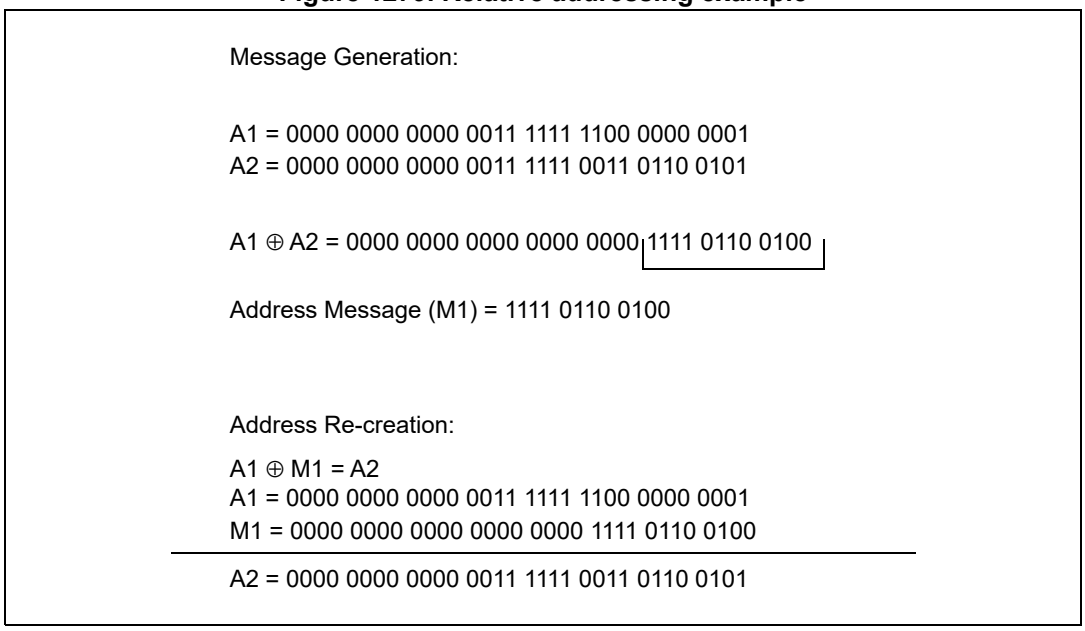
Note: If multiple trace messages need to be queued at the same time, watchpoint messages have the highest priority (WPM > ICTM > DTM).

62.6.1.3.3 Relative addressing

The relative address feature is compliant with IEEE-ISTO 5001-2012 and is designed to reduce the number of bits transmitted for addresses of data trace messages. The address transmitted is relative to the address of the previous data trace message. It is generated by XORing the new address with the previous address, and then using only the results up to the most significant 1 in the result. To recreate this address, an XOR of the (most-significant 0-padded) message address with the previously decoded address gives the current address.

For example, if the previous address (A1) = 0x0003FC01, the new address (A2) = 0x0003F365 as shown in *Figure 1279*.

Figure 1279. Relative addressing example



62.6.1.3.4 Data trace windowing

Data write/read messages are enabled via the RWT1(2) field in the Data Trace Control Register (DTC) for each DTM channel. Data trace windowing is achieved via the address range defined by the DTEA and DTSA Registers and by the RC1(2) field in the DTC. All system bus DMA initiated read/write accesses which fall inside or outside these address ranges, as programmed, are candidates to be traced.

62.6.1.3.5 System bus cycle special cases

The system bus cycle special cases are shown in *Table 1234*.

Table 1234. System Bus cycle special cases

Special case	Action
System bus cycle with data error	Data trace message discarded
System bus cycle completed without error	Cycle captured and transmitted
System bus cycle is an instruction fetch	Cycle ignored

62.6.1.3.6 System bus multi-master tracing

DTM tracing can be done either for all the masters or for a single master (indicated by the MID field of the DTC register) based on the DTC register MME field. In case of single master, the correct value needs to be configured in the MID register before enabling the data trace operation.

The MASTER field of the DTM message (see figure in *Section 62.6.1.2.1: Data write and data read messages* for an example) indicates the system bus master for which the trace has been generated. For each masters (0 to 15) there is a parameter associated (MSID0-15) for indicating its actual ID, which finally gets inserted in the MASTER field. The MASTER field values are described in the device-specific chapter that describes how the modules are configured.

62.6.2 Watchpoint support

The NXMC module provides watchpoint messaging via the auxiliary pins, as defined by IEEE-ISTO 5001-2012. Watchpoint messages can be generated by either using the NXMC defined internal watchpoints or by externally defined watchpoint signals.

62.6.2.1 Watchpoint messaging

Enabling watchpoint messaging is accomplished by setting the WEN bit in the DC Register. This bit is set either via JTAG write 1 access or external start indication, and cleared via JTAG write 0 access or external start indication. In case of conflict between JTAG write and external start/stop control, JTAG write takes precedence.

Note: In case of enabling the watchpoint messaging, either via JTAG or external start/stop control, it is important to make sure that all the other Watch Point Message (WPM) controls (that are BWC1/2, BWA1/2 register fields) are configured with the correct values in order to ensure correct WPM operation.

Watchpoint setting is supported through two methods:

- Internal watchpoints: Using the BWC1(2) registers, two independently controlled internal watchpoints can be initialized. When a system bus access address matches on BWA1(2), a watchpoint message is transmitted. The system bus multi-master or single master controls applies here too as explained for DTM (in *Section 62.6.1.3.6: System bus multi-master tracing*)
- External watchpoints: The NXMC module supports two external watchpoint inputs. The optional logic to generate these watchpoint signals is implemented at the device level.The watchpoints are described in the device-specific chapter that describes how the modules are configured. If either of these signals asserts (that is on 0-to-1 transition), a watchpoint message is transmitted

The Nexus module provides watchpoint messaging using the IEEE-ISTO 5001 defined TCODE. When any of the four possible watchpoint sources asserts, a message is sent to

the queue to be messaged out. This message indicates the watchpoint number as shown in *Figure 1280* and *Table 1235*.

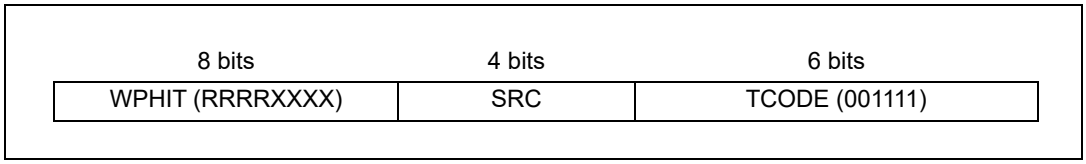


Figure 1280. Watchpoint message format (fixed length = 18 bits)

Table 1235. Watchpoint source description

Watchpoint source (8-bits)	Watchpoint description
RRRRXXX1(1)	External Watchpoint #1(device defined)
RRRRXX1X	External Watchpoint #2 (device defined)
RRRRX1XX	Internal Watchpoint #1 (BWA1 match)
RRRR1XXX	Internal Watchpoint #2 (BWA2 match)

1. R = reserved for future use (insert 0s).

62.6.2.2 Watchpoint error message

An error message occurs when a new message cannot be queued due to the message queue being full. The FIFO discards messages until it has completely emptied the queue. Once emptied, an error message is queued. The error encoding indicates which type(s) of messages attempted to be queued while the FIFO was being emptied.

If only a watchpoint message attempts to enter the queue while it is being emptied, the error message incorporates the watchpoint only error encoding (ECODE = 00000000000001). If a data trace message also attempts to enter the queue while it is being emptied, the error message incorporates error encoding (00000000000011). Error information is messaged out in the format shown in *Figure 1281*.

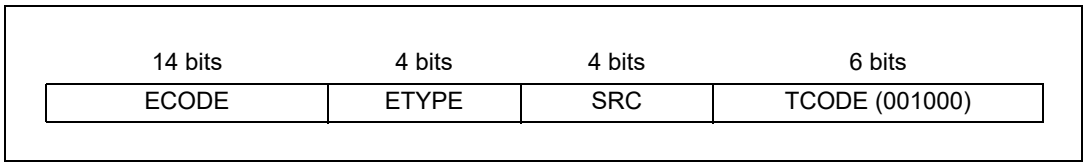


Figure 1281. Error message format (fixed length = 28 bits)

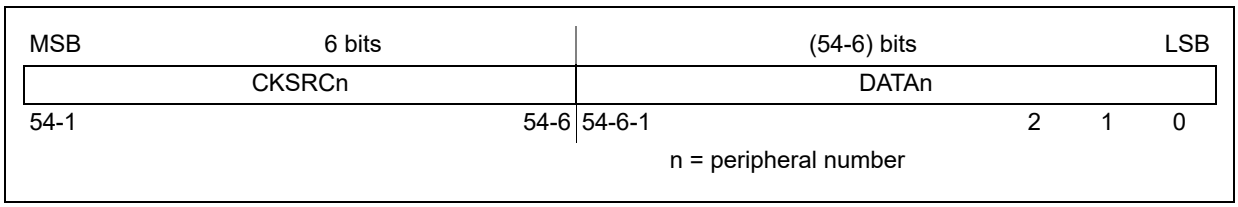


Figure 1282. Peripheral DATA bus fields

63 Cyclic Redundancy Check (CRC) Unit

63.1 Introduction

The Cyclic Redundancy Check (CRC) computing unit is dedicated to the computation of CRC, off-loading this work from the CPU. Each context has a separate CRC computation engine in order to allow the concurrent computation of the CRC of multiple data streams. The CRC computation is performed at speed without wait-state insertion. Bit-swap and bit-inversion operations can be applied on the final CRC signature. Each context can be configured with one of four hard-wired polynomials, normally used for most of the standard communication protocols. The data stream supports multiple data width (byte/halfword/word) formats.

63.2 Main features

- One contexts for the concurrent CRC computation
- Separate CRC engine for each context
- Zero-wait states during the CRC computation (pipeline scheme)
- Four hard-wired polynomials (two CRC-8, CRC-32 Ethernet, and CRC-16-CCITT)
  - Support for byte, halfword, or word width of the input data stream

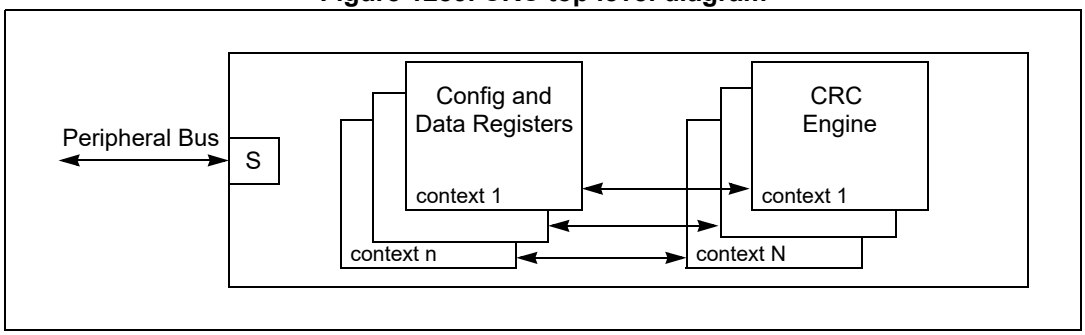
63.2.1 Standard features

- Peripheral bus interface
- CRC-8 VDA CAN
- CRC-16-CCITT
- CRC-32 Ethernet
- CRC-8-H2F Autosar polynomial

63.3 Block diagram

The top level diagram of the CRC module is given in *Figure 1283*.

Figure 1283. CRC top level diagram



1. Where context number, n = 1.

63.4 Signal description

The CRC module does not generate any external signals.

63.4.1 Peripheral bus interface

The peripheral bus interface is a slave bus used for configuration and data streaming (CRC computation) purposes via CPU or DMA. The following bus operations (contiguous byte enables) are supported:

- Word (32 bits) data write/read operations to any registers
- Low and high halfword (16 bits, data[31:16] or data[15:0]) data write/read operations to any registers
- Byte (8 bits, data[31:24], data[23:16], data[15:8], or data[7:0]) data write/read operations to any registers

Any other operation (free byte enables or other operations) must be avoided.

The CRC module generates a transfer error in the following cases:

- Any write/read access to the register addresses not mapped on the peripheral but included in the address space of the peripheral
- Any write/read operation different from byte/halfword/word (free byte enables or other operations) on each register

The registers of the CRC module are accessible (read/write) in each access mode: user, supervisor, and test.

The following summarizes bus operation performance:

- Zero wait states (single bus cycle) for each write/read operation to the CRC_CFG and CRC_INP registers
- Zero wait states (single bus cycle) for each write operation to the CRC_CSTAT register
- Double wait states (3 bus cycles) for each read operation to the CRC_CSTAT or CRC_OUTP registers immediately following (next clock cycle) a write operation to the CRC_CSTAT, CRC_INP, or CRC_CFG registers belonging to the same context; in all the other cases, no wait states are inserted

63.5 Memory map and registers

This section describes, in address-order, all the CRC registers.

63.5.1 Register description

Table 1236 shows the memory map for the CRC registers.

Table 1236. CRC unit memory map

Offset	Register	Register classification(1)	Location
CRC registers per context (n=0 to CRC_CNTX_NUM - 1 with CRC_CNTX_NUM = 1)			
0x000 + *n**0x10	CRC Configuration Register *n* (CRC_CFG*n*)	Saf-Lat	*Section 63.5.1.1*
0x004 + *n**0x10	CRC Input Register *n* (CRC_INP*n*)	Saf-Relv	*Section 63.5.1.2*
0x008 + *n**0x10	CRC Current Status Register *n* (CRC_CSTAT*n*)	Saf-Relv	*Section 63.5.1.3*
0x00C + *n**0x10	CRC Output Register *n* (CRC_OUTP*n*)	Saf-Relv	*Section 63.5.1.4*

1. No-Saf = Not safety relevant, Saf-Lat = Safety Latent, Saf-Relv = Safety Relevant, Saf-Crit = Safety Critical.

63.5.1.1 CRC Configuration Register *n* (CRC_CFG*n*)

Offset: 0x000 + *n* *0x10 (*n* = 0 to CRC_CNTX_NUM - 1) Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	SWAP_BYTEWISE	SWAP_BITWISE	POLYG		SWAP	INV
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	f(n)(1)	0	0

Figure 1284. CRC Configuration Register *n* (CRC_CFG*n*)

1. Reset value is a function of n. When n%2 = 0, reset value = 1. When n%2 = 1, reset value = 0.

Table 1237. CRC_CFG*n* field descriptions

Field	Description
26 SWAP_BYTEWISE	SWAP_BYTEWISE 0 No swap 1 Byte wise (MSB to LSB, LSB to MSB) bytes swap performed on CRC_INP input data internally for CRC-32 polynomial calculations **Note:** INV and SWAP bits are set to '1' for CRC-32 polynomial calculations.
27 SWAP_BITWISE	SWAP_BITWISE 0 No swap 1 Bit wise (MSB to LSB, LSB to MSB) bits swap performed on CRC_INP input data internally for CRC-16 polynomial calculations

Table 1237. CRC_CFG*n* field descriptions (continued)

Field	Description
28:29 POLYG	Polynomial selection 00 CRC-CCITT polynomial 01 CRC-32 polynomial 10 CRC-8 polynomial 11 CRC-8H2F autosar polynomial This bit can be read and written by software. **Note:** This bit can be written only during the configuration phase.
30 SWAP	SWAP selection 0 No swap selection applied on the CRC_OUTP content 1 Swap selection (MSB to LSB, LSB to MSB) applied on the CRC_OUTP content. The swap operation is a bit-by-bit rotation of the content. In case of CRC-CCITT polynomial, the swap operation is applied on the 16 LSB bits. This bit can be read and written by software. **Note:** This bit can be written only during the configuration phase.
31 INV	INV selection 0 No inversion selection applied on the CRC_OUTP content 1 Inversion selection (bit x bit) applied on the CRC_OUTP content. The inversion operation is a complement or negation of the content. In case of CRC-CCITT polynomial the inversion operation is applied on the 16 LSB bits. This bit can be read and written by software. **Note:** This bit can be written only during the configuration phase.

63.5.1.2 CRC Input Register *n* (CRC_INP*n*)

Offset: 0x004 + *n* * 0x10 (*n* = 0 to CRC_CNTX_NUM - 1) Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	INP															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	INP															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1285. CRC Input Register *n* (CRC_INP*n*)

Table 1238. CRC_INP*n* field descriptions

Field	Description
0:31 INP	Input data for the CRC computation The INP register can be written at byte, halfword (high and low), or word in any sequence. In case of halfword write operation, the bytes must be contiguous. Only the bits written are fed to the CRC engine. This register can be read and written by software.

63.5.1.3 CRC Current Status Register *n* (CRC_CSTAT*n*)

Offset: 0x008 + *n* * 0x10 (*n* = 0 to CRC_CNTX_NUM - 1) Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	CSTAT															
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	CSTAT															
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Figure 1286. CRC Current Status Register *n* (CRC_CSTAT*n*)

Table 1239. CRC_CSTATn field descriptions

Field	Description
0:31 CSTAT	Status of the CRC signature The CSTAT register includes the current status of the CRC signature. No bit swap and inversion are applied to this register. In the case of the CRC-CCITT polynomial, only the16 LSB bits are significative. The 16 MSB bits are tied at 0b during the computation. In the case of the CRC-8 polynomial, only the 8 LSB bits are significative. The 24 MSB bits are tied at 0b during the computation. The CSTAT register can be written at byte, halfword, or word. This register can be read and written by software. **Note:** This register can be written only during the configuration phase.

63.5.1.4 CRC Output Register *n* (CRC_OUTP*n*)

Offset: 0x00C + *n*0x10 (*n* = 0 to CRC_CNTX_NUM - 1) Access: User read-only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	OUTP															
W																
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	OUTP															
W																
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Figure 1287. CRC Output Register *n* (CRC_OUTP*n*)

Table 1240. CRC_OUTP*n* field descriptions

Field	Description
0:31 OUTP	Final CRC signature The OUTP register includes the final signature corresponding to the CRC_CSTAT register value eventually swapped and inverted. In the case of the CRC-CCITT polynomial, only the16 LSB bits are significative. The 16 MSB bits are tied at 0b during the computation. In the case of the CRC-8 polynomial, only the 8 LSB bits are significative. The 24 MSB bits are tied at 0b during the computation. This register can be read by software.

63.6 Functional description

The CRC module supports the CRC computation for each context. Each context has its own complete set of registers, including the CRC engine. The data flow of each context can be interleaved. The data stream can be structured as a sequence of bytes, halfwords, or words. The input data sequence is provided, eventually mixing the data formats (byte, halfword, and word), writing to the input data register (CRC_INP).

The data stream is generally executed by *n* concurrent DMA data transfers (mem2mem) where *n* is less than or equal to the number of contexts.

- The standard generator polynomials are given in *Equation 73*, *Equation 74* and *Equation 75* for the CRC computation of each context. Two separate registers are available: CRC_INP for writing data and CRC_CSTAT for retrieving the (accumulated up to now) CRC.

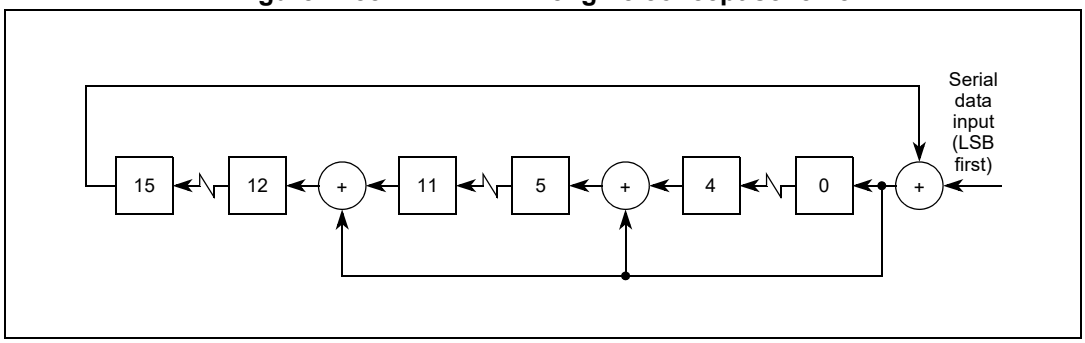
Equation 73 CRC8 VDA CAN:
$$x^8 + x^4 + x^3 + x^2 + 1$$

Equation 74 CRC-CCITT (x.25 protocol):
$$x^{16} + x^{12} + x^5 + 1$$

Equation 75 CRC-32 (Ethernet protocol):
$$x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$$

Equation 76 CRC-8-H2F (Autosar 4.0 protocol):
$$x^8 + x^5 + x^3 + x^2 + x + 1$$

Figure 1288. CRC-CCITT engine concept scheme

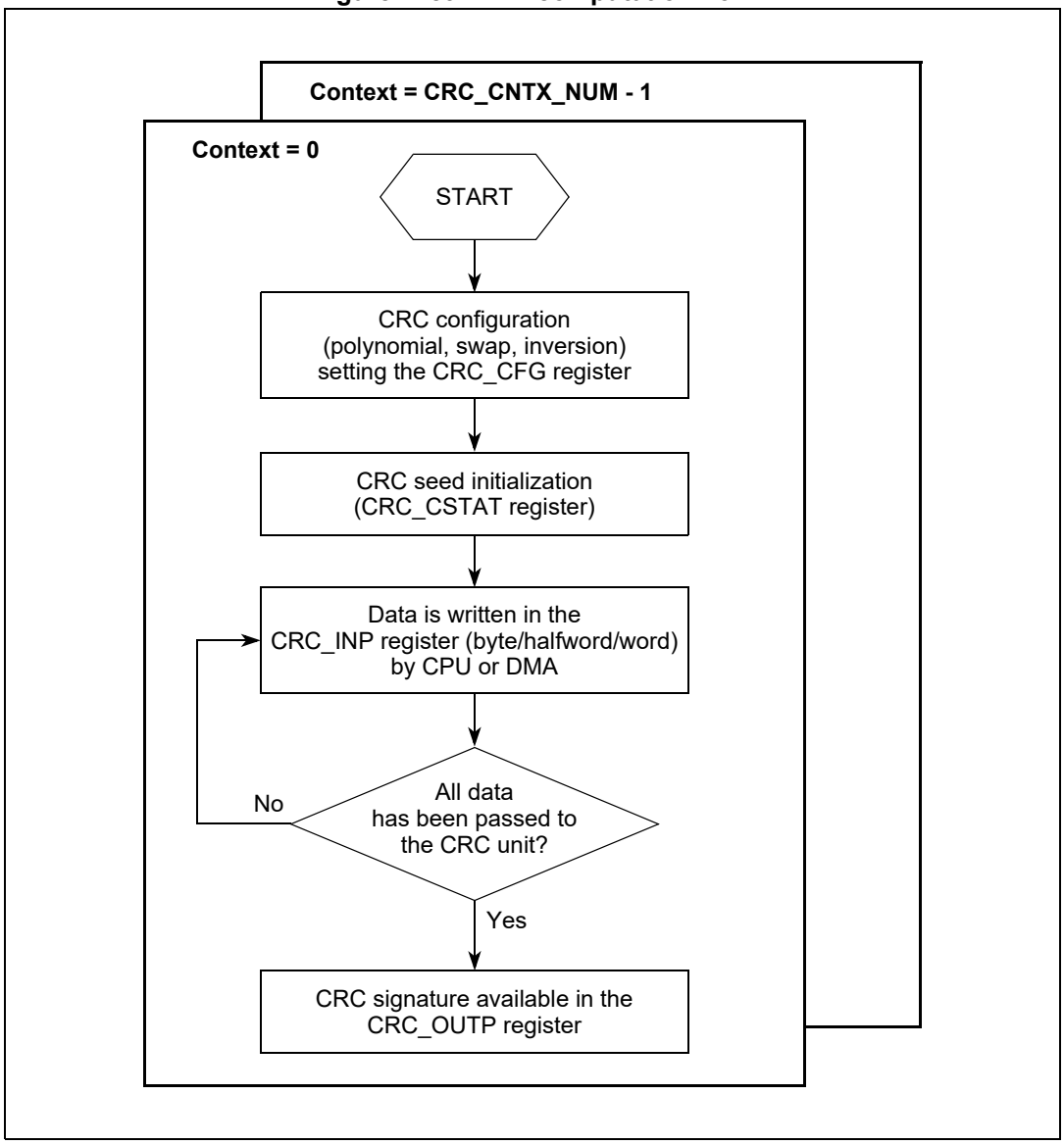


The initial seed value of the CRC can be programmed by initializing the CRC_CSTAT register. The concept scheme (serial data loading) of the CRC engine is given in *Figure 1288* for the CRC-CCITT. The design implementation executes the CRC computation in a single clock cycle (parallel data loading). A pipeline scheme has been adopted to decouple the peripheral bus interface from the CRC engine in order to allow the computation of the CRC at speed (zero wait states).

If the CRC signature is used for encapsulation in the data frame of a communication protocol (such as SPI), a bit swap (MSB → LSB, LSB → MSB) or bit inversion of the final CRC signature, or both can be applied (CRC_OUTP register) before transmitting the CRC.

Use of the CRC module is summarized in the flow chart given in *Figure 1289*.

Figure 1289. CRC computation flow



63.7 Use cases

63.7.1 Programming example

The number of contexts depends on the application. The overall number of contexts for the CRC peripheral depends on the number of peripherals that concurrently require intervention by the CRC module. Two main use cases are considered:

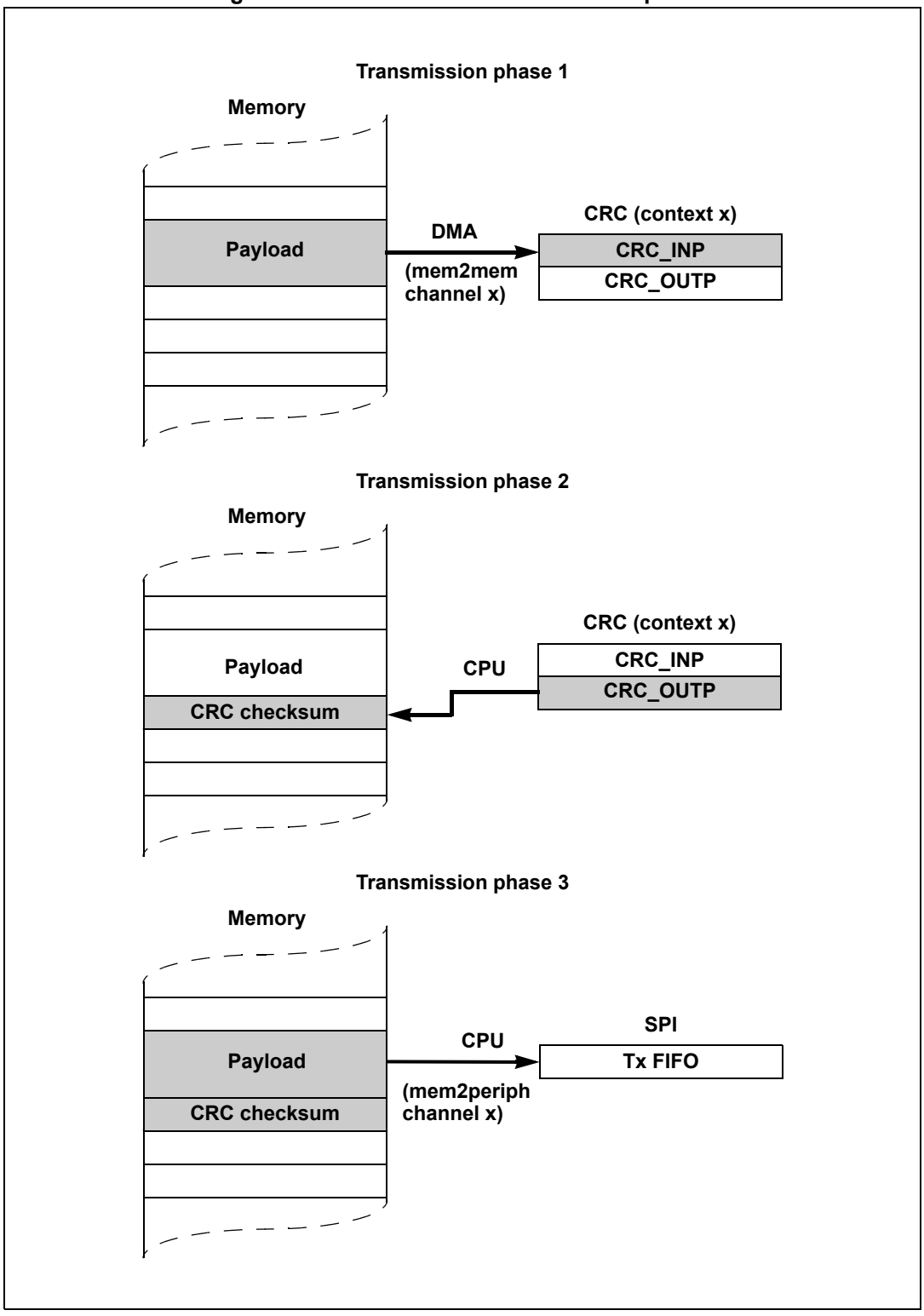
- Calculation of the CRC of the configuration registers during the process safety time
- Calculation of the CRC on the incoming/outcoming frames for the communication protocols (not protected with CRC by definition of the protocol itself) used as a safety-relevant peripheral

The signature of the configuration registers is computed correctly only if these registers do not contain any status bit.

Assuming that the DMA engine has *n* channels (greater than or equal to the number of contexts) configurable for the following types of data transfer: mem2mem, periph2mem, mem2periph; then this sequence, as shown in *Figure 1290*, is applied to manage the transmission data flow:

1. DMA/CRC module configuration (context x, channel x) by CPU
2. Payload transfer from the MEM to the CRC module (CRC_INP register) after MSB → LSB change (input mirroring) to calculate the CRC signature (phase1) by DMA (mem2mem data transfer, channel x)
3. CRC signature copy from the CRC module (CRC_OUTP register) to the MEM (phase 2) by CPU
4. Data block (payload + CRC) transfer from the MEM to the PERIPH module (for example, SPI Tx FIFO) (phase 3) by DMA (mem2periph data transfer, channel x)

Figure 1290. DMA-CRC transmission sequence



63.7.2 Register programming

- INV or SWAP, or both (affecting the output only) can be applied to CRC-8, CRC-16 and CRC-32. Both can be applied together.
- SWAP_BITWISE (affecting the input only) can be applied for all polynomials if required by an application.
- SWAP_BYTEWISE (affecting the input only) can be applied for CRC-16 and CRC-32 polynomials only.
- SWAP_BITWISE has priority over SWAP_BYTEWISE when both are applied together.
- When generating CRC-32 for the Ethernet standard the user needs to set SWAP_BYTEWISE together with INV and SWAP.
- When generating CRC-16 the user needs to set SWAP_BITWISE bit.

64 Memory Error Management Unit (MEMU)

64.1 Introduction

The MEMU is responsible for collection and reporting of error events associated with ECC (Error Correction Code) logic used on:

- SRAM
- Peripheral RAM
- Flash memory.

When any of the following events occur the MEMU receives an error signal which causes an event to be recorded and corresponding error flags to be set and reported to FCCU (Fault Collection and Control Unit).

- Correctable Error: It comprises the following:
  - Single/Double bit error in the data part that is detected via ECC for flash memory. Reporting of single/double bit error as correctable error depends on CTRL.FLASH_ERR_SELECT configuration.
  - Single Bit error in the data part that is detected via MBIST on any RAM.
- Uncorrectable Error: It comprises the following:
  - Double/Multiple(more than two) bit error that is detected via ECC for Flash Memory. Reporting of double bit error as uncorrectable error depends on CTRL.FLASH_ERR_SELECTION configuration.
  - Multiple bit error that is detected via MBIST on any SRAM. Based on MBIST implementation, the MBIST might report several correctable errors in the same word instead of an uncorrectable one. It is the task of the software after MBIST to aggregate these reports and detect the uncorrectable error.
  - Addressing errors and unused data bit errors detected by ECC logic.

The MEMU system connections are chip-specific; see *Chapter 7: Device configuration* that describes how modules are configured and connected.

When multiple errors are indicated from various sources at same instant, an Overflow can be indicated by the MEMU to the FCCU. Overflow can also be indicated if the reporting table entries are full and a new unique error is reported by the system.

MEMU processes the errors based on whether they are correctable or uncorrectable (from either ECC or MBIST logic), independent of the source generating these errors.

The MEMU has multiple instances of the basic reporting block; one each for:

- System RAM ECC and MBIST
- Peripheral SRAM ECC
- Flash memory ECC.

Each instance has two reporting tables–Correctable errors are reported in one table while the uncorrectable are reported in the other.

See *Section 64.4: Design overview* for the reporting block details.

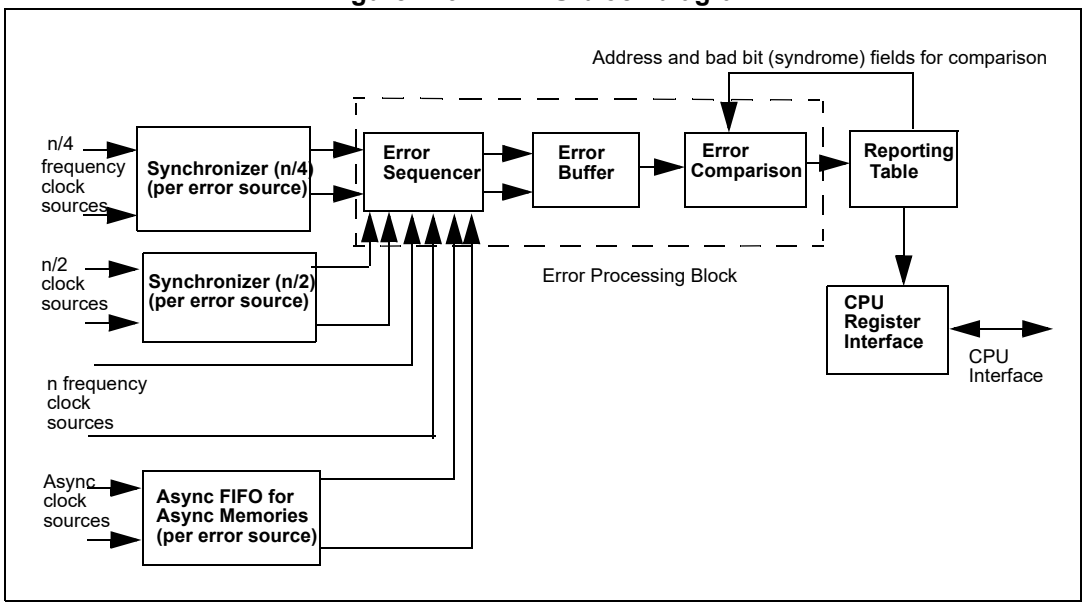
64.2 Features

The MEMU has the following features:

- Supports up to 128 error sources per Reporting Block (System RAM ECC and MBIST, Peripheral SRAM ECC or flash memory ECC): The number of error sources are chip-specific; see *Chapter 7: Device configuration* that describes how modules are configured and connected.
- Support for error reporting from the following category of error sources (both for ECC and MBIST):
  - System RAM
  - Peripheral SRAM
  - Flash memory
- Unique reported errors are logged into the module's reporting table which is accessible to CPU via memory mapped CPU register programming interface.
- MEMU handles overflow during error assertion and reports status accordingly.
- Unique errors are stored in correctable and non correctable section of the reporting table and corresponding indication is generated to the FCCU.
- CPU can program the known errors into the reporting table to avoid their re-reporting by MEMU.
- CPU can clear the reporting table errors.
- The reporting table for uncorrectable errors supports only 1 entry.

64.3 Block diagram

Figure 1291. MEMU block diagram



64.4 Design overview

The MEMU block has three instantiations of the basic reporting block along with a common CPU interface for the user software to access the storage block.

The MEMU should be enabled on power-on reset. The user software can write known error addresses into the reporting table to prevent reporting of those errors addresses to FCCU should they later be accessed in operation. All writes into the reporting table by the CPU that assert the Valid Bit (of the respective reporting table entry) will be indicated as a new error detected and will be reported to the FCCU. This feature can also be used as self-checking option of the MEMU and the MEMU-FCCU connection.

The MEMU design does not limit any particular sequence or errors. See *Section 64.7.3: Handling overflows (Multiple error reporting)* for details on overflow behaviors.

The basic algorithm flow/architecture for error reporting in the MEMU is summarized below:

1. Synchronizer block

   For asynchronous inputs, a FIFO-based structure (per asynchronous channel) is implemented. All the error inputs (correctable error reported, uncorrectable error reported, error address and bad bits/syndrome) are concatenated and stored in the FIFO. While storing the entry in the FIFO, a duplicity check is performed with the previously stored entry (the entry which was stored in the last cycle). If both the entries are duplicate the new entry is deleted and the pointers are rolled back to the old value, otherwise the new entry is stored. Whenever there is an overflow (FULL condition) in the Asynchronous FIFO, it is denoted by setting the Error Buffer Overflow or the EBO flag to the FCCU and the corresponding bit in the Concurrent Overflow Register is set.

   For synchronous inputs, inputs are sampled whenever the corresponding clock synchronizing signal is asserted for a particular channel.

2. Error Processing block

   This block is used to process the incoming errors coming from the device.

   – Error Sequencer is used for sequencing the errors so that these are processed one per clock. In case of more than one errors detected by the sequencer logic, one is sent for the processing (error comparison block), the other is stored in the error buffer and the rest are marked as overflows in the Concurrent overflow register.

   – All the sequences of incoming errors are supported. However, in case of multiple errors being asserted in back to back clock cycles, MEMU will process the errors being forwarded by the sequencer logic and register the rest of the errors as Concurrent Overflows in the Concurrent Overflow Register.

   – While storing the error in the error buffer, uniqueness is determined by comparing the error address, correctable or uncorrectable error type and the bad bit/syndrome (depending on ECC or MBIST logic) of the incoming error with the one stored in buffer. If the error buffer is full, and the incoming error to be stored in the error buffer is not yet in the error table, Error Buffer Overflow (EBO) is registered and the corresponding bit in the Concurrent Overflow Register is set, else the incoming error is dropped.

   – While doing the error buffer uniqueness check, the type of error (Correctable Error: CE or Uncorrectable Error: UCE) is also compared, hence even if the Error Address and the Bad bit/syndrome fields (Depending on whether the incoming

error is generated from ECC or MBIST logic) of the incoming error matches with the error buffer, if the type of errors (CE and UCE) differ they are treated as unique

– Determine the type and uniqueness of errors in the error comparison block and store them in the correct reporting table (correctable or uncorrectable)

– Generate signals to assert the necessary flags. Be aware that it takes several MEMU clock cycles (<=5 unless the error gets temporarily stored in the error buffer due to a collision) for an error report to cause a signal to be sent to the FCCU.

3. Reporting table

This is a pre-defined table in the MEMU to store the details for the device. For the reporting table if the entry is unique it is stored in the reporting table. If the entry is not unique it is not stored and discarded.

The MEMU asserts proper flags when error is stored in the reporting table or an overflow is asserted. Flags are also asserted when the software writes the VLD bit to the reporting table.

The CPU can clear the flags signaling errors or overflows to the FCCU directly by writing 1 at the corresponding valid bit of the status register (*Section 64.6.2.4: System RAM correctable error reporting table status register (SYS_RAM_CERR_STSn)* or *Section 64.6.2.6: System RAM uncorrectable error reporting table status register (SYS_RAM_UNCERR_STS) on page 2203*). The clearing of flags is totally independent from the status of VLD bits in the reporting table.

The reporting table sizes are chip-specific; see *Chapter 7: Device configuration* that describes how modules are configured and connected.

4. Register block

The CPU register programming interface provides the memory-mapped registers necessary for the CPU to access the reporting table and other control and status registers for the modules. Flags to FCCU will be asserted on write by CPU to the valid bit or by MEMU itself if a unique error is stored. However, in the situation when both MEMU and the CPU are accessing the same Register location, a wait cycle will be signaled to the CPU and the CPU command will be completed in the next clock cycle.

The software, in principle, is not supposed to write VLD to 1 if it is already 1.

The CPU register interface provides the decoding of the peripheral signals to allow access to the reporting table and other registers in module.

Figure 1292. MEMU error buffer uniqueness check flow chart

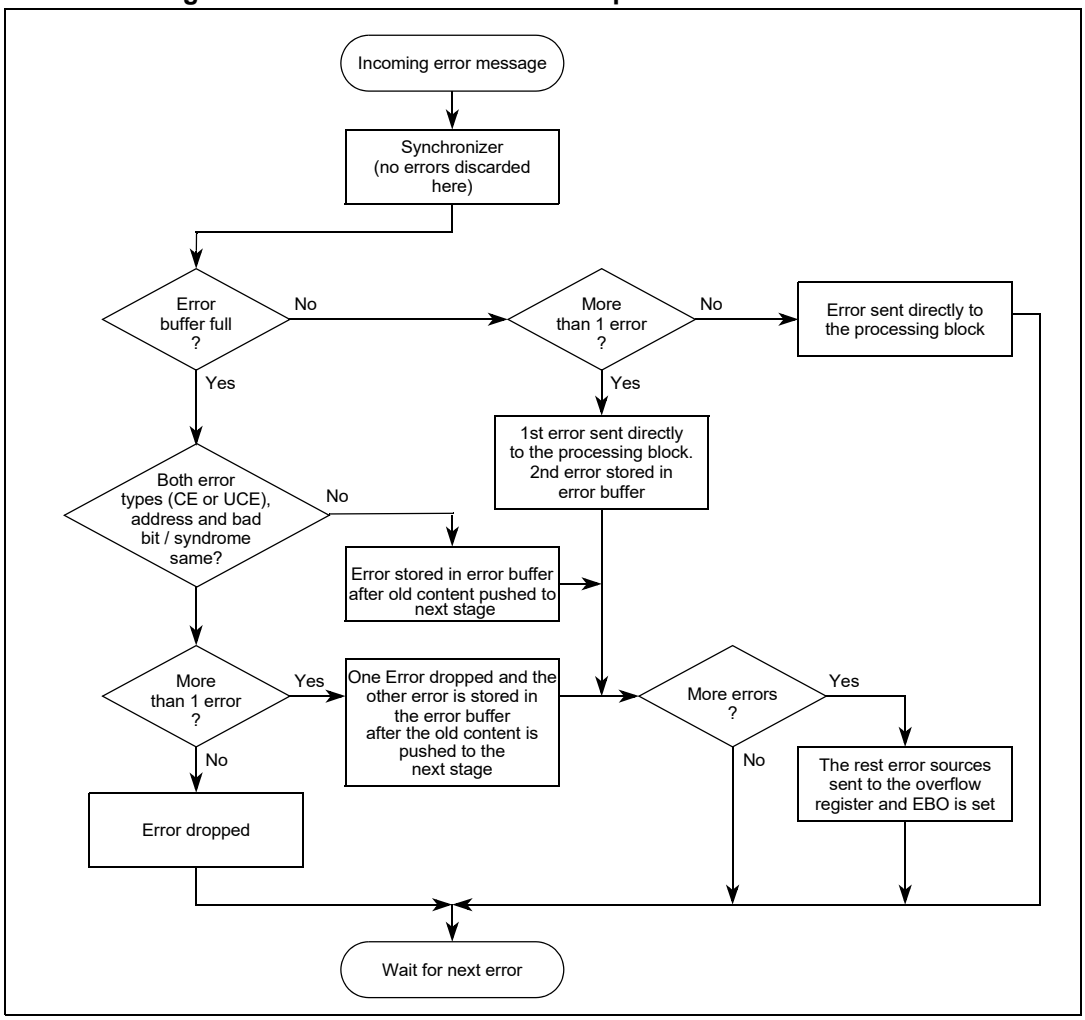
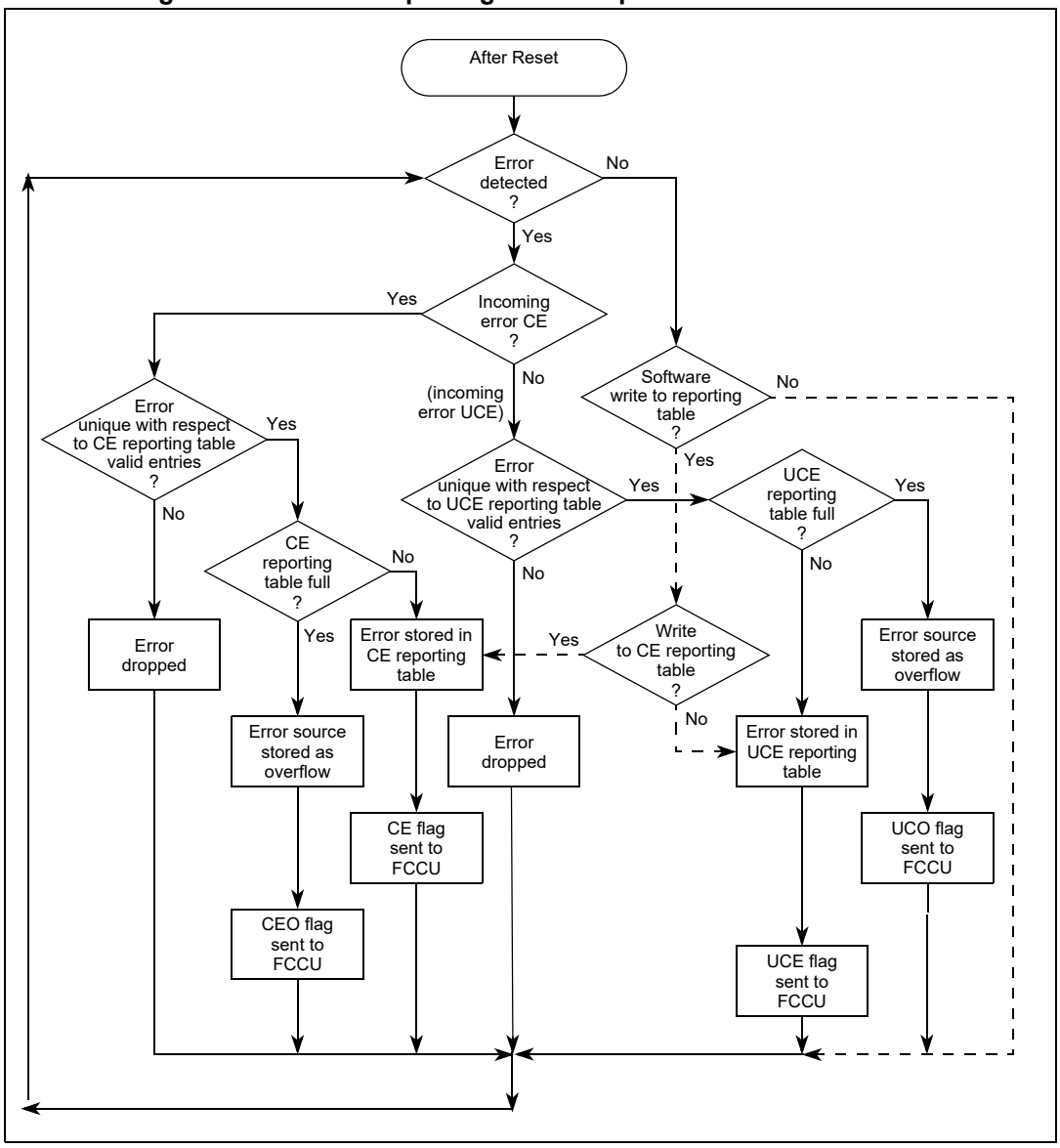


Figure 1293. MEMU reporting table uniqueness check flow chart



64.5 External signal description

MEMU has no external signals.

64.6 Memory map and register definition

64.6.1 Memory map

The memory map comprises 32-bit aligned registers that can be accessed via 8-bit,16-bit, or 32-bit accesses. Write access to reserved address locations will generate a transfer error. Read data from reserved locations will also generate a transfer error and the read data bus will return all 0's.

Note: The module does not check for correctness of the programmed values in registers. Software must ensure that the correct values are being written.

The Control, Error Flag and Debug registers are common for all MEMU error reporting blocks. The correctable error status, correctable error address, uncorrectable error status, uncorrectable error address and concurrent error overflow registers are replicated for each MEMU instance.

Note: The actual availability, number of registers and number of reported errors in the reporting table are chip-specific; see Chapter 7: Device configuration that describes how modules are configured and connected.

Table 1241. MEMU memory map

Offset or address	Register	Access	Reset value	Page
0x0000	Control register (CTRL)	R/W	0x0000_0000	*Section 64.6.2.1*
0x0004	Error Flag register (ERR_FLAG)	R/W	0x0000_0000	*Section 64.6.2.2*
0x000C	Debug register (DEBUG)	R/W	0x0000_0000	*Section 64.6.2.3*
0x0020 + *n**0x8 (*n* = 0 to 10 - 1)	System RAM correctable error reporting table status registers (SYS_RAM_CERR_STS*n*)	R/W	0x0000_0000	*Section 64.6.2.4*
0x0024 + *n**0x8 (*n* = 0 to 10 - 1)	System RAM correctable error reporting table address registers (SYS_RAM_CERR_ADDR*n*)	R/W	0x0000_0000	*Section 64.6.2.5*
0x0028 + (10 - 1)*0x8	System RAM uncorrectable error reporting table status register (SYS_RAM_UNCERR_STS)	R/W	0x0000_0000	*Section 64.6.2.6*
0x002C + (10 - 1)*0x8	System RAM uncorrectable error reporting table address register (SYS_RAM_UNCERR_ADDR)	R/W	0x0000_0000	*Section 64.6.2.7*
0x0030 + *n**0x4 + (10 - 1)*0x8 (*n* = 0 to 28÷32 - 1)(1)	System RAM concurrent overflow registers(2) (SYS_RAM_OFLW*n*)	w1c	0x0000_0000	*Section 64.6.2.8*
0x0620 + *n**0x8 (*n* = 0 to 2 - 1)	Peripheral RAM correctable error reporting table status registers (PERIPH_RAM_CERR_STS*n*)	R/W	0x0000_0000	*Section 64.6.2.9*
0x0624 + *n**0x8 (*n* = 0 to 2 - 1)	Peripheral RAM correctable error reporting table address registers (PERIPH_RAM_CERR_ADDR*n*)	R/W	0x0000_0000	*Section 64.6.2.10*
0x0628 + (2 - 1)*0x8	Peripheral RAM uncorrectable error reporting table status register (PERIPH_RAM_UNCERR_STS)	R/W	0x0000_0000	*Section 64.6.2.11*
0x062C + (2 - 1)*0x8	Peripheral RAM uncorrectable error reporting table address register (PERIPH_RAM_UNCERR_ADDR)	R/W	0x0000_0000	*Section 64.6.2.12*
0x0630 + *n**0x4 + (2 - 1)*0x8 (*n* = 0 to 6÷32 - 1)(1)	Peripheral RAM concurrent overflow registers(1) (PERIPH_RAM_OFLW*n*)	w1c	0x0000_0000	*Section 64.6.2.13*

Table 1241. MEMU memory map (continued)

Offset or address	Register	Access	Reset value	Page
0x0C20 + *n**0x8 (*n* = 0 to 20 - 1)	Flash memory correctable error reporting table status registers (FLASH_CERR_STS*n*)	R/W	0x0000_0000	*Section 64.6.2.14*
0x0C24 + *n**0x8 (*n* = 0 to 20 - 1)	Flash memory correctable error reporting table address registers (FLASH_CERR_ADDR*n*)	R/W	0x0000_0000	*Section 64.6.2.15*
0x0C28 + (20 - 1)*8	Flash memory uncorrectable error reporting table status register (FLASH_UNCERR_STS)	R/W	0x0000_0000	*Section 64.6.2.16*
0x0C2C + (20 - 1)*8	Flash memory uncorrectable error reporting table address register (FLASH_UNCERR_ADDR)	R/W	0x0000_0000	*Section 64.6.2.17*
0x0C30 + *n**0x4 + (20 - 1)*0x8 (*n* = 0 to 2÷32 - 1)(1)	Flash memory concurrent overflow registers(1)(FLASH_OFLW*n*)	w1c	0x0000_0000	*Section 64.6.2.18*

1. In this calculation, round any fractional result of the division to the next highest integer. For example, if the division is 94 ÷ 32 = 2.9375 or 66 ÷ 32 = 2.0625, round the result to 3.
2. The corresponding error sources for the Concurrent Overflow Register (OFLW) are chip-specific; see the Device Configuration chapter that describes how modules are configured and connected.

64.6.2 Register descriptions

64.6.2.1 Control register (CTRL)

Offset: 0x0000 Access: R/W

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	SWR	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1294. Control register (CTRL)

Table 1242. CTRL field descriptions

Field	Description
16 SWR	Software Reset bit. Setting this will clear status flags and overflow register. However, the reporting table registers are not cleared. This bit will auto clear on next clock cycle. 0 No reset. 1 Reset asserted.

64.6.2.2 Error Flag register (ERR_FLAG)

Bits in the ERR_FLAG register indicate:

- A new entry in an error reporting table has been made (indicated by PR_CE, PR_UCE, F_CE, F_UCE, SR_CE, and SR_UCE)
- An overflow condition has been encountered when attempting to make a new entry in an error reporting table (indicated by SR_UCO, SR_CEO, F_UCO,F_CEO, PR_UCO, and PR_CEO)
- An overflow condition has been encountered in the internal error processing buffer of a MEMU reporting block.

Individual flags can be cleared by writing a '1'. This also resets the corresponding error indication to FCCU. Flags will not automatically reset if entries are removed from the reporting table. They have to be explicitly cleared by SW.

Offset: 0x0004 Access: R/W

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	PR_CE	PR_UCE	PR_CEO	PR_UCO	PR_EBO
W												w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	F_CE	F_UCE	F_CEO	F_UCO	F_EBO	0	0	0	SR_CE	SR_UCE	SR_CEO	SR_UCO	SR_EBO
W				w1c	w1c	w1c	w1c	w1c				w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1295. Error flag register (ERR_FLAG)

Bits SR_UCO, SR_CEO, F_UCO, F_CEO, PR_UCO, and PR_CEO are asserted when an overflow in the uncorrectable error reporting table is detected.

Table 1243. ERR_FLAG field descriptions

Field	Description
11 PR_CE	Peripheral RAM ECC Correctable Error detect flag. Indicates that a new and unique Peripheral RAM ECC correctable error is detected. Asserted when a new entry is inserted into the correctable error reporting table (either by CPU or module). 0 No new and unique error detected. 1 New entry in correctable error reporting table is made.
12 PR_UCE	Peripheral RAM ECC Uncorrectable Error Detect flag. Asserted when a new Peripheral RAM ECC uncorrectable error was detected or a new entry is inserted into the uncorrectable error reporting table is done (either by CPU or module). 0 No new and unique error detected. 1 New entry in uncorrectable error reporting table is made.

Table 1243. ERR_FLAG field descriptions (continued)

Field	Description
13 PR_CEO	Peripheral RAM ECC Correctable error Overflow flag. Asserted when the Peripheral RAM ECC correctable error reporting table is full and a new entry is attempted to be made to this table. 0 No Overflow. 1 Overflow in the correctable error reporting table detected.
14 PR_UCO	Peripheral RAM ECC Uncorrectable error Overflow flag. Asserted when the Peripheral RAM ECC uncorrectable error reporting table is full and a new entry is made to this table. 0 No Overflow. 1 Overflow in the uncorrectable error reporting table detected.
15 PR_EBO	Peripheral RAM ECC Error buffer Overflow flag. Asserted when the internal error processing buffer of Peripheral RAM ECC MEMU reporting block has overflowed. This can happen when too many errors are reported in a short interval of time for processing by MEMU. The channel(s) from which error reports were dropped due to the overflow are indicated in the XXXXX_OFLW registers. This field will also be set to ‘1’ if the input ASYNC FIFOs overflow. 0 No Overflow. 1 Overflow in the internal error processing buffer detected.
19 F_CE	Flash ECC Correctable Error Detect flag. Indicates that a new and unique Flash ECC correctable error was detected. Asserted when a new entry to the correctable error reporting table is done (either by CPU or module). 0 No new and unique error detected. 1 New entry in correctable error reporting table is made.
20 F_UCE	Flash ECC Uncorrectable Error Detect flag. Asserted when: – a new Flash ECC uncorrectable error was detected. – a new entry to the uncorrectable error reporting table is done (either by CPU or module) 0 No new and unique error detected. 1 New entry in uncorrectable error reporting table is made.
21 F_CEO	Flash ECC Correctable Error Overflow flag. Asserted when the Flash ECC correctable error reporting table is full and a new entry is made to this table. 0 No Overflow. 1 Overflow in the correctable error reporting table detected.
22 F_UCO	Flash ECC Uncorrectable Error Overflow flag. Asserted when the Flash ECC uncorrectable error reporting table is full and a new entry is made to this table. 0 No Overflow. 1 Overflow in the uncorrectable error reporting table detected.

Table 1243. ERR_FLAG field descriptions (continued)

Field	Description
23 F_EBO	Flash ECC Error buffer Overflow. Flag Asserted when the internal error processing buffer of Flash ECC MEMU reporting block has overflowed. This can happen when too many errors are reported in a short interval of time for MEMU to process them all. The channel(s) from which error reports were dropped due to the overflow are indicated in the XXXXX_OFLW registers. This field will also be set to '1' if the input ASYNC FIFOs overflow. 0 No Overflow. 1 Overflow in the internal error processing buffer detected.
27 SR_CE	System RAM ECC and MBIST Correctable Error detect flag. Indicates that a new and unique System RAM ECC and MBIST correctable error is detected. Asserted when a new entry to the correctable error reporting table is done (either by CPU or module). 0 No new and unique error detected. 1 New entry in correctable error reporting table is made.
28 SR_UCE	System RAM ECC and MBIST Uncorrectable Error Detect flag. Asserted when: – A new System RAM ECC and MBIST uncorrectable error is detected – A new entry to the uncorrectable error reporting table is done (either by CPU or module) 0 No new and unique error detected. 1 New entry in uncorrectable error reporting table is made.
29 SR_CEO	System RAM ECC and MBIST Correctable error Overflow flag. Asserted when the System RAM ECC and MBIST correctable error reporting table is full and a new entry is made to this table. 0 No Overflow. 1 Overflow in the correctable error reporting table detected.
30 SR_UCO	System RAM ECC and MBIST Uncorrectable error Overflow flag. Asserted when the System RAM ECC and MBIST uncorrectable error reporting table is full and a new entry is made to this table. 0 No Overflow. 1 Overflow in the uncorrectable error reporting table detected.
31 SR_EBO	System RAM ECC and MBIST Error buffer Overflow. Flag asserted when the internal error processing buffer of System RAM ECC and MBIST MEMU reporting block has overflowed. This can happen when too many errors are reported in a short interval of time for processing by MEMU. The channel(s) from which error reports were dropped due to the overflow are indicated in the XXXXX_OFLW registers. This field will also be set to '1' if the input ASYNC FIFOs overflow. 0 No Overflow. 1 Overflow in the internal error processing buffer detected.

64.6.2.3 Debug register (DEBUG)

This register is used to check the connections between MEMU & FCCU.

The error output of the MEMU towards FCCU shall stay set if forced until reset by SW via the MEMU in the normal way.

Offset 0x000C Access: R/W

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	FR_PR_CE	FR_PR_UCE	FR_PR_CEO	FR_PR_UCO	FR_PR_EBO
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	FR_F_CE	FR_F_UCE	FR_F_CEO	FR_F_UCO	FR_F_EBO	0	0	0	FR_SR_CE	FR_SR_UCE	FR_SR_CEO	FR_SR_UCO	FR_SR_EBO
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1296. Debug register (DEBUG)

Table 1244. DEBUG field descriptions

Field	Description
11 FR_PR_CE	Force Peripheral RAM Correctable Error detect flag. 0 Forcing is disabled. 1 Forces PR_CE flag going towards FCCU to 1.
12 FR_PR_UCE	Force Peripheral RAM Uncorrectable Error detect flag. 0 Forcing is disabled. 1 Forces PR_UCE flag going towards FCCU to 1.
13 FR_PR_CEO	Force Peripheral RAM Correctable Error overflow flag 0 Forcing is disabled. 1 Forces PR_CEO flag going towards FCCU to 1.
14 FR_PR_UCO	Forces Peripheral RAM Uncorrectable Error overflow flag. 0 Forcing is disabled. 1 Forces PR_UCO flag going towards FCCU to 1.
15 FR_PR_EBO	Forces Peripheral RAM Error Buffer Overflow Flag. 0 Forcing is disabled. 1 Forces PR_EBO flag going towards FCCU to 1.
19 FR_F_CE	Forces Flash Correctable Error detect flag 0 Forcing is disabled. 1 Forces F_CE flag going towards FCCU to 1.
20 FR_F_UCE	Forces Flash Uncorrectable Error detect flag 0 Forcing is disabled. 1 Forces F_UCE flag going towards FCCU to 1.
21 FR_F_CEO	Forces Flash Correctable Error overflow flag 0 Forcing is disabled. 1 Forces F_CEO flag going towards FCCU to 1.

Table 1244. DEBUG field descriptions (continued)

Field	Description
22 FR_F_UCO	Forces Flash Uncorrectable Error overflow flag 0 Forcing is disabled. 1 Forces F_UCO flag going towards FCCU to 1.
23 FR_F_EBO	Forces Flash Error buffer Overflow flag 0 Forcing is disabled. 1 Forces F_EBO flag going towards FCCU to 1.
27 FR_SR_CE	Forces System RAM Correctable Error detect flag 0 Forcing is disabled. 1 Forces SR_CE flag going towards FCCU to 1.
28 FR_SR_UCE	Forces System RAM Uncorrectable Error detect flag 0 Forcing is disabled. 1 Forces SR_UCE flag going towards FCCU to 1.
29 FR_SR_CEO	Forces System RAM Correctable Error overflow flag 0 Forcing is disabled. 1 Forces SR_CEO flag going towards FCCU to 1.
30 FR_SR_UCO	Forces System RAM Uncorrectable Error overflow flag 0 Forcing is disabled. 1 Forces SR_UCO flag going towards FCCU to 1.
31 FR_SR_EBO	Forces System RAM Error buffer Overflow Flag 0 Forcing is disabled. 1 Forces SR_EBO flag going towards FCCU to 1.

64.6.2.4 System RAM correctable error reporting table status register (SYS_RAM_CERR_STS*n*)

Note: The reporting table is not cleared on software reset.

Offset: 0x0020 + *n**0x8 (*n* = 0 to (10 - 1)) Access: R/W

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	VLD	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	BAD_BIT							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1297. System RAM correctable error reporting table status register (SYS_RAM_CERR_STS*n*)

Table 1245. SYS_RAM_CERR_STS*n* field descriptions

Field	Description
0 VLD	Valid bit This indicates that the entry in reporting table is valid. Assertion of this bit causes the correctable error detect flag to be asserted. 0 Entry in table is invalid. 1 Entry in table is valid.
24:31 BAD_BIT	Bad bit field Indicates the bit position in RAM where the error is detected. This is also known as the syndrome. For non-BIST type of errors, this field is not used and reads as 0xFF. Any value other than 0xFF - Position of the bit in RAM where error is found. 0xFF - Invalid Bad Bit. This field should not be used when processing errors.

64.6.2.5 System RAM correctable error reporting table address register (SYS_RAM_CERR_ADDR*n*)

Offset: 0x0024 + *n**0x8 (*n* = 0 to (10 - 1)) Access: R/W

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	ERR_ADD															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	ERR_ADD															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1298. System RAM correctable error reporting table address register (SYS_RAM_CERR_ADDR*n*)

Table 1246. SYS_RAM_CERR_ADDR*n* field descriptions

Field	Description
0:31 ERR_ADD	Error address field Indicates the address on which the error was detected.

64.6.2.6 System RAM uncorrectable error reporting table status register (SYS_RAM_UNCERR_STS)

Note: The reporting table is not cleared on software reset.

Offset: 0x0028 + (10 - 1)*0x8 Access: R/W

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	VLD	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

Figure 1299. System RAM uncorrectable error reporting table status register (SYS_RAM_UNCERR_STS)

Table 1247. SYS_RAM_UNCERR_STS field descriptions

Field	Description
0 VLD	Valid bit This indicates that the entry in reporting table is valid. Assertion of this bit causes the uncorrectable error detect flag to be asserted. 0 Entry in table is invalid. 1 Entry in table is valid.

64.6.2.7 System RAM uncorrectable error reporting table address register (SYS_RAM_UNCERR_ADDR)

Note: The reporting table is not cleared on software reset.

Offset: 0x002C + (10 - 1)*0x8 Access: R/W

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	ERR_ADD															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	ERR_ADD															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1300. System RAM uncorrectable error reporting table address register (SYS_RAM_UNCERR_ADDR)

Table 1248. SYS_RAM_UNCERR_ADDR field descriptions

Field	Description
0:31 ERR_ADD	Error address field Indicates the address on which the error was detected. This register must be READ ONLY when its corresponding valid bit is not asserted.

64.6.2.8 System RAM concurrent overflow registers (SYS_RAM_OFLW*n*)

Note: The reporting table is not cleared on software reset.

Offset: 0x0030 + *n**0x4 + (10 - 1)*0x8 (*n* = 0 to (28÷32 - 1))(1) Access: R/W

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	OFLW															
W	w1c															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	OFLW															
W	w1c															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. In this calculation, round any fractional result of the division to the next highest integer. For example, if the division is 94 ÷ 32 = 2.9375 or 66 ÷ 32 = 2.0625, round the result to 3.

Figure 1301. System RAM concurrent overflow registers (SYS_RAM_OFLW*n*)

Table 1249. SYS_RAM_OFLW*n* field descriptions

Field	Description
0:31 OFLW	Overflow Bit Each bit corresponds to an error source and is asserted when any of the conditions mentioned in *Section 64.7.3: Handling overflows (Multiple error reporting)* occurs.

64.6.2.9 Peripheral RAM correctable error reporting table status register (PERIPH_RAM_CERR_STSn)

Note: The reporting table is not cleared on software reset.

Offset: 0x0620 + *n**0x8 (*n* = 0 to(2 - 1)) Access: R/W

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	VLD	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	BAD_BIT							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1302. Peripheral RAM correctable error reporting table status register (PERIPH_RAM_CERR_STS*n*)

Table 1250. PERIPH_RAM_CERR_STS*n* field descriptions

Field	Description
0 VLD	Valid bit This indicates that the entry in reporting table is valid. Assertion of this bit causes the correctable error detect flag to be asserted. 0 Entry in table is invalid. 1 Entry in table is valid.
24:31 BAD_BIT	Bad bit field Indicates the bit position in RAM where the error is detected. This is also known as the syndrome. For non-BIST type of errors, this field is not used and reads as 0xFF. Any value other than 0xFF - Position of the bit in RAM where error is found. 0xFF - Invalid Bad Bit. This field should not be used when processing errors.

64.6.2.10 Peripheral RAM correctable error reporting table address register (PERIPH_RAM_CERR_ADDR*n*)

Offset: 0x0624 + *n**0x8 (*n* = 0 to (2 - 1)) Access: R/W

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	ERR_ADD															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	ERR_ADD															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1303. Peripheral RAM correctable error reporting table address register (PERIPH_RAM_CERR_ADDR*n*)

Table 1251. PERIPH_RAM_CERR_ADDR*n* field descriptions

Field	Description
0:31 ERR_ADD	Error address field Indicates the address on which the error was detected.

64.6.2.11 Peripheral RAM uncorrectable error reporting table status register (PERIPH_RAM_UNCERR_STS)

Note: The reporting table is not cleared on software reset.

Offset: 0x0628 + (2 - 1)*0x8 Access: R/W

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	VLD	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

Figure 1304. Peripheral RAM uncorrectable error reporting table status register (PERIPH_RAM_UNCERR_STS)

Table 1252. PERIPH_RAM_UNCERR_STS field descriptions

Field	Description
0 VLD	Valid bit This indicates that the entry in reporting table is valid. Assertion of this bit causes the uncorrectable error detect flag to be asserted. 0 Entry in table is invalid. 1 Entry in table is valid.

64.6.2.12 Peripheral RAM uncorrectable error reporting table address register (PERIPH_RAM_UNCERR_ADDR)

Note: The reporting table is not cleared on software reset.

Offset: 0x062C + (2 - 1)*0x8 Access: R/W

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	ERR_ADD															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	ERR_ADD															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1305. Peripheral RAM uncorrectable error reporting table address register (PERIPH_RAM_UNCERR_ADDR)

Table 1253. PERIPH_RAM_UNCERR_ADDR field descriptions

Field	Description
0:31 ERR_ADD	Error address field Indicates the address on which the error was detected. This register must be READ ONLY when its corresponding valid bit is not asserted.

64.6.2.13 Peripheral RAM concurrent overflow registers (PERIPH_RAM_OFLW*n*)

Note: The reporting table is not cleared on software reset.

Offset: 0x0630 + *n**0x4 + (2 - 1)*0x8 (*n* = 0 to (6÷32 - 1))(1) Access: R/W

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	OFLW															
W	w1c															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	OFLW															
W	w1c															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. In this calculation, round any fractional result of the division to the next highest integer. For example, if the division is 94 ÷ 32 = 2.9375 or 66 ÷ 32 = 2.0625, round the result to 3.

Figure 1306. Peripheral RAM concurrent overflow registers (PERIPH_RAM_OFLW*n*)

Table 1254. PERIPH_RAM_OFLW*n* field descriptions

Field	Description
0:31 OFLW	Overflow Bit Each bit corresponds to an error source and is asserted when any of the conditions mentioned in *Section 64.7.3: Handling overflows (Multiple error reporting)* occurs.

64.6.2.14 Flash memory correctable error reporting table status register (FLASH_CERR_STS*n*)

Note: The reporting table is not cleared on software reset.

MEMU captures flash ecc error address on 128bit boundary.

Offset: 0x0C20 + *n**0x8 (*n* = 0 to (20 - 1)) Access: R/W

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	VLD	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	BAD_BIT							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1307. Flash memory correctable error reporting table status register (FLASH_CERR_STS*n*)

Table 1255. FLASH_CERR_STS*n* register field descriptions

Field	Description
0 VLD	Valid bit This indicates that the entry in reporting table is valid. Assertion of this bit causes the correctable error detect flag to be asserted. 0 Entry in table is invalid. 1 Entry in table is valid.
24:31 BAD_BIT	Bad bit field Indicates the bit position in FLASH where the error is detected. This is also known as the syndrome. For non-BIST type of errors, this field is not used and reads as 0xFF. Any value other than 0xFF - Position of the bit in FLASH where error is found. 0xFF Invalid Bad Bit. This field should not be used when processing errors.

64.6.2.15 Flash memory correctable error reporting table address register (FLASH_CERR_ADDR*n*)

Offset: 0x0C24 + *n**0x8 (*n* = 0 to (20 - 1)) Access: R/W

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	ERR_ADD															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	ERR_ADD															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1308. Flash memory correctable error reporting table address register (FLASH_CERR_ADDR*n*)

Table 1256. FLASH_CERR_ADDR*n* register field descriptions

Field	Description
0:31 ERR_ADD	Error address field Indicates the address on which the error was detected.

64.6.2.16 Flash memory uncorrectable error reporting table status register (FLASH_UNCERR_STS)

Note: The reporting table is not cleared on software reset.

Offset: 0x0C28 + (20 - 1)*0x8 Access: R/W

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	VLD	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

Figure 1309. Flash memory uncorrectable error reporting table status register (FLASH_UNCERR_STS)

Table 1257. UNCERR_STS field descriptions

Field	Description
0 VLD	Valid bit This indicates that the entry in reporting table is valid. Assertion of this bit causes the uncorrectable error detect flag to be asserted. 0 Entry in table is invalid. 1 Entry in table is valid.

64.6.2.17 Flash memory uncorrectable error reporting table address register (FLASH_UNCERR_ADDR)

Note: The reporting table is not cleared on software reset.

Offset: 0x0C2C + (20 - 1)*8 Access: R/W

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	ERR_ADD															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	ERR_ADD															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1310. Flash memory uncorrectable error reporting table address register (FLASH_UNCERR_ADDR)

Table 1258. FLASH_UNCERR_ADDR field descriptions

Field	Description
0:31 ERR_ADD	Error address field Indicates the address on which the error was detected. This register must be READ ONLY when its corresponding valid bit is not asserted.

64.6.2.18 Flash memory concurrent overflow registers (FLASH_OFLWn)

Note: The reporting table is not cleared on software reset.

Offset: 0x0C30 + *n**0x4 + (20 - 1)*0x8 (*n* = 0 to (2÷32 - 1))(1) Access: R/W

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	OFLW															
W	w1c															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	OFLW															
W	w1c															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. In this calculation, round any fractional result of the division to the next highest integer. For example, if the division is 94 ÷ 32 = 2.9375 or 66 ÷ 32 = 2.0625, round the result to 3.

Figure 1311. Flash memory concurrent overflow registers (FLASH_OFLWn)

Table 1259. FLASH_OFLWn field descriptions

Field	Description
0:31 OFLW	Overflow Bit Each bit corresponds to an error source and is asserted when any of the conditions mentioned in *Section 64.7.3: Handling overflows (Multiple error reporting)* occurs.

64.7 Functional description

In this section, the term “bad bit” and “syndrome” are equivalent, and are used to indicate the position of the bit in error.

64.7.1 Initializing MEMU

MEMU is always enabled and can be configured by user software for known errors which the system/application has collected over a period of time. The MEMU logic will not update the reporting table corresponding to the programmed values which have the VLD bit SET.

The user software can write into the reporting table and must make VLD bit = ‘1’ for the programmed entry to be valid. Any write by the CPU that asserts the VLD bit will cause a corresponding flag to FCCU.

The software can program the table with known error addresses by setting the valid bit and storing the corresponding error address.The sequence of programming to prevent a collision of reporting table entries written by the HW and the SW is as follows:

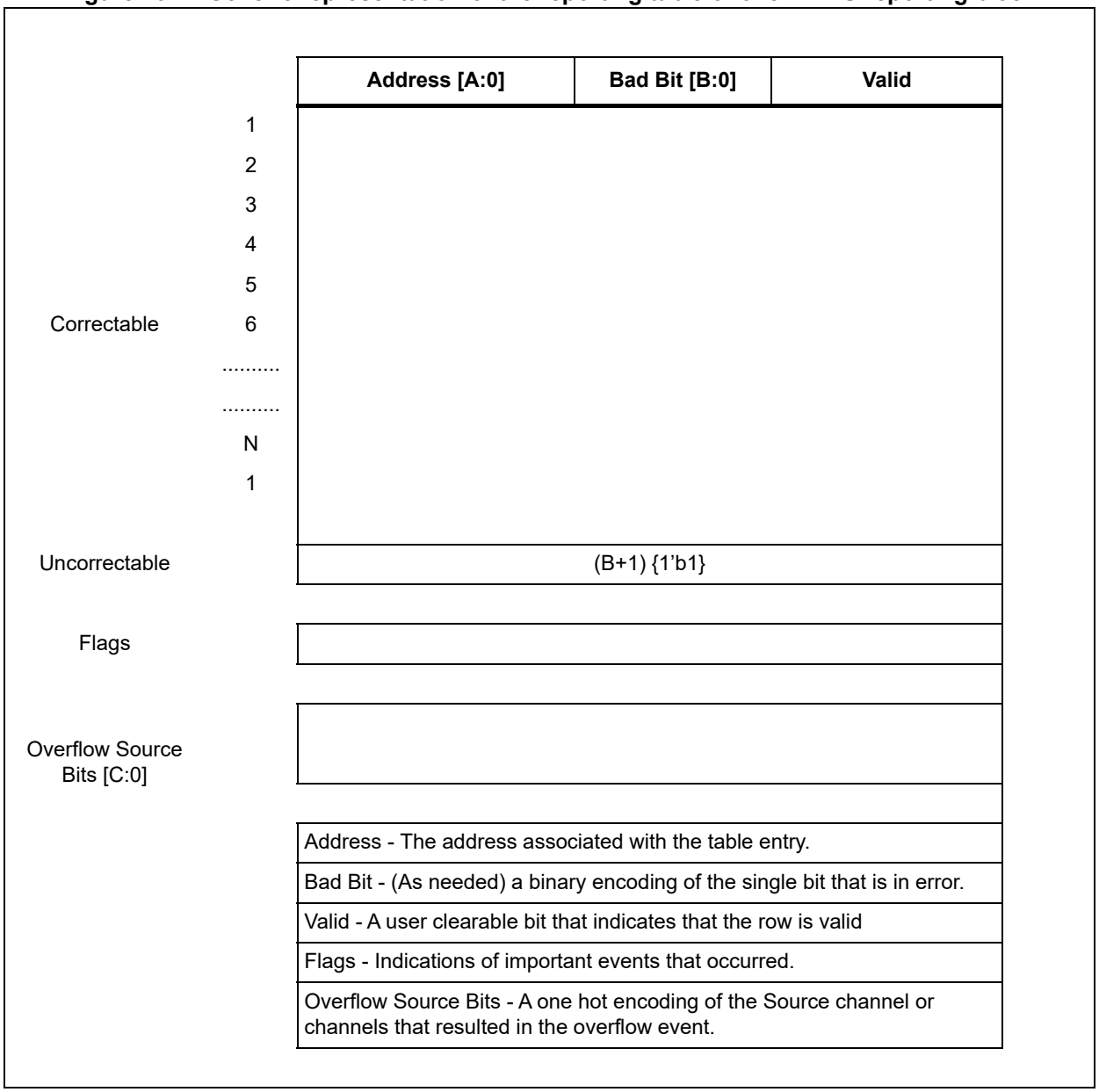
1. Check that the VLD bit is not set,
2. Write an invalid address/bad bit information,
3. Set VLD bit,
4. Check that the address still has an invalid value,
5. Write address/bad bit information to desired address.

If the FCCU is programmed to cause an IRQ on new MEMU table entries, steps 3 to 5 should be executed with interrupts disabled to prevent any error handling SW to read the invalid address written in step 2). Note that in the case the software is in between the 3rd and the 5th step, and the MEMU writes an error with the same address that Software wants to program, both the errors will be stored in the reporting table (duplicate entries).

64.7.2 Reading the reporting table

The software at any time can read the reporting table via the software programming interface. It should read the entries with the valid bit set and take necessary action. To invalidate any entry, the valid bit must be cleared. Reading an entry which does not have the VLD bit set will return invalid data.

Figure 1312. Generic representation of the reporting table of one MEMU reporting block



64.7.3 Handling overflows (Multiple error reporting)

When a bit in the overflow registers (either error buffer or reporting table though the overflow registers are same) is asserted it can indicate the occurrence of one of the following conditions:

1. Error buffer overflow–More than two memories reporting an error at the same instant or the input ASYNC FIFOs reach the FULL level (overflow)
   - More than 2 errors detected at same time. This would lead to overflow in the error buffer. MEMU can process only one error at a time while storing the other in the error buffer. So this would be indicated as an overflow. The uniqueness check in the input buffer does not “guarantee” that an ECC-supervised memory with only one error will not be marked as overflow source. If that one error occurs together with two (or more) additional errors at the same cycle it can get flagged as an overflow.
2. Correctable/Uncorrectable Error Overflow–Overflow in correctable and uncorrectable reporting table occurs only when a unique entry is to be stored but the tables are already full (all entries have valid bit set).
   - An ECC error which matches the address of an MBIST error is dropped (same as if it matched the address of an ECC error). In case, where a MBIST error comes with ECC error already in the table, both the address and bad bits are compared to determine uniqueness.
3. The Overflows will be stored in the bit-wise order which means that if the error bit 31 reports an error, the bit 31 of the overflow register will be set to ‘1’.

65 Fault Collection and Control Unit (FCCU)

65.1 Introduction

The Fault Collection and Control Unit (FCCU) offers a systematic approach to fault detection and control.The FCCU provides a hardware channel to collect errors and to place the device into a safety state when a failure in the device is detected. No CPU intervention is requested for collection and control operation.

65.2 Features

The FCCU offers a systematic approach to fault detection and control. The distinctive features of the module are:

- 1 to 128 recoverable faults management
- HW or SW fault recovery management
- Fault detection collection
- Fault injection (fake faults; see the implementation-specific details)
- Bi-stable, Dual Rail and Time Switching output protocols on EOUT
- Watchdog timer for the reconfiguration phase
- Redundant collection of hardware checker (for example RCCU) results
- Redundant collection of error information from safety relevant modules on the device
- Collection of test results
- Configurable and graded fault control
- Internal (SoC) reactions
  - ALARM state: interrupt request
  - FAULT state: long functional reset request pulse, short functional reset request pulse, NMI
- Internal reactions (independently configurable for each RF)
  - No reset reaction
  - IRQ
  - Short functional reset
  - Long functional reset
  - NMI
- External reaction (failure is reported to the outside world via one or more output pins)
- External reaction (fault state): EOUT signaling. Failure indication via the pin(s) is controlled by the FCCU
- Configuration lock: The FCCU’s configuration is lockable in two ways
  - Permanently locked until next reset or transiently locked until a specific key is written. FCCU gets transiently locked again if an invalid key is written into

FCCU_TRANS_LOCK register (a value different to 0xBC). Key to lock the FCCU permanently is 0xFF, written in the FCCU_PERMNT_LOCK register

- One of the failure indication pins is high to indicate operational (OK) state (in bi-stable operation mode). The above is not true in case the failure indication protocol is configured to be a toggling protocol that is dual rail and time switching
- After power-on the error out pins have high impedance[u]. FCCU goes to operational state only on software request
- In case of a failure event or on software request for Error pin indication, the pin(s) are set to faulty state for a minimum time T_min, even if SW tries to release it before (for the case of error pin configured in bi-stable mode only). T_min = 250 µs + delta_T, with delta_T parameter being configurable by SW up to 10 ms
- The overall error detection, processing and indication time is less than 10 ms
- The actual maximum time is five safe (IRCOSC) clock cycles

Two classes of fault are identified based on the criticality and the related reactions. Internal (short or long functional reset request pulse, interrupt request) and external (EOUT signaling) reactions are statically defined or programmable based on the fault criticality. The default configuration can be modified only in a specific FCCU state for application/test/debugging purposes. FCCU is designed for assuming that the system clock is faster than the IRC clock.

65.2.1 Standard features

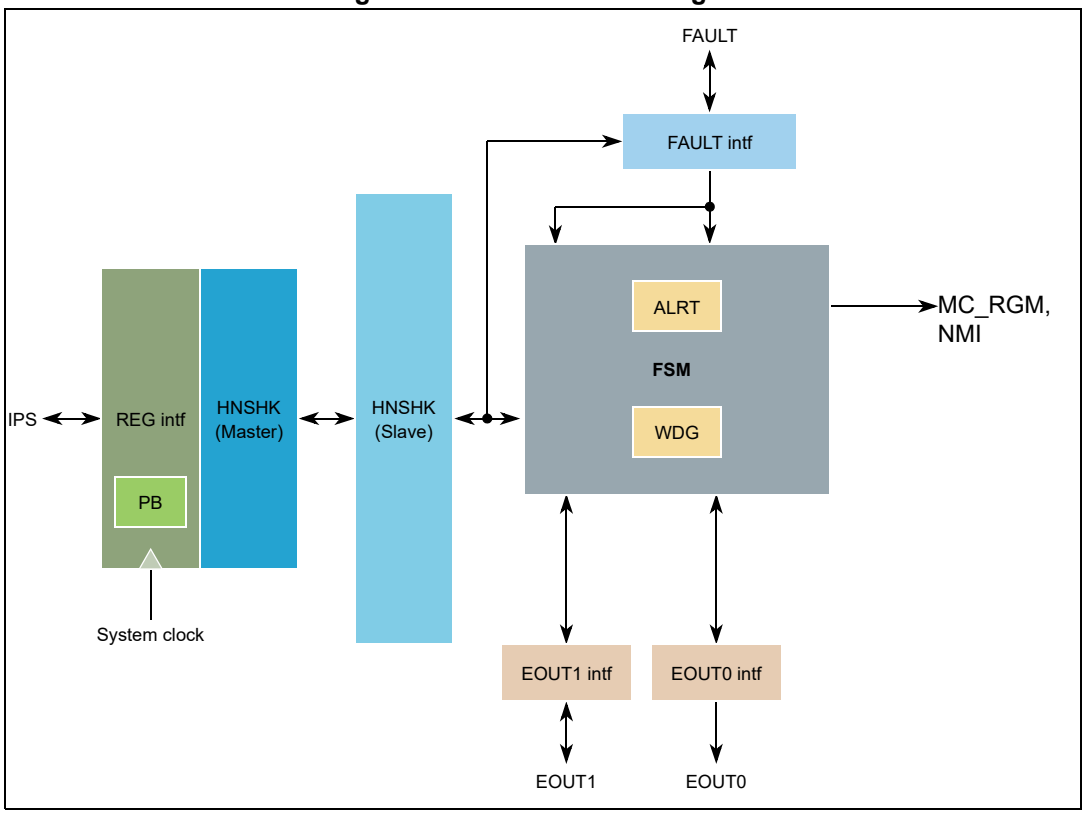
- IPS (slave bus signals) bus interface

65.3 Block diagram

The top-level diagram of the FCCU module is given in *Figure 1313*.

u. Actual value depends on the SoC settings at pad level

Figure 1313. FCCU block diagram



The FCCU module includes the submodules listed in *Table 1260*.

Table 1260. FCCU submodules

Submodule	Description
REG intf	Includes the register file, the IPS bus interface, the IRQ interface and the parity block (PB) for the configuration registers
HNSHK blocks (master and slave blocks)	Includes the FSMs to support the handshake between the REG if and the FSM unit due to the usage of two asynchronous clocks (IPS system clock and RC oscillator clock)
FSM unit	Implements the main functions of the FCCU. The FSM also includes the: Watchdog timer (WDG) Alarm timer (ALRT)
FAULT intf	Implements the interface for the fault conditioning and management
EOUTx units	Implement the output stage to manage the EOUT interfaces

65.4 Signal description

65.4.1 Pinout

The FCCU generates two external signals, EOUT[0] and EOUT[1]. These are described in *Section 65.5.9*.

For the availability of these signals on a chip, see the device configuration chapter.

65.4.2 IPS bus interface

The IPS bus interface is a slave bus used for configuration purposes via CPU. The following bus operations (contiguous byte enables) are supported:

- word (32 bits) data write/read operations to any registers.
- low and high half-words (16 bits, data[31:16] or data[15:0]) data write/read operations to any registers.
- byte (8 bits, data[31:24] or data[23:16] or data[15:8] or data[7:0] data write/read operations to any registers.
- any other operation (free byte enables or other operations) must be avoided.

The FCCU module generates a transfer error in the following cases:

- any write/read access executed outside the register address space of the peripheral. Note that if some registers are not implemented (for example FCCU_CFS_CFGx registers), they generate no bus transfer error when accessed in writing.
- any write/read operation different from byte/hword/word (free byte enables or other operations) on each register.
- any write access executed on configuration registers, when the fccu is not in the CONFIG state. Configuration registers are those registers which can be written only in CONFIG state. Please see *Table 1269*.
- any write executed on FCCU_CFG_TO register, when the fccu is in the CONFIG state.
- any write access to the Transient and permanent Lock Register in any mode other than supervisor access mode.

The registers of the FCCU module are accessible (read/write) in each access mode: user, supervisor or test.

65.5 Functional description

65.5.1 Definitions

In general, the following definitions are applicable for the fault management:

- HW recoverable fault: the fault indication is an edge-triggered and level-sensitive signal that remains asserted until the fault cause is asserted. That is, if 0 on the fault signal indicates fault, then the status flags are valid until the fault line is ‘0’. The status is automatically cleared when the fault signal goes to 1. Typically the fault signal is latched in an external module at the FCCU. The FCCU state transitions are consequently executed on the state changes of the input fault signal. No SW intervention in the FCCU is required to recover the fault condition.
- SW recoverable fault: the fault indication is a signal asserted without a defined time duration. The fault signal is latched in the FCCU. The fault recovery is executed following a SW recovery procedure (status/flag register clearing).

HW recoverable is an option to exclude the handling of error source/s by FCCU management SW, in case it is known that the error is recoverable by itself when the error condition gets corrected.

The following types of reset are applicable:

- Destructive reset: any type of reset related to a power failure condition that implies a complete system re-initialization.
- Long functional reset: implies the Flash and digital circuitry (most of it except FCCU, STCU) initialization.
- Short functional reset: implies the digital circuitry (most of it except FCCU, STCU) initialization.

65.5.2 FSM description

The FCCU module functionality is depicted by the FSM diagram given in *Figure 1314*.

Basically four states are identified with the following meaning:

- CONFIG: The configuration state is used only to modify the default configuration of the FCCU. A subset of the FCCU registers, dedicated to define the FCCU configuration (global configuration, reactions to fault, timeout, recoverable fault masking) can be accessed in write mode only in the CONFIG state.
  The CONFIG state is accessible only in NORMAL state and if the configuration is not locked. The permanent configuration lock can be disabled by a reset of the FCCU, the transient lock register is unlocked by writing 0xBC into it. FCCU gets transiently locked again if an invalid key is written into FCCU_TRANS_LOCK register (a value different to 0xBC). Key to lock the FCCU permanently is 0xFF, written in the FCCU_PERMNT_LOCK register. After reset lifts, the state shall be transiently locked (permanent lock → is unlocked and transient lock → is locked).
  The CONFIG to NORMAL state transition can be executed by SW or automatically following a timeout condition of the watchdog. In case the timeout information and the SW request to mode change to NORMAL appears at the same time, watchdog timeout had the priority and hence the Configuration registers (those that are writable only in CONFIG mode) are reset to their default values. Also, FCCU_CFG_TO and FCCU_EINOUT are reset to their default values, although they do not meet the condition of being a configuration register. The movement to NORMAL is made.
  The incoming faults, occurring during the configuration phase (CONFIG state) are anyway latched in order to process them when the FCCU is moved into the NORMAL state, according to the new configuration.
- NORMAL: This is the FCCU's operating state when no faults are occurring. It is also the default state on the reset exit. Succeeding one of the following events:
  - unmasked recoverable faults with the timeout disabled → the FCCU moves to the FAULT state.
  - unmasked recoverable faults with the timeout enabled → the FCCU moves to the ALARM state.
  - masked recoverable faults → the FCCU stays in NORMAL state.
- ALARM: the FCCU moves into the ALARM state when a unmasked recoverable fault occurs and the timeout is enabled. The transition to the ALARM state goes along with an interrupt request (ALARM state), if enabled. By definition, this fault may be recovered within a programmable timeout period, before it generates a transition to the

FAULT state. The timeout is re-initialized if the FCCU state moves to the NORMAL state. The timeout restarts following the recovery from the FAULT state.

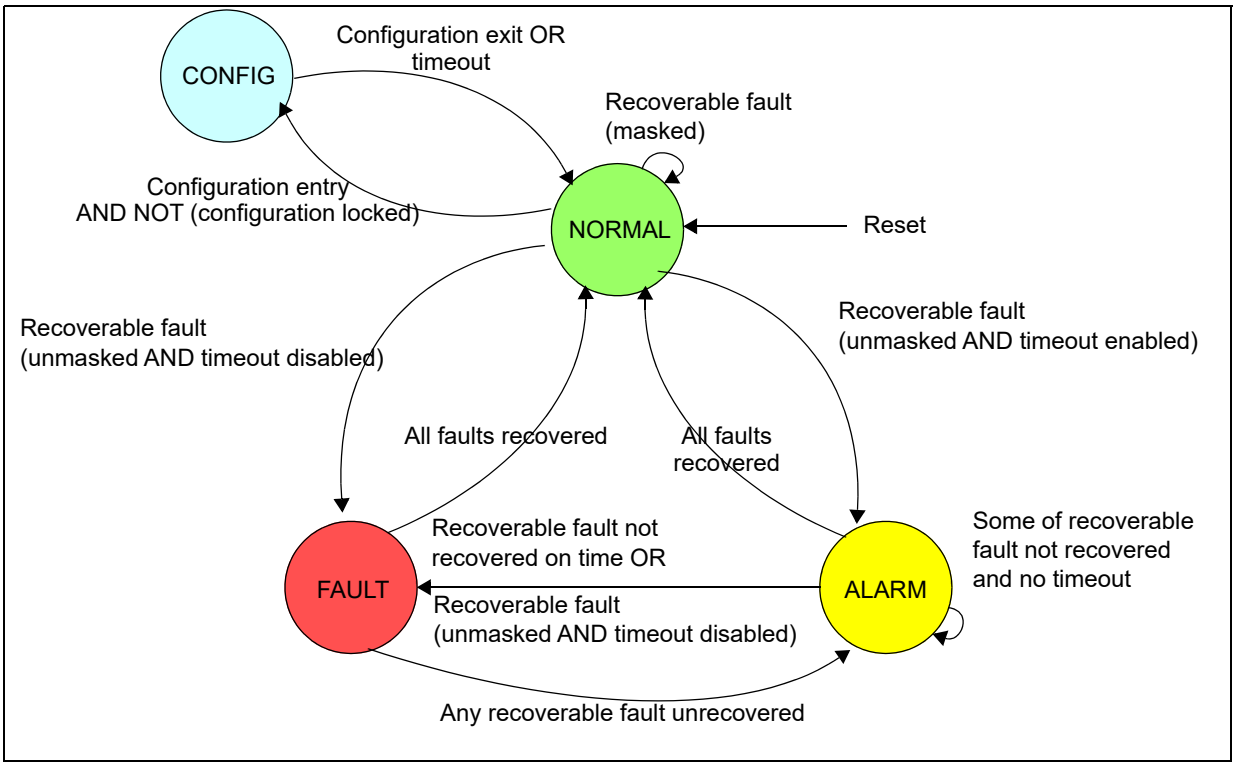
- FAULT: the FCCU moves into the FAULT state when one of the following condition occurs:
  - timeout related to a recoverable fault when the FCCU is in the ALARM state.
  - unmasked recoverable faults with the timeout disabled.

The transition from NORMAL/ALARM state goes along with the generation of:

- NMI interrupt (optional).
- EOUT signaling (optional).
- SW option: Soft reaction (Short functional reset request pulse if configured).
- SW option: Hard reaction (Long functional reset request pulse if configured).

Non Maskable Interrupt (NMI) is routed only to the Safety Core.

Figure 1314. FCCU state diagram



65.5.3 Reset interface

The FCCU has two input resets, as described in *Table 1261* and two output resets related to the FAULT state. In FAULT state, each fault can be programmed to generate a soft or hard reaction request pulse.

In *Table 1262* for each type of fault reaction (hard, soft or self-checking), the respective output reset (Long functional reset, Short functional reset) is activated. In *Table 1263* for each type of reset source (external reset, POR, STCU, FCCU failure, short functional, long functional), the respective FCCU input reset state is described.

The MC unit generates the corresponding input reset as described in *Table 1261*.

Table 1261. Input reset

Input reset	Description
System reset	Global reset of the FCCU synchronous to the system clock.
System reset (RC oscillator)	Global reset of the FCCU synchronous to the IRCOSC.

Table 1262 and *Table 1263* describe a typical reset strategy at system level.

Table 1262. Reset reactions

FAULT reaction	Active reset	System reset signal
Hard	Long functional reset	disabled
Soft	Short functional reset	disabled

Table 1263 shows the Reset sources.

Table 1263. Reset sources

Reset source	System reset signal
External reset	enabled
POR	enabled
STCU	enabled
FCCU failure	application dependent
Functional reset	disabled

Table 1264 summarizes the MC actions corresponding to each reset generated by FCCU.

Table 1264. MC actions

Active reset	MC action
Long functional reset	Reset FLASH and digital modules excluded FCCU, STCU
Short functional reset	Reset digital modules excluded FCCU, STCU

In reset state the FCCU is not able to collect any RF. The FCCU reset phase (POR, external reset, STCU, destructive reset, FCCU failure) must guarantee the assertion of the FCCU input resets at least for two IRCOSC clock periods.

65.5.4 Fault priority scheme and nesting

The FAULT state has a higher priority than the ALARM state in case of concurrent fault events (possible for RFs only) that occur in the NORMAL state.

The ALARM to FAULT state transition occurs if a RF (unmasked and with timeout disabled) is asserted in the ALARM state.

The ALARM to NORMAL state transition occurs only if all the RF (including the faults that have been collected after the entry in the ALARM state) have been cleared (SW or HW recovery) otherwise the FCCU will remain in the ALARM state.

The FAULT to NORMAL state transition occurs only if all the RF (including the faults that have been collected after the entry in the FAULT/ALARM state) have been cleared (SW or HW recovery) otherwise the FCCU will return in the ALARM state (if any RF is still pending and the timeout is not elapsed).

In general, no fault nesting is supported except for the RF that causes a ALARM to FAULT state transition. In this case the alarm timer is stopped until the FAULT state is recovered. If FCCU is in ALARM state and another fault comes, which has its alarm timeout enabled, then the alarm timer shall not reload and shall not start again.

65.5.5 Fault recovery

The following timing diagrams describe the main use cases of the FCCU in terms of fault events and related recovery.

A typical sequence related to a RF management (ALARM state), see *Figure 1315* and *Figure 1316*, is as follows:

- RF assertion
- FCCU state transition (automatic): NORMAL → ALARM
  - alarm interrupt request (if configured)
  - timeout running
- system state: RUN
- alarm interrupt management: FAULT recovery (by SW): FCCU state transition ALARM → NORMAL

Figure 1315. RF (ALARM state) recovery (a)

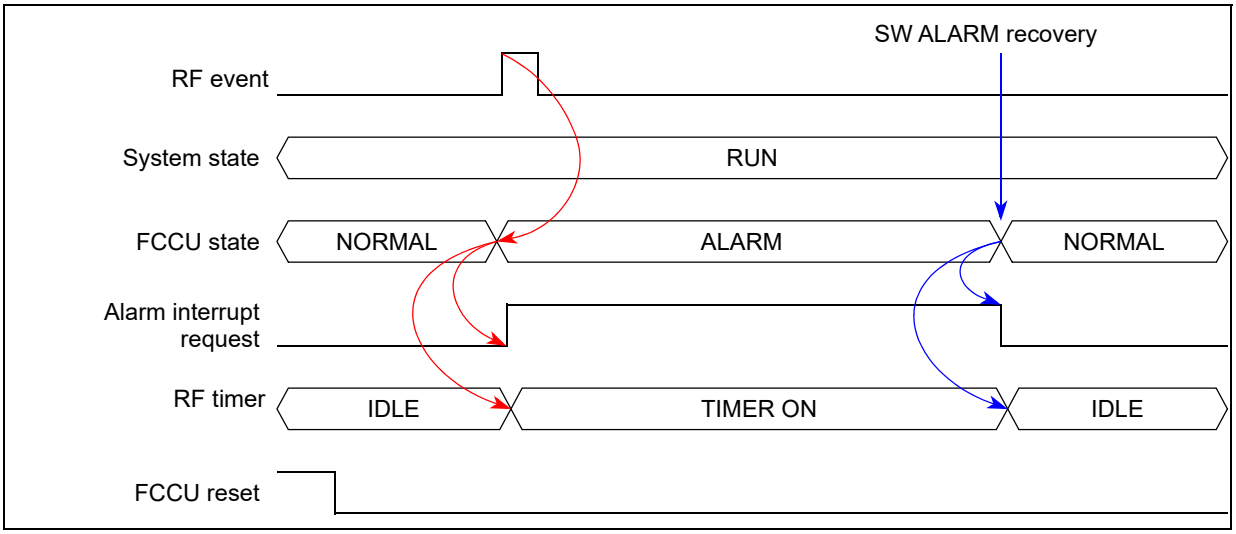
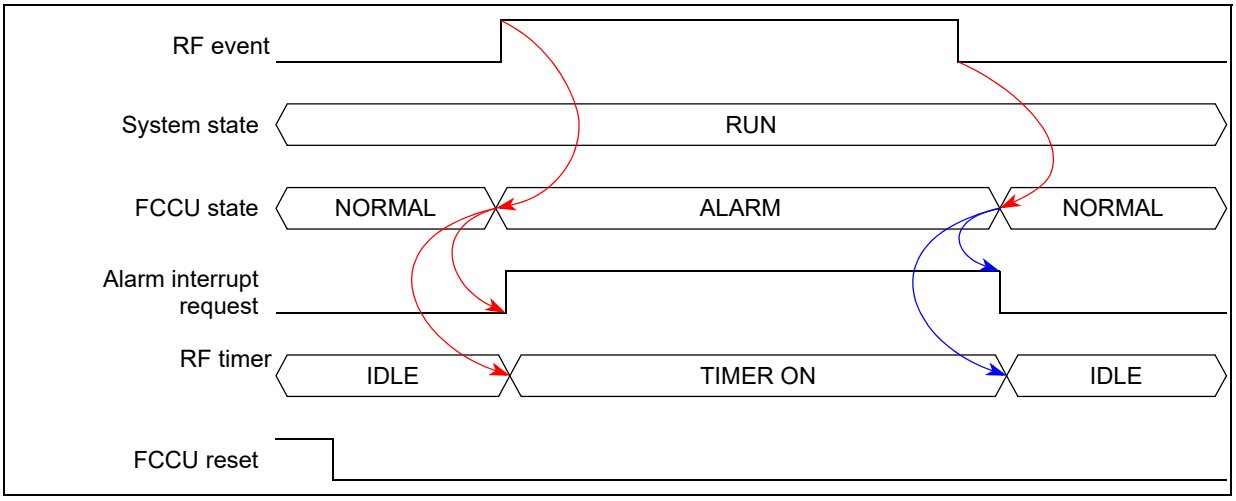


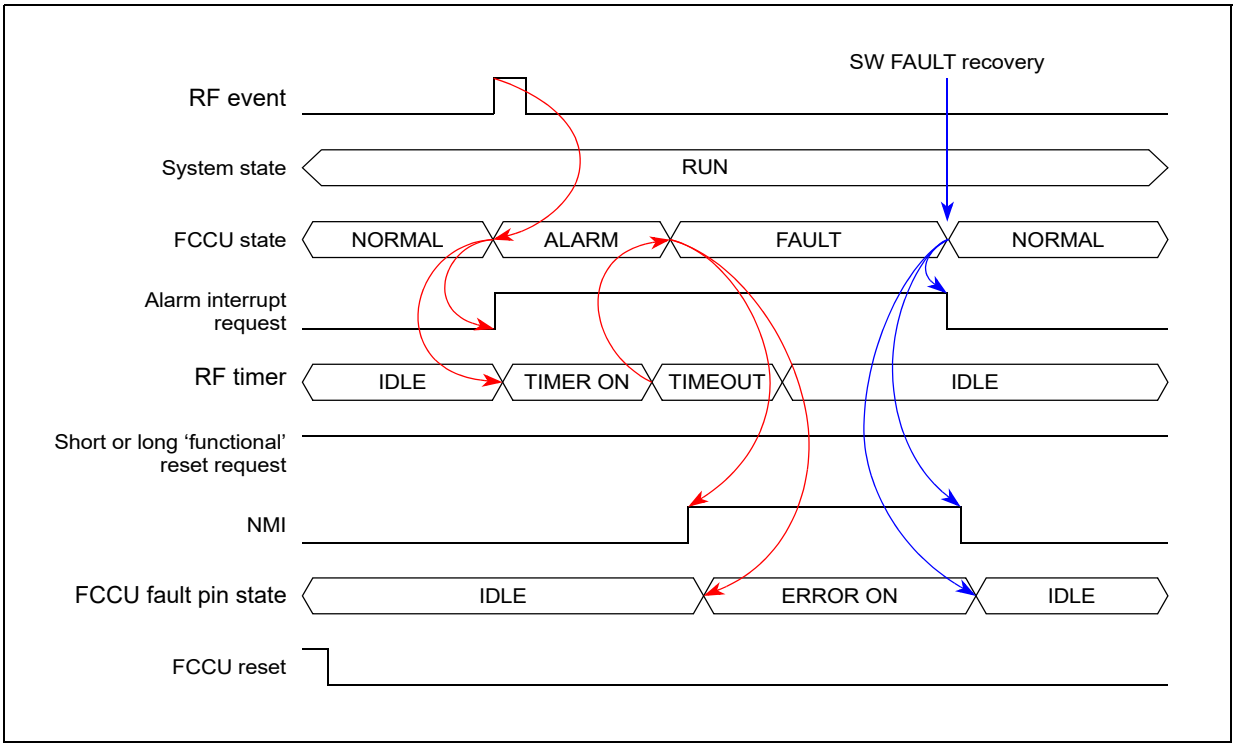
Figure 1316. RF (ALARM state) recovery (b)



A typical sequence related to a RF management (ALARM → FAULT state), see *Figure 1317*, is as follows:

- RF assertion
- FCCU state transition (automatic): NORMAL → ALARM
  - alarm interrupt request (if enabled)
  - timeout running
- FCCU state transition (following the timeout trigger): ALARM → FAULT: NMI assertion (if enabled)
- NMI interrupt management (if enabled)
  - FAULT recovery (by SW): FCCU state transition FAULT → NORMAL

Figure 1317. RF(ALARM -> FAULT state) recovery



65.5.6 NMI/WKPU interface

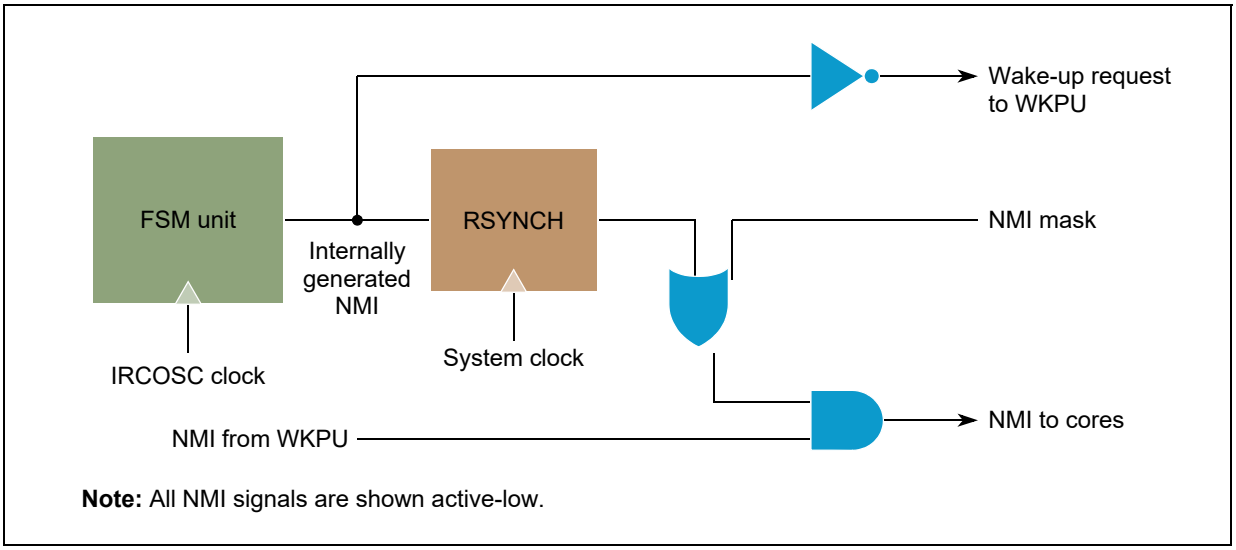
The wake-up interface includes Nonmaskable interrupt: NMI interrupt source generated by the WKPU. It is synchronous with the system clock and active low. A logical AND interconnects the NMI sources (internally generated by FCCU and by WKPU) in order to provide a single NMI output.

The NMI signal internally generated by FCCU is masked when:

- FCCU asynchronous reset
- MC's current mode (MC state[3:0]) = RESET

and unmasked when a SW change request of the MC state is triggered (MC mode SW change request = 1).

Figure 1318. NMI/WKUP scheme



Note: All NMI signals are shown active-low.

65.5.7 FAULT interface

The basic functionality of the fault interface is to provide a simple, programmable edge-sensitive interface (active high or low) to signal a fault condition to the FCCU module.

To support a fault injection mechanism an additional and optional signal is available. It is an active high signal and is asserted for a single clock cycle synchronously at the system clock. The behavior is given in *Figure 1319*. The fault injection is executed by a write operation in the FCCU_RFF register.The reaction following a fake RF cannot be masked. The fault injection mechanism is optional; see the implementation-specific details. Refer to “FCCU failure inputs” table in Functional Safety chapter, for more details.

To support a fault clearing mechanism, in order to clear a fault latched directly in the FAULT root, an additional and optional signal (FCCU recoverable faults clear) is available. This active-high signal is asserted for a single clock cycle synchronously at the system clock. The behavior is given in *Figure 1320*. The fault clearing is executed when the SW “clears” the status (flag) bits of the FCCU_RF_S*n*. The fault clearing mechanism is optional and its support must be specified at system level.

Figure 1319. FAULT injection

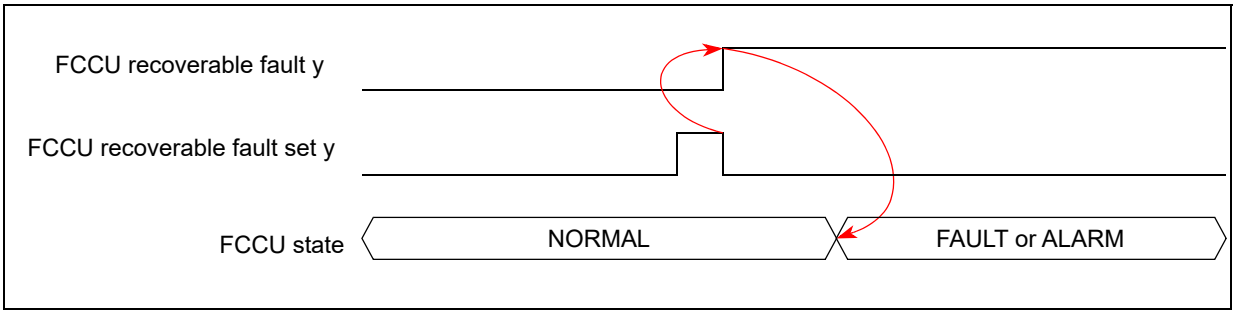
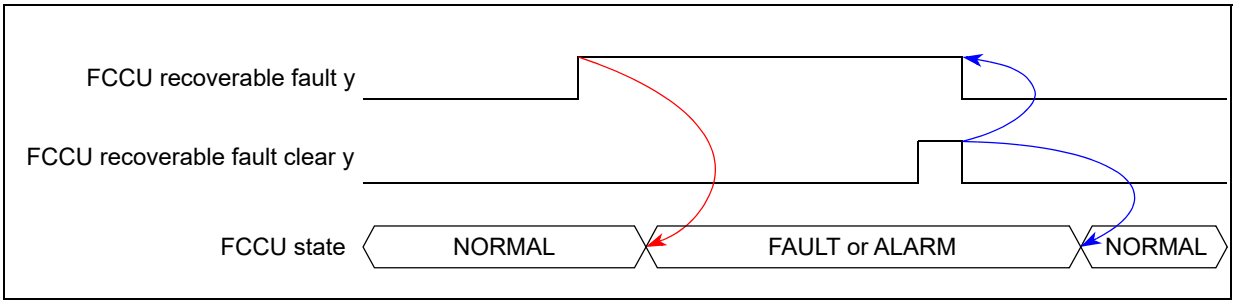


Figure 1320. FAULT clearing



65.5.8 STCU interface

The STCU interface includes:

- a set of signals resulting from the self-checking procedure connected externally at these FCCU faults:
  - Recoverable
- the STCU fault signals are processed by the FCCU when the SOC is rebooted following the self-testing procedure. The STCU includes also a status register that stores the self-testing results (flags).
- a mask (EOUT mask[1:0]) that inhibits the EOUT dummy signaling until the STCU self-checking procedure has been completed.

During the self testing procedure, depending on the STCU results, three cases are applicable:

a) STCU completes the self testing procedure successfully. The SOC reboots and the FCCU is responsible to provide a reaction.

b) STCU completes the self testing procedure with low severity failures. The FCCU is responsible to provide the proper reactions according to the fault occurred.

c) STCU completes the self testing procedure with serious failures. The FCCU or other critical parts of the SoC could not be able to provide the proper reaction. STCU should keep permanently the SoC in RESET state. This feature is optionally programmable inside the STCU.

Figure 1321. STCU-FCCU (case a)

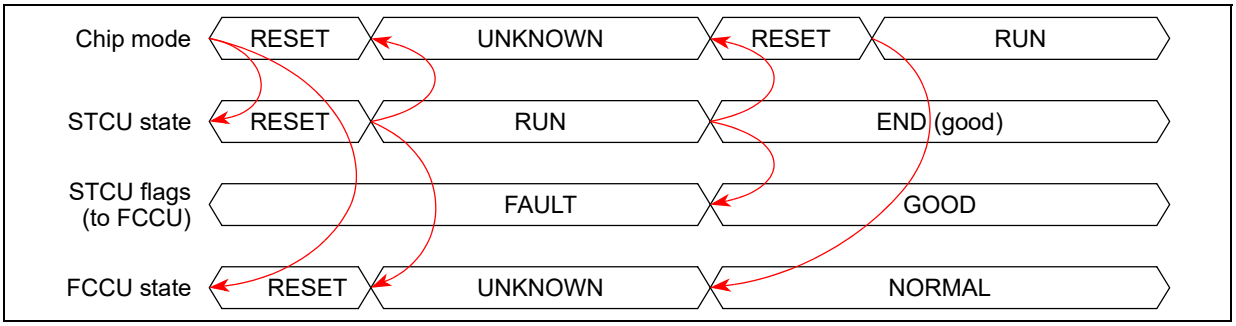


Figure 1322. STCU-FCCU (case b)

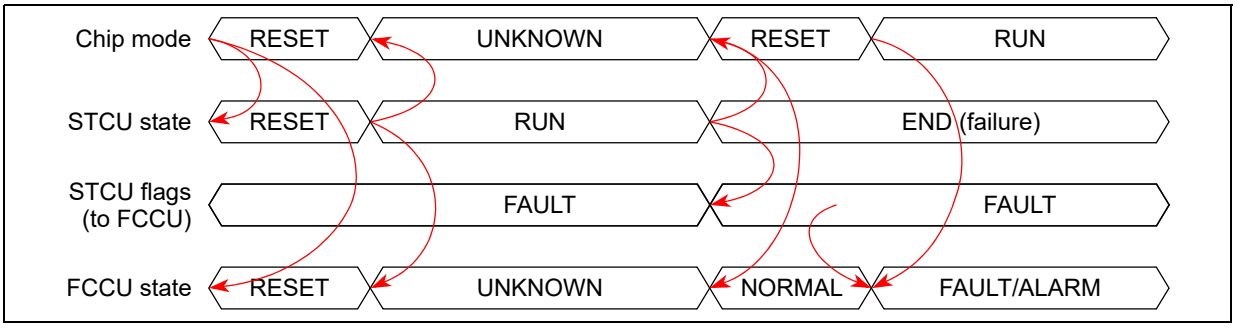
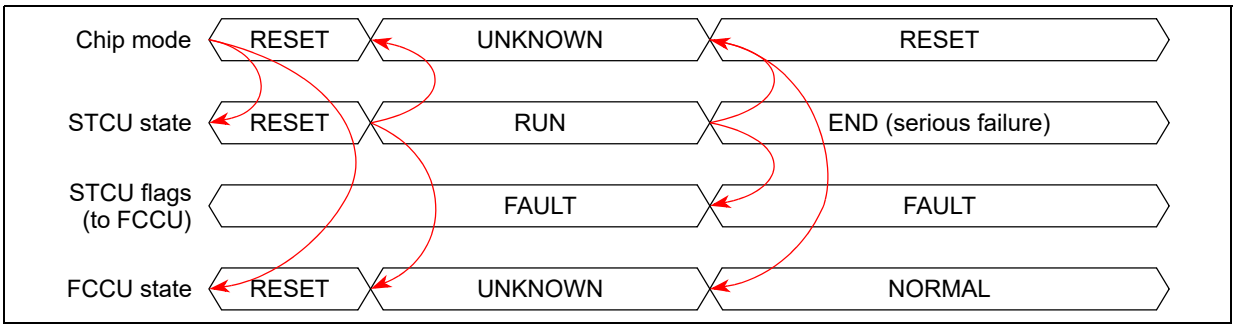


Figure 1323. STCU-FCCU (case c)



65.5.9 EOUT interface

The FCCU provides up to two bidirectional signals (EOUT interface) as a failure indication to the external world (see table “FCCU failure inputs” in the “Functional safety” chapter for details). These signals need to be mapped to appropriate pins for external visibility. Since SIUL2 does not control the pad buffer enable during Alternate function, FCCU provides a mechanism to drive the pad in OD (Open Drain) mode where the pad is driven Hi-Z when FCCU drives logic ‘1’ else it drives logic ‘0’ (when FCCU drives logic '0'). The selection to OD is done by OD bit in FCCU_CFG register. Different protocols for the EOUT interface are supported, selecting the FCCU_CFG.FOM register field:

- Dual rail protocol
- Time switching protocol
- Bi-stable protocol
- Test mode

The signal polarity can be programmed by setting the PS and FOP field in the FCCU_CFG register. For example, for PS = 0, FCCU Error output[0] = 0 and FCCU Error output[1] = 1 in FAULT state for bi-stable and for PS = 1, it is inverted. All the diagrams and tables are related to the default configuration selection (FCCU_CFG.FOP = 0b): switching mode (FCCU_CFG.SM = 0b). In case of inverted polarity (FCCU_CFG.PS = 1b) all the values on the EOUT output pins are inverted.

Two modes can be programmed to define the EOUT protocol transitions in dual rail or time switching mode:

- slow switching mode: no EOUT frequency violation during the FCCU state transition (NORMAL to FAULT or viceversa and CONFIG to NORMAL). The EOUT protocol

transition occurs after a max delay equal to the duration of the semi-period of the EOUT frequency.

- fast switching mode: The EOUT protocol transition (NORMAL to FAULT or viceversa and CONFIG to NORMAL) occurs immediately. A pulse with the minimum duration corresponding to 16 MHz/1024 (IRCOSC clock) period can occurs in fast switching mode. It implies a frequency violation of the EOUT protocol.

The EOUT frequency is generated by dividing IRCOSC clock by ((FOP + 1) * 2 * 2048) where FOP vary between 0 and 63. With an IRC clock of 16 MHz, this drives a signal between 61 and 3906 Hz on EOUT.

The external monitor of the EOUT protocol should oversample the EOUT signals in order to synchronize periodically the external clock (used by the monitor) and the IRCOSC clock detecting the edge transition of the EOUT protocol in dual rail or time switching mode.

In case of a failure event or on software request for Error pin indication, the pin(s) are set to faulty state for a minimum time T_min, even if SW tries to release it before. If SW configures the error pins to OK(1) and if a fault comes trying to drive the pin to NOK(0), then priority is given to the fault indication and error pins indicate NOK, that is an incoming fault is not masked when SW has set the error pin to high. During the T_min by a non-SW fault, the FCCU FSM moves independently of this pin state (low) and as soon as the timer expires, the pin behavior is dictated by the state in which the FSM finds itself in and it is not possible to set the pins to OK by SW moving FCCU to CONFIG state, as long as this timer is running. No SW intervention is needed to bring the pin from the low state. The SW can bring the pin back to OK state by clearing the faults and waiting for the T_min to expire, after which the FCCU automatically enters NORMAL state and the error pin indicates OK.

In case another failure event happens within T_min after a first one, the T_min counter is restarted.

Resets (except the power on reset, and software bit FCCU_RST in RGM_PRST5) should not influence the state of the failure indication, pins.

Note: The transition from FAULT to CONFIG state is not possible but is shown to display the behavior in all four states: RESET, NORMAL, FAULT and CONFIG.

Table 1265. FCCU error pin behavior on STATE transitions(1)

Fault signaling state	Transition from NORMAL to FAULT state			NORMAL+CONFIG states			Transition from NORMAL to ALARM state		
	with Fault disabled | enabled						with Fault disabled | enabled		
	Internal Error pin	Error pin enable	Error pad	Internal error pin	Error pin enable	Error pad	Internal error pin	Error pin enable	Error pad
Disabled	NA | OK	NA | 1	NA | OK	OK	1	OK	NA | OK	NA | 1	NA | OK
Enabled	NA | NOK	NA | 1	NA | NOK	OK	1	OK	NA | OK	NA | 1	NA | OK

1. Table valid only after SW has written FCCU_SET_AFTER_RESET bit to 1.

Note: FCCU remains in NORMAL state when fault is disabled hence, “NA” (Not Applicable) appears in the above table.

65.5.9.1 Dual rail protocol

Dual rail encoding is an alternate method for encoding bits. In contrast with classical encoding, where each signal carries a single-bit value, dual rail encoded circuits use two wires to carry each bit. The encoding scheme is given in *Table 1266* and the related timing diagram is given in *Figure 1324* and *Figure 1325*.

Table 1266. Dual rail encoding

Logical state	Dual rail encoding (FCCU Error output pins [1:0])	Note
nonfaulty	10	toggling
nonfaulty	01	
faulty	00	toggling
faulty	11	
reset	high-Z(1)	no toggling
configuration	same as NORMAL(2)	toggling

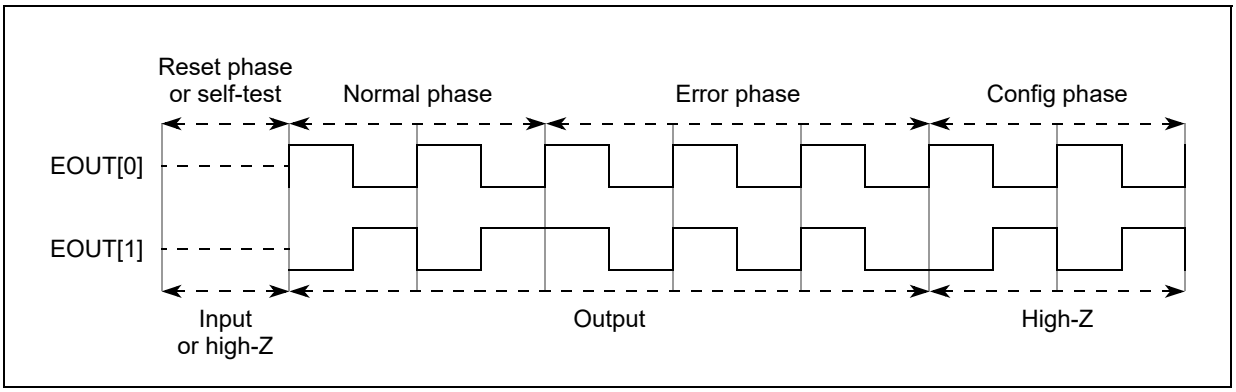
1. Final value depends on the SoC settings at pad level.
2. Till FCCU_SET_AFTER_RESET is written, the Error indicating pads are driven to High impedance from FCCU's side. That bit is write-able only after the FCCU enters CONFIG state. Hence on entry to CONFIG state, user may still see high impedance.

As long as FCCU is in NORMAL or ALARM state, output will show “nonfaulty” signal. Output pins #0, #1 will toggle between 01 and 10.

In the RESET phase the output pins are set as “high impedance”(v).

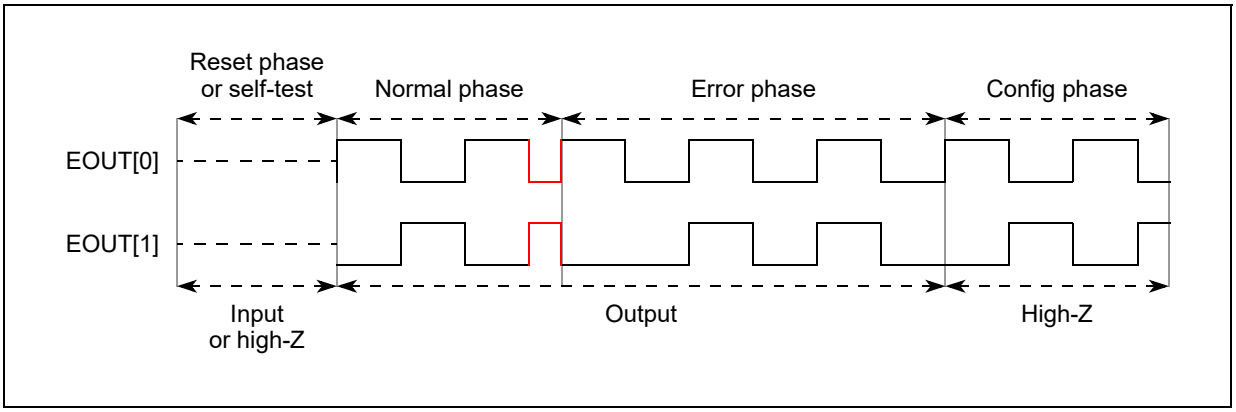
Note: Figure 1324 and Figure 1325 are formatted to display the behavior in all four phases (reset, normal, error, and config), not to imply transitions between one phase to another. In particular, transition from error phase to config phase is not possible.

Figure 1324. Dual rail protocol (slow switching mode)



v. Actual value depends on the SOC settings at pad level

Figure 1325. Dual rail protocol (fast switching mode)



65.5.9.2 Time switching protocol

The encoding scheme is given in *Table 1267* and the related timing diagram is given in *Figure 1326*.

Table 1267. Time switching encoding

Logical state	Time switching encoding (FCCU Error output pins [1:0])	Note
nonfaulty	10	toggling
nonfaulty	01	
faulty	10	no toggling
reset	high-Z(1)	no toggling
configuration	same as NORMAL(2)	toggling

1. Final value depends on the SoC settings at pad level.
2. Till FCCU_SET_AFTER_RESET is written, the Error indicating pads are driven to High impedance from FCCU's side. That bit is write-able only after the FCCU enters CONFIG state. Hence on entry to CONFIG state, user may still see high impedance.

As long as FCCU is in NORMAL or ALARM state, outputs will show “nonfaulty” signal. Output pins #0, #1 will toggle between 01 and 10.

In the FAULT state, the output pin FCCU Error output pin[0] is set as low.

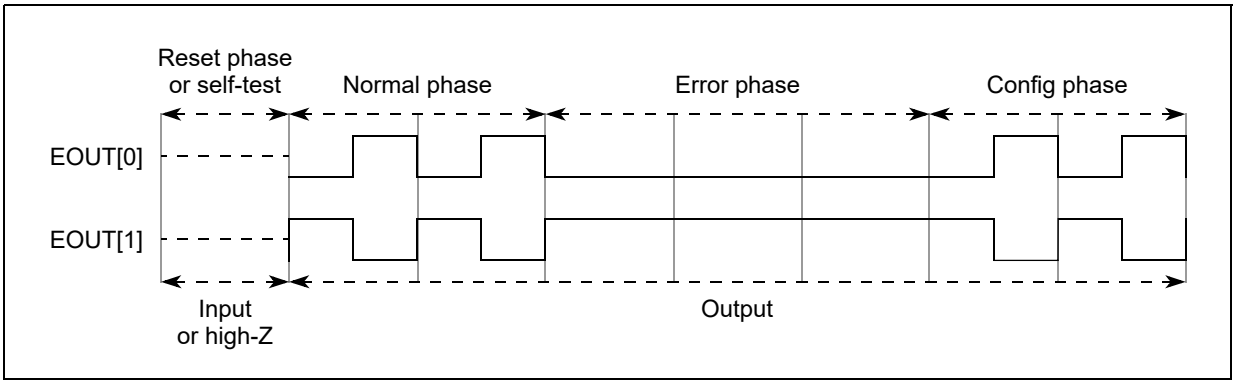
In Time Switching mode the second output (FCCU Error output pin[1]) is the inverted signal of first output (FCCU Error output pin[0]).

In the RESET phase the output pins are set as “high impedance”[w].

Note: Figure 1326 is formatted to display the behavior in all four phases (reset, normal, error, and config), not to imply transitions between one phase to another. In particular, transition from error phase to config phase is not possible.

w. Actual value depends on the SOC settings at pad level

Figure 1326. Time switching protocol



65.5.9.3 Bi-stable protocol

The encoding scheme is given in *Table 1268* and the related timing diagram is given in *Figure 1327*.

Table 1268. Bi-stable encoding

Logical state	Bi-stable encoding (FCCU Error output pins [1:0])	Note
nonfaulty	01	no toggling
faulty	10	no toggling
reset	high-Z(1)	no toggling
configuration	same as NORMAL(2)	no toggling

1. Final value depends on the SoC settings at pad level.
2. Till FCCU_SET_AFTER_RESET is written, the Error indicating pads are driven to High impedance from FCCU's side. That bit is write-able only after the FCCU enters CONFIG state. Hence on entry to CONFIG state, user may still see high impedance.

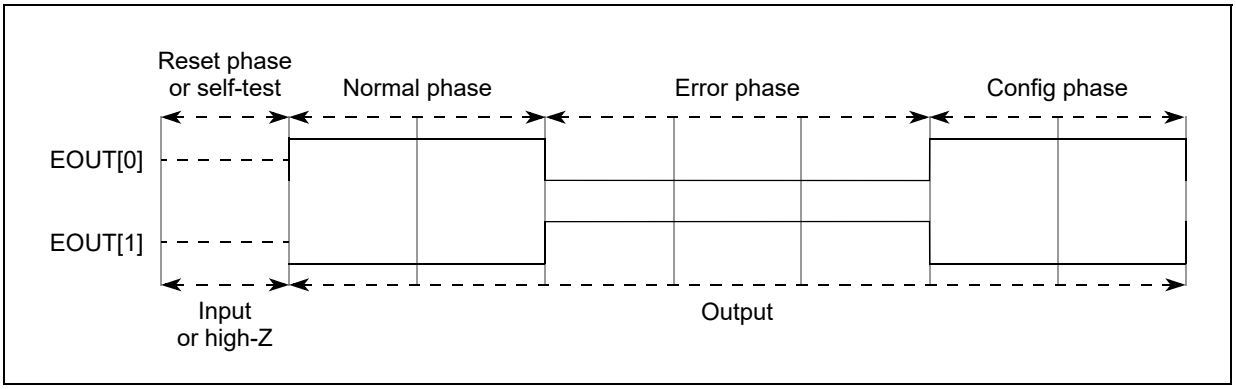
In the FAULT state, the faulty logical state is indicated. In NORMAL or ALARM state, “no-faulty” state is indicated. In Bi-stable mode the second output (FCCU Error output[1]) is the inverted signal of first output (FCCU Error output[0]).

In the RESET phase the output pins are set as “high impedance”(x).

Note: Figure 1327 is formatted to display the behavior in all four phases (reset, normal, error, and config), not to imply transitions between one phase to another. In particular, transition from error phase to config phase is not possible.

x. Actual value depends on the SOC settings at pad level

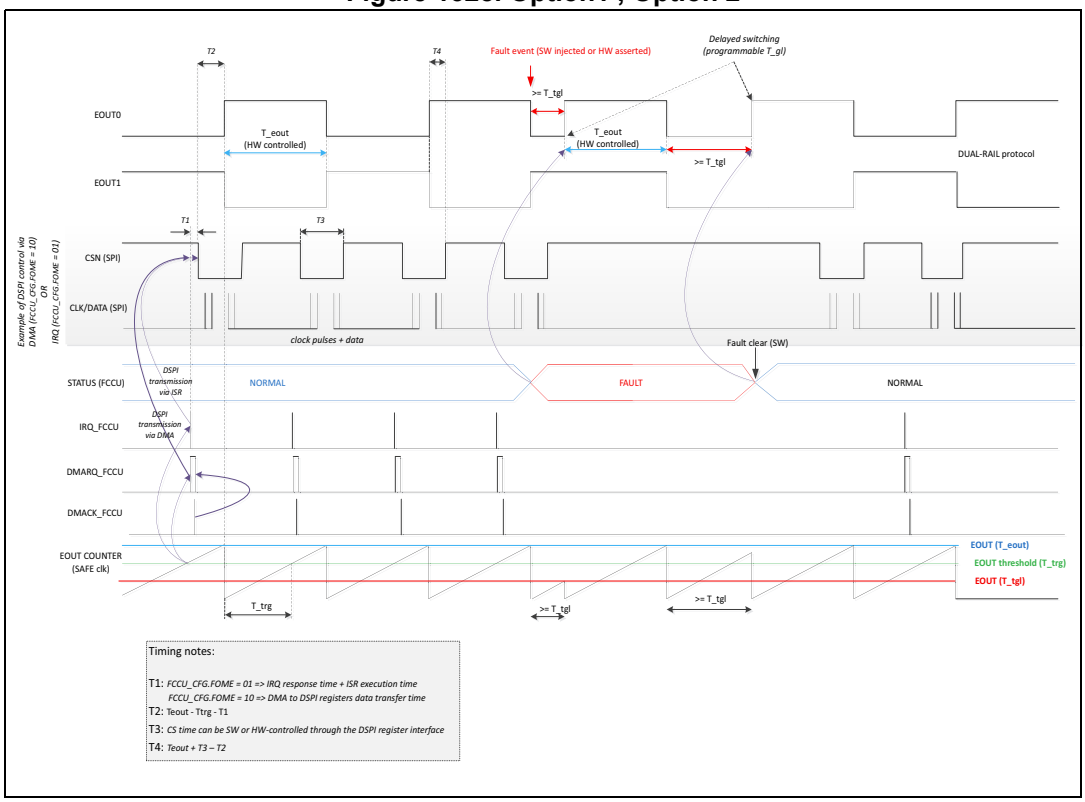
Figure 1327. Bi-stable protocol



65.5.9.4 Option 1, Option 2

The internal counter used to generate the EOUT waveforms is used to control a IRQ or DMA trigger event. This event shall appear only when the FCCU is not in FAULT mode (see *Figure 1328*).

Figure 1328. Option1 , Option 2

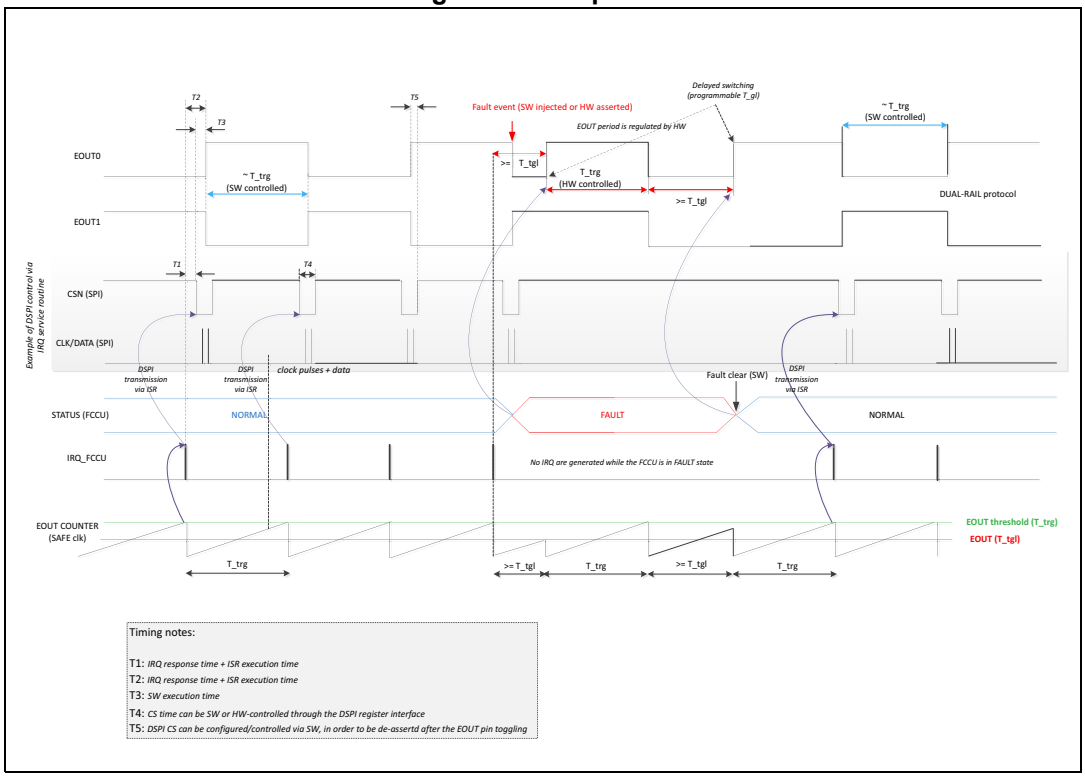


65.5.9.5 Option 3

When FCCU_CFG. FOME = 11, the EOUT pin state is controlled by the application SW as far as the FCCU is in NORMAL or ALARM state. As soon as a the FCCU enters the FAULT state, both the EOUT pins are forced by the HW to the same level, which can be 0-0 or 1-1 depending on the polarity configuration. As far as the FCCU is in FAULT state, the SW has

no control over the EOUT pins. Once the error condition is recovered, the FCCU returns to the NORMAL state and the SW can take back the control of the EOUT pins. *Figure 1329* shows how this is accomplished. As far as the FCCU is not in FAULT mode, the IRQ interrupts are asserted when the FCCU internal timer reaches the programmed threshold. If needed, it is responsibility of the SW to toggle the EOUT pins within the ISR. The EOUT pin toggling period depends on the IRQ assertion frequency (controlled by the HW) plus the interrupt service time and SW execution. This period will be stable and close to the IRQ programmed period as far as the interrupt service routine can be serviced in a deterministic time frame and executed at the highest priority. During the ISR, additional operations can be done: the figure shows an example of a DSPI frame sent before every EOUT toggling, but with the DSPI CS being kept asserted for some time after the EOUT has toggled. When the FCCU enters into the FAULT state, no IRQ interrupt is fired anymore and the EOUT pins are controlled by the HW. During this time frame, the EOUT pin toggling is exactly matching the programmed period.

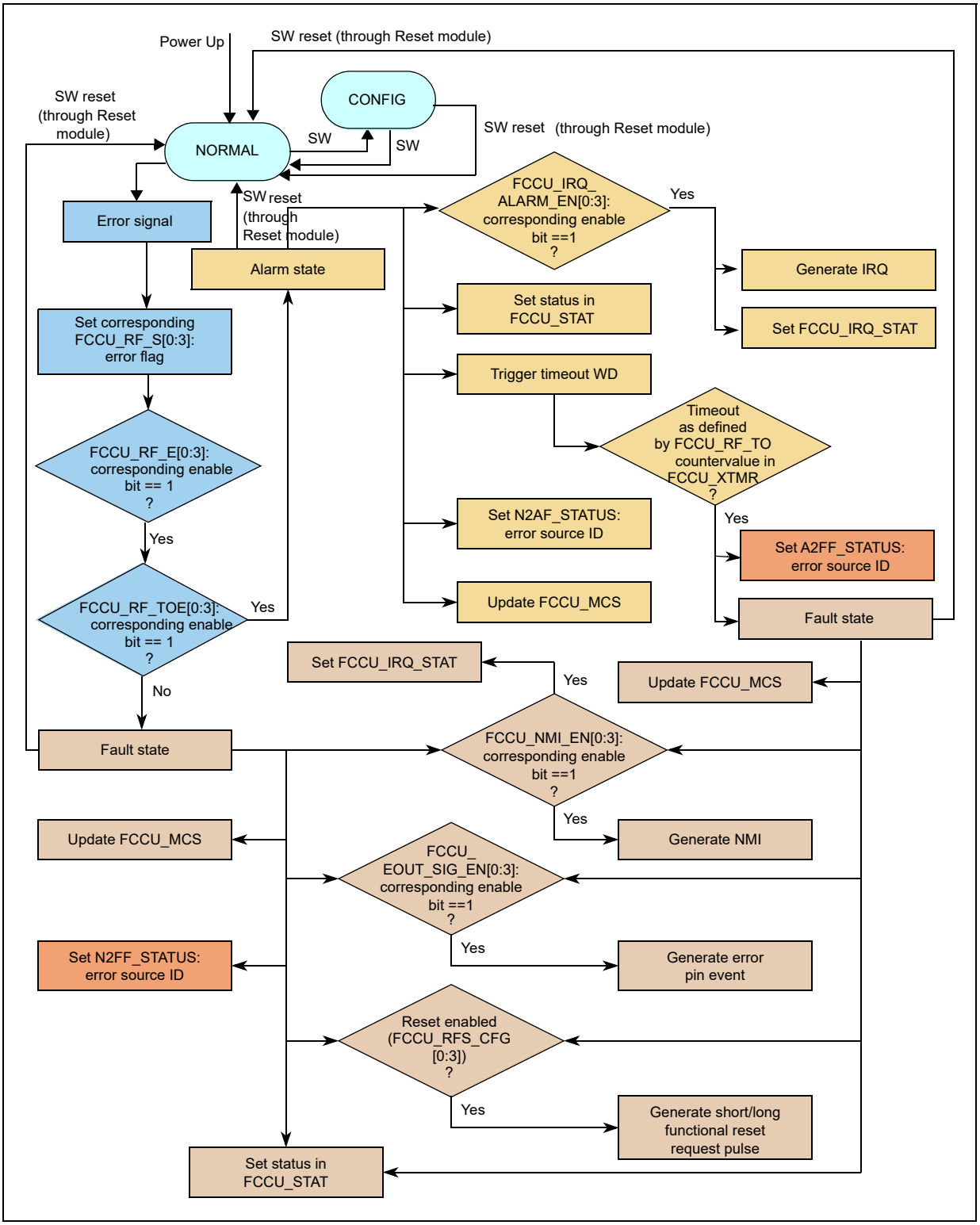
Figure 1329. Option 3



65.5.10 Error signal flow for RF case

On Power up, FCCU moves to NORMAL state. SW can configure it by moving it to CONFIG and back again. SW reset leads to movement back to NORMAL. On reaching ALARM state, depending on ALARM being enabled, FCCU generates IRQ(alarm) and the FCCU_IRQ_STAT's alarm status is also set. FSM status is set in FCCU_STAT along with N2AF_STATUS. If Error is enabled (FCCU_RF_E) and timeout disabled (FCCU_RF_TOE), FCCU moves straight to FAULT state and depending on FCCU_NMI_EN or FCCU_SIG_EN, or both, being set, it generates a NMI or signals error, or both, out to the external world. FCCU FSM status is also captured in FCCU_STAT.

Figure 1330. Error signal flow



65.6 Register description

The FCCU registers are listed in *Table 1269*. Any address offset not explicitly mentioned in *Table 1269* is reserved.

All the registers accessible in write mode only in CONFIG state (last column of the *Table 1269*) are referred as configuration registers. These configuration registers return to the default value after configuration watchdog timer expires. These are the registers protected by FCCU_TRANS_LOCK and FCCU_PERMNT_LOCK registers. The configuration register setting has effect only when the FCCU state exits from the CONFIG state.

For each possible RF failure source a different reaction shall be configurable through the use of NMI, IRQ, long/short reset selection registers as well as no reaction by disabling the former registers. It is not possible for a single event upset to switch off all reactions on failures as implementation is per fault source (but it will be possible to switch them all off by SW if intended). Failures themselves are not able to disable all reactions and indications.

The FCCU is not reset by short or long functional resets.

Table 1269. FCCU memory map

Address offset	Register	Access(1)	Register classification(2)	Location
0x000	FCCU Control Register (FCCU_CTRL)	R/W always	Saf-Lat	*Section 65.6.1*
0x004	FCCU CTRL Key Register (FCCU_CTRLK)	W always	Saf-Relv	*Section 65.6.2*
0x008	FCCU Configuration Register (FCCU_CFG)	R always; W in CONFIG state only	Saf-Lat	*Section 65.6.3*
0x01C	FCCU RF Configuration Register 0 (FCCU_RF_CFG0)	R always; W in CONFIG state only	Saf-Lat	*Section 65.6.4*
0x020	FCCU RF Configuration Register 1 (FCCU_RF_CFG1)		Saf-Lat	
0x024	FCCU RF Configuration Register 2 (FCCU_RF_CFG2)		Saf-Lat	
0x028	FCCU RF Configuration Register 3 (FCCU_RF_CFG3)		Saf-Lat	

Table 1269. FCCU memory map (continued)

Address offset	Register	Access(1)	Register classification(2)	Location
0x04C	FCCU RFS Configuration Register 0 (FCCU_RFS_CFG0)	R always; W in CONFIG state only	Saf-Lat	Section 65.6.5
0x050	FCCU RFS Configuration Register 1 (FCCU_RFS_CFG1)		Saf-Lat	
0x054	FCCU RFS Configuration Register 2 (FCCU_RFS_CFG2)		Saf-Lat	
0x058	FCCU RFS Configuration Register 3 (FCCU_RFS_CFG3)		Saf-Lat	
0x05C	FCCU RFS Configuration Register 4 (FCCU_RFS_CFG4)		Saf-Lat	
0x060	FCCU RFS Configuration Register 5 (FCCU_RFS_CFG5)		Saf-Lat	
0x064	FCCU RFS Configuration Register 6 (FCCU_RFS_CFG6)		Saf-Lat	
0x068	FCCU RFS Configuration Register 7 (FCCU_RFS_CFG7)		Saf-Lat	
0x080	FCCU RF Status Register 0 (FCCU_RF_S0)	R/W always	Saf-Lat	Section 65.6.6
0x084	FCCU RF Status Register 1 (FCCU_RF_S1)		Saf-Lat	
0x088	FCCU RF Status Register 2 (FCCU_RF_S2)		Saf-Lat	
0x08C	FCCU RF Status Register 3 (FCCU_RF_S3)		Saf-Lat	
0x090	FCCU RF Key Register (FCCU_RFK)	W always	Saf-Relv	Section 65.6.7
0x094	FCCU RF Enable Register 0 (FCCU_RF_E0)	R always; W in CONFIG state only	Saf-Lat	Section 65.6.8
0x098	FCCU RF Enable Register 1 (FCCU_RF_E1)		Saf-Lat	
0x09C	FCCU RF Enable Register 2 (FCCU_RF_E2)		Saf-Lat	
0x0A0	FCCU RF Enable Register 3 (FCCU_RF_E3)		Saf-Lat	
0x0A4	FCCU RF Timeout Enable Register 0 (FCCU_RF_TOE0)	R always; W in CONFIG state only	Saf-Lat	Section 65.6.9
0x0A8	FCCU RF Timeout Enable Register 1 (FCCU_RF_TOE1)		Saf-Lat	
0x0AC	FCCU RF Timeout Enable Register 2 (FCCU_RF_TOE2)		Saf-Lat	
0x0B0	FCCU RF Timeout Enable Register 3 (FCCU_RF_TOE3)		Saf-Lat	
0x0B4	FCCU RF Timeout Register (FCCU_RF_TO)	R always; W in CONFIG state only	Saf-Lat	Section 65.6.10

Table 1269. FCCU memory map (continued)

Address offset	Register	Access(1)	Register classification(2)	Location
0x0B8	FCCU CFG Timeout Register (FCCU_CFG_TO)	R always; W in all states except CONFIG	Saf-Lat	*Section 65.6.11*
0x0BC	FCCU IO Control Register (FCCU_EINOUT)	R/W always	Saf-Lat	*Section 65.6.12*
0x0C0	FCCU Status Register (FCCU_STAT)	R only	Saf-Lat	*Section 65.6.13*
0x0C4	FCCU NA Freeze Status Register (N2AF_STATUS)	R/W always	No-Saf	*Section 65.6.14*
0x0C8	FCCU AF Freeze Status Register (A2FF_STATUS)	R/W always	No-Saf	*Section 65.6.15*
0x0CC	FCCU NF Freeze Status Register (N2FF_STATUS)	R/W always	No-Saf	*Section 65.6.16*
0x0D0	FCCU FA Freeze Status Register (F2A_STATUS)	R/W always	No-Saf	*Section 65.6.17*
0x0DC	FCCU RF Fake Register (FCCU_RFF)	R/W always	No-Saf	*Section 65.6.18*
0x0E0	FCCU IRQ Status Register (FCCU_IRQ_STAT)	R/W always	Saf-Relv	*Section 65.6.19*
0x0E4	FCCU IRQ Enable Register (FCCU_IRQ_EN)	R/W always	Saf-Lat	*Section 65.6.20*
0x0E8	FCCU XTMR Register (FCCU_XTMR)	R/W always	No-Saf	*Section 65.6.21*
0x0EC	FCCU MCS Register (FCCU_MCS)	R only	Saf-Lat	*Section 65.6.22*
0x0F0	FCCU Transient Lock Register (FCCU_TRANS_LOCK)	W always	Saf-Lat	*Section 65.6.23*
0x0F4	FCCU Permanent Lock Register (FCCU_PERMNT_LOCK)	W always	Saf-Lat	*Section 65.6.24*
0x0F8	FCCU Delta T Register(FCCU_DELTA_T)	R always; W in CONFIG state only	Saf-Lat	*Section 65.6.25*
0x0FC	FCCU IRQ Alarm Enable Register 0 (FCCU_IRQ_ALARM_EN0)	R always; W in CONFIG state only	Saf-Lat	*Section 65.6.26*
0x100	FCCU IRQ Alarm Enable Register 1 (FCCU_IRQ_ALARM_EN1)		Saf-Lat	
0x104	FCCU IRQ Alarm Enable Register 2 (FCCU_IRQ_ALARM_EN2)		Saf-Lat	
0x108	FCCU IRQ Alarm Enable Register 3 (FCCU_IRQ_ALARM_EN3)		Saf-Lat	

Table 1269. FCCU memory map (continued)

Address offset	Register	Access(1)	Register classification(2)	Location
0x10C	FCCU NMI Enable Register 0 (FCCU_NMI_EN0)	R always; W in CONFIG state only	Saf-Lat	Section 65.6.27
0x110	FCCU NMI Enable Register 1 (FCCU_NMI_EN1)		Saf-Lat	
0x114	FCCU NMI Enable Register 2 (FCCU_NMI_EN2)		Saf-Lat	
0x118	FCCU NMI Enable Register 3 (FCCU_NMI_EN3)		Saf-Lat	
0x11C	FCCU EOUT Signaling Enable Register 0 (FCCU_EOUT_SIG_EN0)	R always; W in CONFIG state only	Saf-Lat	Section 65.6.28
0x120	FCCU EOUT Signaling Enable Register 1 (FCCU_EOUT_SIG_EN1)		Saf-Lat	
0x124	FCCU EOUT Signaling Enable Register 2 (FCCU_EOUT_SIG_EN2)		Saf-Lat	
0x128	FCCU EOUT Signaling Enable Register 3 (FCCU_EOUT_SIG_EN3)		Saf-Lat	

1. In this column, R/W = read/write, R = read-only, and W = write-only.
2. No-Saf = Not Safety Relevant, Saf-Relv = Safety Relevant, Saf-Lat = Safety Latent

65.6.1 FCCU Control Register (FCCU_CTRL)

The FCCU_CTRL register allows to execute the following operations:

- moving the FCCU state from the NORMAL state into the CONFIG state.
- moving the FCCU state from the CONFIG state into the NORMAL state.
- reading or to clear the RF status register.
- reading the FCCU FSM status register.
- reading or to clear the FCCU freeze registers.
- reading the ALARM timer.
- reading the Watchdog timer.

Some critical operations requires a key as defined in the FCCU_CTRLK register.

Offset: 0x000 Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	DEBUG	0	OPS		0	OPR				
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1331. FCCU Control Register (FCCU_CTRL)

Table 1270. FCCU_CTRL field descriptions

Field	Description
DEBUG	Debug mode entry 0 Normal operation 1 Put FCCU into debug mode When Debug pin signal is asserted and this bit is set, FCCU moves into debug state and suspend the transition of the FSM by suspending timer elapse time effect.
OPS	Operation status 00 Idle 01 In progress 10 Aborted 11 Successful These bits can be read and cleared (via OP15 operation) by the software.

Table 1270. FCCU_CTRL field descriptions (continued)

Field	Description
OPR	Operation run 00000 No operation [OP0] 00001 Set the FCCU into the CONFIG state [OP1] 00010 Set the FCCU into the NORMAL state [OP2] 00011 Read the FCCU state (see the FCCU_STAT register) [OP3] 00100 Read the FCCU frozen status flags (see the N2AF_STATUS register) [OP4] 00101 Read the FCCU frozen status flags (see the A2FF_STATUS register) [OP5] 00110 Read the FCCU frozen status flags (see the N2FF_STATUS register) [OP6] 00111 Read the FCCU frozen status flags (see the F2AF_STATUS register) [OP7] 01000 Reserved 01001 Reserved 01010 Read the RF status register (see the FCCU_RF_S register) [OP10] 01011 Reserved 01100 RF status clear operation in progress (see the FCCU_RF_S register) [OP12] 01101 Clear the freeze status registers (see the freeze registers) [OP13] 01110 CONFIG to NORMAL FCCU state (configuration timeout) in progress [OP14] 01111 Clear the operation status (OPS=Idle) [OP15] 10000 Reserved 10001 Read the ALARM timer (see the FCCU_XTMR register) [OP17] 10010 RESERVED 10011 Read the CFG timer (see the FCCU_XTMR register) [OP19] 10100 Read the Error Pin low counter value (see *Section 65.6.21*) [OP20] 10101–11111: Reserved The SW application must not modify the OPR field (any write operation will be ignored) until the completion of the operation. Once the operation has been completed, the operation status (OPS) is set and the OPR field is automatically cleared (OPR = 000). The opcode OP11, OP12 must not be programmed. The OPR field is automatically set to OP11 or OP12 when the FCCU_RF_S registers are cleared by a write-clear operation into the related register. The opcode OP14 must not be programmed. The OPR field is automatically set to OP14 when the timeout occurs (FCCU_CFG_TO) during the configuration procedure => the FCCU state is automatically forced in NORMAL mode setting the default configuration. In this phase any write operation to the FCCU configuration registers is inhibited. The reserved operations (OP21 -OP30) are forbidden. In case of execution, they return an ABORT response without any side effect. The ABORT response occurs in the following cases: – wrong access (missing or wrong key) to the FCCU_RF_S register (clear operation OP12) – wrong access (missing or wrong key) to the FCCU_CTRL register (OP1, OP2 operation) – OP1 (CONFIG command) execution when FCCU state /= NORMAL or configuration locked – OP21-OP30 (RESERVED operations) execution The max timing needed for ABORT/SUCCESS response is 12 IRC cycles

65.6.2 FCCU CTRL Key Register (FCCU_CTRLK)

The FCCU_CTRLK register implements the key access for the operations OP1, OP2 according to the following sequence:

1. Write the key into the FCCU_CTRLK register.
2. Write the FCCU_CTRL register (operations OP1 or OP2).

Note: There shall not be any other operation in between the above steps and the 32 bits need to be writed in one cycle. so store FCCU_CTRLK, merge with OP1 or OP2 then send it.

The FCCU_CTRLK register is not readable, a 0x0000_0000 value is always returned in case of read operation. The key must be written by a word (32 bits) data write operation.

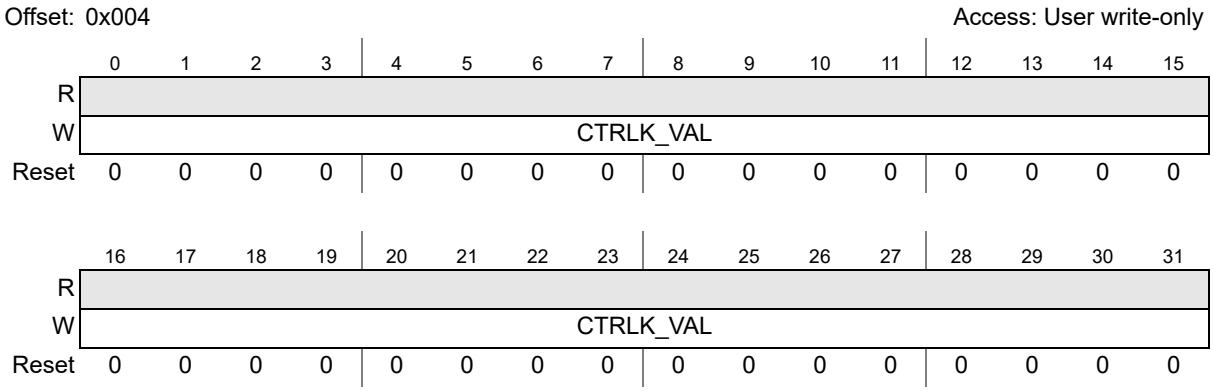


Figure 1332. FCCU CTRL Key Register (FCCU_CTRLK)

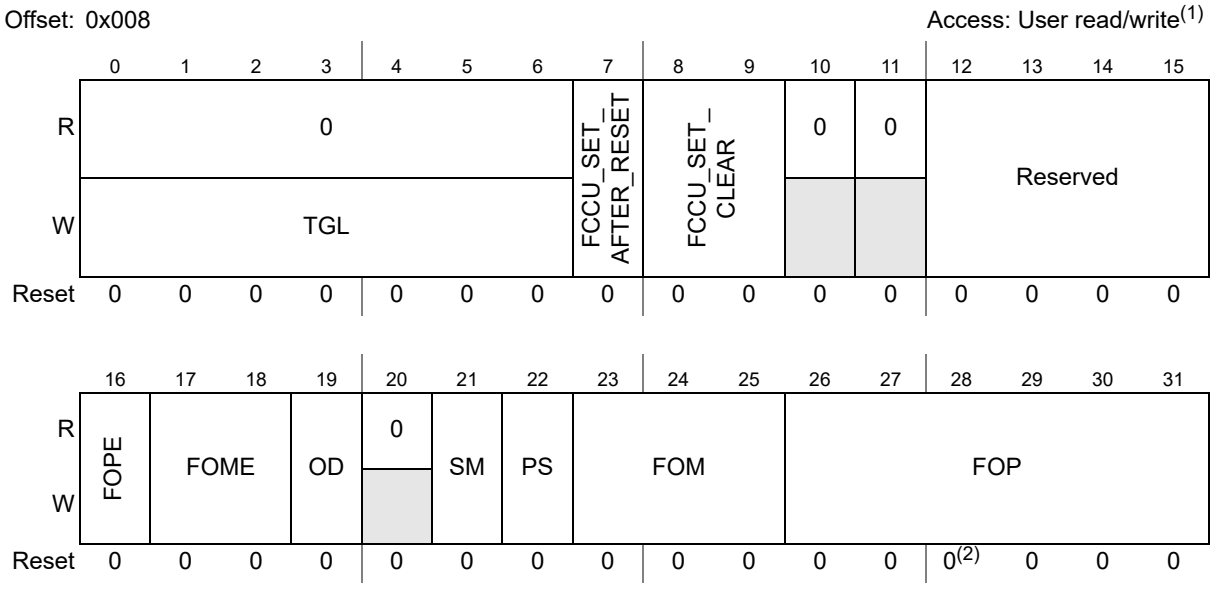
Table 1271. FCCU_CTRLK field descriptions

Field	Description
0:31 CTRLK_VAL	Control register key 0x913756AF Key for the operation OP1 0x825A132B Key for the operation OP2

65.6.3 FCCU Configuration Register (FCCU_CFG)

The FCCU_CFG register defines the global configuration for the FCCU module.

This register is writable only in the CONFIG state.



1. Writable only in the CONFIG state.
2. After powerup, reset value will be 0. If configuration timeout occurs then default value will be changed to 1.

Figure 1333. FCCU Configuration Register (FCCU_CFG)

Table 1272. FCCU_CFG field descriptions

Field	Description
0:6 TGL	EOUT toggling time(3) Active only if CFG.SM = 0 (switch mode = slow transition) MIN duration of EOUT pulse during FAULT -> NORMAL/ALARM and NORMAL/ALARM -> FAULT transitions Min = 1 µs, Max = 128 µs, Resolution = 1 µs
7 FCCU_SET_ AFTER_RESET	This bit controls the enable of the o/p error pin after reset lifts 0 FCCU's Error indication stops functioning and error pins are in HI-Z state(1). 1 write to start FCCU functioning. After power-on the error out pins shall be in high impedance(1). They will go to normal state only on software request, this bit is set to '1'. It is SW's responsibility to write this bit to 1 else the error pin stays in High-Z state after reset lifts and FCCU is not functioning.
8:9 FCCU_SET_ CLEAR	Error pin state can be controlled by these bits 01 Error Pin driven to logic 0. 11 Error Pin driven to logic 1 (write not possible when FCCU already in the T min timer phase or if FSM enters in FAULT state by capture of a fault). 00 FCCU acts independent of above SW control. 10 FCCU acts independent of above SW control. These bits clear(0) and set(1) the error pin. Higher priority is of the FCCU_SET_AFTER_RESET bit's capability to lead the error pins to Hi-Z(1). Switch to software configuration (01, 11) comes into effect when FCCU FSM leaves CONFIG state. After this is in effect, software maintains priority over FCCU FSM except for the case of '11' mentioned above. The switch back to FSM control is again done by writing into these fields with 00 or 10 by going to CONFIG state. This bit feature may not work with toggling protocols for example say EOUT pins are set high by SW using this bit followed by switching to a toggling protocol after returning control to FCCU FSM via setting this bit to 2'b00 or 2'b10, the user may find that EOUT pins do not toggle as was being expected.
16 FOPE	Fault Output Prescaler Extension of CFG[FOP] [5:0] For toggling frequency formula, please see FOP field description.
17:18 FOME	Fault Output Mode Extension 00 current behave (default) 01 option #1(2) 10 option #2(3) 11 option #3 (only active if CFG.FOM = 000)(4) - 000 and option#3 disabled => Dual rail mode (current behave) - 000 and option#3 enabled => option #3 enabled (GPIO mode in NORMAL/ALARM mode and DUAL-RAIL in FAULT mode)
19 OD	Open Drain Mechanism to select between Push-pull and Open drain(OD) mode for the error indicating pin(s). 0 push-pull 1 OD
21 SM	Switching mode 0 EOUT protocol (dual rail, time switching) slow switching mode. 1 EOUT protocol (dual rail, time switching) fast switching mode. SM has no effect on the bi-stable protocol.

Table 1272. FCCU_CFG field descriptions (continued)

Field	Description
22 PS	Polarity selection 0 FCCU Error output[1] active high, FCCU Error output[0] active low. 1 FCCU Error output[1] active low, FCCU Error output[0] active high. This bit can be read and written by the software. Active state here means a state indicating FAULT and applicable when pins are nontoggling in Error phase.
23:25 FOM	Fault Output Mode selection 000 Dual Rail (default state) [FCCU Error output[1:0]= outputs] 001 Time Switching [FCCU Error output[1:0]= output to be used] 010 Bi-Stable 011 Reserved 100 Reserved 101 Test0 (controlled by FCCU_EOUT register) [FCCU Error output[0]= input, FCCU Error output[1]= output] 110 Test1 (controlled by FCCU_EOUT register) [FCCU Error output[0]= output, FCCU Error output[1]= output] 111 Test2 (controlled by FCCU_EOUT register) [FCCU Error output[0]= output, FCCU Error output[1]= input] **Note:** In Test mode, a simple double-stage resynchronization stage is used to resynchronize the EOUT input/outputs on the system/IRCOSC clock.
26:31 FOP	Fault Output Prescaler FOP along with FOPE (FOPE being the MSB) defines the prescaler setting used to generate the EOUT protocol frequency. 0x00 Input clock frequency (RC oscillator clock) is divided by 2 x 2048 0x01 Input clock frequency (RC oscillator clock) is divided by 4 x 2048 0x02 Input clock frequency (RC oscillator clock) is divided by 6 x 2048 0x03 Input clock frequency (RC oscillator clock) is divided by 8 x 2048 0x04 Input clock frequency (RC oscillator clock) is divided by 10 x 2048 0x05 Input clock frequency (RC oscillator clock) is divided by 12 x 2048 0x06 Input clock frequency (RC oscillator clock) is divided by 14 x 2048 ... 0x7F Input clock frequency (RC oscillator clock) is divided by 256 x 2048 The equation below gives the EOUT frequency: EOUTfreq = RCOSCfreq / ((FOPe + 1) * 2 * 2048) where FOPe = {FOPE,FOP}

1. Actual value depends on the SOC settings at pad level.
2. SPI frame is SW controlled following a HW interrupt trigger (IRQ_EOUT) asserted by the FCCU (EOUT counter saturation) when in NORMAL/ALARM state. For details, please refer to *Figure 1328*.
3. SPI frame is DMA controlled following a DMA request (DMARQ_EOUT) asserted by the FCCU (EOUT counter saturation) when in NORMAL/ALARM state. An interrupt by SPI/DMA engine is asserted in order to update the data packet for the next watchdog refresh SPI controlled only via DMA in time division mode. For details, please refer to *Figure 1328*.
4. EOUT protocol is SW controlled in NORMAL/ALARM state and HW controlled in FAULT state (dual rail, in phase toggling with programmable delay) a HW interrupt trigger (IRQ_EOUT) asserted by the FCCU (EOUT counter saturation) when in NORMAL/ALARM state in order to generate the EOUT protocol via SW. As alternative an external TMR can be optionally used. EOUT dual rail protocol:
-NORMAL/ALARM : for example 10 ms toggling in opposite phase
-FAULT : for example 10 ms toggling in phase starting after a programmable delay T_tgl, slow switching mode (SM=0)
-FAULT : fast switching mode (SM =1) in phase (no periodic toggling). For details, please refer to *Figure 1329*

65.6.4 FCCU RF Configuration Register *n* (FCCU_RF_CFG*n*)

The FCCU_RF_CFG*n* registers contain the configuration of each recoverable fault in terms of fault recovery management. The configuration depends on the type of signaling of a fault event. HW recoverable faults should be configured only if a previous latching stage captures and hold the physical fault otherwise the fault can be lost. All the other faults should be configured as SW fault.

These registers are writable only in the CONFIG state.

Offset: 0x01C + *n**0x4 (*n*=0 to 3) Access: User read/write(1)

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	RFC31(2)	RFC30(2)	RFC29(2)	RFC28(2)	RFC27(2)	RFC26(2)	RFC25(2)	RFC24(2)	RFC23(2)	RFC22(2)	RFC21(2)	RFC20(2)	RFC19(2)	RFC18(2)	RFC17(2)	RFC16(2)
W																
Reset	The reset value is chip-specific; see the chapter that describes how modules are configured and connected															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	RFC15(2)	RFC14(2)	RFC13(2)	RFC12(2)	RFC11(2)	RFC10(2)	RFC9(2)	RFC8(2)	RFC7(2)	RFC6(2)	RFC5(2)	RFC4(2)	RFC3(2)	RFC2(2)	RFC1(2)	RFC0(2)
W																
Reset	The reset value is chip-specific; see the chapter that describes how modules are configured and connected															

1. Writable only in the CONFIG state
2. See *Table 1274* for register offset to channel number relationship

Figure 1334. FCCU RF Configuration Register *n* (FCCU_RF_CFG*n*)

Table 1273. FCCU_RF_CFG*n* field descriptions

Field	Description
0:31 RFC[31:0]	Recoverable fault configuration 0 HW recoverable fault 1 SW recoverable fault The recoverable fault configuration defines the fault recovery mode. HW recoverable faults are self recovered (status flag clearing and related) if the root cause (input fault) has been removed. SW recoverable faults are recovered (status flag clearing) by SW clearing of the related status flag.

Table 1274 shows FCCU RF configuration register channels.

Table 1274. FCCU RF configuration register channels

Address offset	Register name	Channel range (x) (bit location [0:31])
0x01C	FCCU_RF_CFG0	RFC[31:0]
0x020	FCCU_RF_CFG1	RFC[63:32]
0x024	FCCU_RF_CFG2	RFC[95:64]
0x028	FCCU_RF_CFG3	RFC[127:96]

65.6.5 FCCU RFS Configuration Register *n* (FCCU_RFS_CFG*n*)

The FCCU_RFS_CFG*n* registers contain the configuration of each recoverable fault in terms of fault reaction (short or long functional reset request pulse) when it is the root cause for the FAULT state transition.

These registers are writable only in the CONFIG state.

Offset: 0x04C +*n**0x4 (*n*=0 to 7) Access: User read/write(1)

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	RFSC15(2)		RFSC14(2)		RFSC13(2)		RFSC12(2)		RFSC11(2)		RFSC10(2)		RFSC9(2)		RFSC8(2)	
Reset	The reset value is chip-specific; see the chapter that describes how modules are configured and connected.															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	RFSC7(2)		RFSC6(2)		RFSC5(2)		RFSC4(2)		RFSC3(2)		RFSC2(2)		RFSC1(2)		RFSC0(2)	
Reset	The reset value is chip-specific; see the chapter that describes how modules are configured and connected.															

1. Writable only in the CONFIG state.
2. See *Table 1276* for register offset to channel number relationship.

Figure 1335. FCCU RFS Configuration Register *n* (FCCU_RFS_CFG*n*)

Table 1275. FCCU_RFS_CFG*n* field descriptions

Field	Description
0:31 RFSC[15:0]	Recoverable fault state configuration 00 No reset reaction 01 Short functional reset request pulse (FAULT state reaction) 10 Long functional reset request pulse (FAULT state reaction) 11 No reset reaction

Table 1276 shows FCCU **RFS** configuration register channels.

Table 1276. FCCU RFS configuration register channels

Address offset	Register name	Channel range (x) (bit location [0:31])
0x04C	FCCU_RFS_CFG0	RFSC[15:0]
0x050	FCCU_RFS_CFG1	RFSC[31:16]
0x054	FCCU_RFS_CFG2	RFSC[47:32]
0x058	FCCU_RFS_CFG3	RFSC[63:48]
0x05C	FCCU_RFS_CFG4	RFSC[79:64]
0x060	FCCU_RFS_CFG5	RFSC[95:80]
0x064	FCCU_RFS_CFG6	RFSC[111:96]
0x068	FCCU_RFS_CFG7	RFSC[127:112]

65.6.6 FCCU RF Status register *n* (FCCU_RF_S*n*)

The FCCU_RF_S*n* register contains the latched fault indication collected from the recoverable fault sources. Faults are latched also in the CONFIG state and independently from the enabling or reactions programmed for theRF.

No reactions are executed until the FCCU moves in the NORMAL state.

FCCU reacts and moves from the NORMAL state into the ALARM state only if the respective enable bit for a fault is set in the FCCU_RF_E*n* register and the respective enable bit for the timeout is set in the FCCU_TOEx register.

FCCU reacts and moves from the NORMAL or ALARM state into the FAULT state if the respective enable bit for a fault is set in the FCCU_RF_E*n* register and the respective enable bit for the timeout is disabled in the FCCU_TOEx register.

FCCU reacts and moves from the ALARM state into the FAULT state if the timeout (FCCU_TO register) is elapsed before to recovery the fault.

The timeout is stopped only when the FCCU returns in the NORMAL state.

The FCCU_RF_S*n* register is encoded respectively into the N2FF_STATUS or A2FF_STATUS register to freeze the entry condition in the FAULT state. The status bits of the FCCU_RF_S*n* register, configured as SW recoverable faults, can be cleared by the following locked sequence:

1. Write the proper key into the FCCU_RFK register
2. Clear the status (flag) bit RFSx => the opcode OP12 is automatically set into the FCCU_CTRL.OPR field
3. Wait for the completion of the operation (FCCU_CTRL.OPS field)
4. Read the FCCU_RF_S*n* register in order to verify the effective deletion and in case of failure to repeat the sequence

As result of the above sequence, in addition the FAULT interface provides support to clear the external fake FAULT root (see *Section 65.5.7*).

Note: There should not be any other operation in between the above steps.

The FCCU moves from the FAULT or ALARM state into the NORMAL state if all the source faults that caused the transition into the FAULT state have been removed (HW recoverable

fault) or cleared via SW (SW recoverable fault). In case of nested faults that are not all recovered, the FCCU will remain in the FAULT or ALARM state.

The SW application executes the FCCU_RF_S*n* read operation by the following sequence:

1. Set the OP10 operation into the FCCU_CTRL.OPR field
2. Wait for the completion of the operation (FCCU_CTRL.OPS field)
3. Read the FCCU_RF_S*n* register

In case of reconfiguration of the FCCU (CONFIG state), before to return in NORMAL state the pending status bits into the FCCU_RF_S*n* must be cleared in order to avoid a false transition in ALARM/FAULT state.

The following errors are ignored:

- to write a wrong key into the FCCU_RFK register
- to attempt to clear a HW recoverable error

Offsets: 0x080 + *n**0x4 (*n*=0 to 3) Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	RFS31(1)	RFS30(1)	RFS29(1)	RFS28(1)	RFS27(1)	RFS26(1)	RFS25(1)	RFS24(1)	RFS23(1)	RFS22(1)	RFS21(1)	RFS20(1)	RFS19(1)	RFS18(1)	RFS17(1)	RFS16(1)
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	The reset value is chip-specific; see the chapter that describes how modules are configured and connected.															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	RFS15(1)	RFS14(1)	RFS13(1)	RFS12(1)	RFS11(1)	RFS10(1)	RFS9(1)	RFS8(1)	RFS7(1)	RFS6(1)	RFS5(1)	RFS4(1)	RFS3(1)	RFS2(1)	RFS1(1)	RFS0(1)
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	The reset value is chip-specific; see the chapter that describes how modules are configured and connected.															

1. See *Table 1278* for register offset to channel number relationship. Refer to for the errors description.

Figure 1336. FCCU RF Status register *n* (FCCU_RF_S*n*)

Table 1277. FCCU_RF_S*n* field descriptions

Field	Description
0:31 RFS[31:0]	Recoverable fault status 0 No recoverable fault latched 1 Recoverable fault latched The status bits related to the recoverable fault configured as HW recoverable faults are read-only and the flag is self cleared when the fault source is removed.

Table 1278 shows FCCU **RF** status register channels.

Table 1278. FCCU RF status register channels

Address offset	Register name	Channel range (x) (bit location [0:31])
0x080	FCCU_RF_S0	RFS[31:0]
0x084	FCCU_RF_S1	RFS[63:32]
0x088	FCCU_RF_S2	RFS[95:64]
0x08C	FCCU_RF_S3	RFS[127:96]

65.6.7 FCCU RF Key register (FCCU_RFK)

The FCCU_RFK register implements the key access to clear the status flags of the FCCU_RF_S*n* register.

The status bits of the FCCU_RF_S*n* register, configured as SW recoverable faults, can be cleared by the following locked sequence:

1. Write the key into the FCCU_RFK register
2. Clear the status (flag) bit RFSx

The key must be written for each FCCU_RF_S*n* clear operation.

Note: There should not be any other operation in between the above steps.

The FCCU_RFK register is not readable; a 0x0000_0000 value is always returned in case of read operation.

Offset: 0x090 Access: User Write-only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R																
W	RFK_VAL															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R																
W	RFK_VAL															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1337. FCCU RF Key register (FCCU_RFK)

Table 1279. FCCU_RFK field descriptions

Field	Description
0:31 RFK_VAL	recoverable fault key = 0xAB34_98FE

65.6.8 FCCU RF Enable Register *n* (FCCU_RF_E*n*)

The FCCU_RF_E*n* registers enable the fault sources to allow a transition from the NORMAL into the FAULT or ALARM state. In case of fault masking, the respective status bit

into the FCCU_RF_S*n* register is anyway set (for debugging purposes), only the reaction is masked.

These registers are writable only in the CONFIG state.

Offsets: 0x094 + *n**0x4 (*n*=0 to 3) Access: User read/write(1)

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	RFE31(2)	RFE30(2)	RFE29(2)	RFE28(2)	RFE27(2)	RFE26(2)	RFE25(2)	RFE24(2)	RFE23(2)	RFE22(2)	RFE21(2)	RFE20(2)	RFE19(2)	RFE18(2)	RFE17(2)	RFE16(2)
W																
Reset	The reset value is chip-specific; see the chapter that describes how modules are configured and connected.															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	RFE15(2)	RFE14(2)	RFE13(2)	RFE12(2)	RFE11(2)	RFE10(2)	RFE9(2)	RFE8(2)	RFE7(2)	RFE6(2)	RFE5(2)	RFE4(2)	RFE3(2)	RFE2(2)	RFE1(2)	RFE0(2)
W																
Reset	The reset value is chip-specific; see the chapter that describes how modules are configured and connected.															

1. Writable only in the CONFIG state.
2. See *Table 1281* for register offset to channel number relationship.

Figure 1338. FCCU RF Enable Register *n* (FCCU_RF_E*n*)

Table 1280. FCCU_RF_E*n* field descriptions

Field	Description
0:31 RFE[31:0]	Recoverable fault enable 0 No actions following the respective recoverable fault assertion 1 FCCU moves to ALARM or FAULT state

Table 1281 shows FCCU **RF_E*n*** enable register channels.

Table 1281. FCCU RF_E*n* enable register channels

Address offset	Register name	Channel range (x) (bit location [0:31])
0x094	FCCU_RF_E0	RFE[31:0]
0x098	FCCU_RF_E1	RFE[63:32]
0x09C	FCCU_RF_E2	RFE[95:64]
0x0A0	FCCU_RF_E3	RFE[127:96]

65.6.9 FCCU RF Timeout Enable Register *n* (FCCU_RF_TOE*n*)

The FCCU_RF_TOE*n* registers enable a transition from the NORMAL state into the ALARM state if the respective recoverable fault is enabled (RFEx and RFTOEx are set). In case the respective timeout is disabled (RFTOEx is cleared) and the recoverable fault is enabled (RFEx is set) the FCCU moves into the FAULT state if the related recoverable fault is asserted. The timer (preset with the timeout value defined by FCCU_TO register) is started

when the FCCU moves into the ALARM state. If the fault is not recovered within the timeout the FCCU moves from the ALARM state to the FAULT state.

These registers are writable only in the CONFIG state.

Offsets: 0x0A4 + *n**0x4 (*n*=0 to 3) Access: User read/write(1)

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	RFTOE31(2)	RFTOE30(2)	RFTOE29(2)	RFTOE28(2)	RFTOE27(2)	RFTOE26(2)	RFTOE25(2)	RFTOE24(2)	RFTOE23(2)	RFTOE22(2)	RFTOE21(2)	RFTOE20(2)	RFTOE19(2)	RFTOE18(2)	RFTOE17(2)	RFTOE16(2)
W																
Reset	The reset value is chip-specific; see the chapter that describes how modules are configured and connected.															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	RFTOE15(2)	RFTOE14(2)	RFTOE13(2)	RFTOE12(2)	RFTOE11(2)	RFTOE10(2)	RFTOE9(2)	RFTOE8(2)	RFTOE7(2)	RFTOE6(2)	RFTOE5(2)	RFTOE4(2)	RFTOE3(2)	RFTOE2(2)	RFTOE1(2)	RFTOE0(2)
W																
Reset	The reset value is chip-specific; see the chapter that describes how modules are configured and connected.															

1. Writable only in the CONFIG state.
2. See *Table 1283* for register offset to channel number relationship.

Figure 1339. FCCU RF Timeout Enable Register *n* (FCCU_RF_TOE*n*)

Table 1282. FCCU_RF_TOE*n* field descriptions

Field	Description
0:31 RFTOE[31:0]	Fault timeout enable 0 FCCU moves into the FAULT state if the respective fault is enabled 1 FCCU moves into the ALARM state if the respective fault is enabled.

Table 1283 shows FCCU **RF** timeout enable register channels.

Table 1283. FCCU RF timeout enable register channels

Address offset	Register name	Channel range (x) (bit location [0:31])
0x0A4	FCCU_RF_TOE0	RFTOE[31:0]
0x0A8	FCCU_RF_TOE1	RFTOE[63:32]
0x0AC	FCCU_RF_TOE2	RFTOE[95:64]
0x0B0	FCCU_RF_TOE3	RFTOE[127:96]

65.6.10 FCCU RF Timeout Register (FCCU_RF_TO)

The FCCU_RF_TO register defines the preset value of the timer for the recovery of the recoverable faults (enabled). Once FCCU enters in ALARM state, following the assertion of a recoverable fault enabled (RFEx and RFTOEx are set), the timer starts the count down.

If the fault is not recovered within the timeout the FCCU moves from the ALARM state to the FAULT state. The alarm timeouts value should be programmed less than the FOSU_COUNT, else destructive resets may be generated by FOSU timeout.

This register is writable only in the CONFIG state.

Offset: 0x0B4 Access: User read/write(1)

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	TO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	TO															
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

1. Writable only in the CONFIG state.

Figure 1340. FCCU RF Timeout Register (FCCU_RF_TO)

Table 1284. FCCU_RF_TO field descriptions

Field	Description
0:31 TO	Recoverable fault timeout Timeout = (TO) x $T_{RC16MHz}$

65.6.11 FCCU CFG Timeout Register (FCCU_CFG_TO)

The FCCU_CFG_TO register defines the preset value of the watchdog timer for the recovery from the CONFIG state. Once FCCU enters in CONFIG state, following a SW request (OP1 opcode), the watchdog timer is initialized and starts the countdown if the reset is not asserted.

If the configuration is not completed within the timeout, the FCCU moves automatically from the CONFIG state to the NORMAL state and the default values for all the configuration register is restored. Configuration registers are those registers which can be written only in CONFIG state. Please see *Table 1269*. The watchdog timeout is clocked with the RC oscillator clock (16 MHz). The default timeout value is 4.096 ms. Longer activation of CONFIG state can lead to resets if a failure is indicated during the time the FCCU is in CONFIG state due to FOSU.

The FCCU_CFG_TO register is writable in any state excluded the CONFIG state as follows the execution of the OP1 opcode (NORMAL to CONFIG state) and until the completion of the OP2 opcode (CONFIG to NORMAL state).

In case of watchdog timeout the FCCU_CFG_TO register is not accessible until the OP14 operation (CONFIG to NORMAL) has been completed.

Offset: 0x0B8 Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	TO		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0

Figure 1341. FCCU CFG Timeout Register (FCCU_CFG_TO)

Table 1285 shows FCCU_CFG_TO field descriptions.

Table 1285. FCCU_CFG_TO field descriptions

Field	Description
29:31 TO	Configuration timeout Timeout = $T_{RC16MHz}$ x $2^{(TO + 10)}$ 000 Timeout = 64 µs ... 111 Timeout = 8.192 ms

65.6.12 FCCU IO Control Register (FCCU_EINOUT)

The FCCU_EINOUT register allows the following operations typically in NORMAL state:

- to control the FCCU Error output pin[1] output level when the FCCU is configured in “Test1” or “Test0” fault output mode (FCCU_CFG.FOM)
- to control the FCCU Error output pin[0] output level when the FCCU is configured in “Test1” or “Test2” fault output mode (FCCU_CFG.FOM)
- to observe the FCCU Error inputs level

Table 1286 shows Bi-stable encoding.

Table 1286. Bi-stable encoding

Mode = FCCU_CFG.FOM	FCCU_EOUT[0]	FCCU_EOUT[1]
Test1	output	output
Test2	output	input
Test0	input	output

Note: Due to the resynchronization stage of the EOUT interface, there is a latency of a few IRCOSC clock cycles following a write/read operation of the FCCU_EINOUT register.

Offset: 0x0BC Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	EIN1(1)	EIN0(1)	0	0	EOUT(2)	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. FCCU Error input x depends on the PAD configuration, that is, internal pull-up enabled as defaultor pull-down and therefore EINx bit is 1 or 0 upon reset.
2. EOUT protocol is SW controlled in NORMAL/ALARM state and HW controlled in FAULT state (dual rail, in phase toggling with programmable delay) a HW interrupt trigger (IRQ_EOUT) asserted by the FCCU (EOUT counter saturation) when in NORMAL/ALARM state in order to generate the EOUT protocol via SW. As alternative an external EOUTMR can be optionally used. EOUT dual rail protocol: NORMAL/ALARM : for example 10 ms toggling in opposite phase FAULT : for example10 ms toggling in phase starting after a programmable delay T_tgl, slow switching mode (SM=0) FAULT : fast switching mode (SM =1) in phase (no periodic toggling). For details, please refer to *Figure 1329*.

Figure 1342. FCCU IO Control Register (FCCU_EINOUT)

Table 1287. FCCU_EINOUT field descriptions

Field	Description
26 EIN1	Error input 1 0 when FCCU Error input[1] = 0 1 when FCCU Error input[1] = 1 The EIN1 captures the respective FCCU Error input[1] signal.
27 EIN0	Error input 0 0 when FCCU Error input[0] = 0 1 when FCCU Error input[0] = 1 The EIN0 captures the FCCU Error input[0] signal.
30:31 EOUT	See nested table below

Register		Signals		Description
EINOUT.EOUT0	EINOUT.EOUT0	FCCU_EOUT1	FCCU_EOUT0	
0	0	1 xor PS	0 xor PS	Opposite phase
0	1	0 xor PS	1 xor PS	Opposite phase
1	0	0 xor PS	0 xor PS	In phase
1	1	1 xor PS	1 xor PS	In phase

65.6.13 FCCU Status Register (FCCU_STAT)

The FCCU_STAT register includes the FCCU status for debugging/test purposes. The FCCU finite state machine (FSM) operates by the RC oscillator clock asynchronous with the System clock. The FCCU status read operation requires a safe mechanism operated by a HW/SW synchronization sequence. The SW application executes a FCCU status read operation by the following sequence:

1. Set the OP3 operation into the FCCU_CTRL.OPR field
2. Wait for the completion of the operation (FCCU_CTRL.OPS field)
3. Read the FCCU status (FCCU_STAT register)

Offset: 0x0C0 Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	PhysicError Pin		ESTAT	STATUS		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0

Figure 1343. FCCU Status Register (FCCU_STAT)

Table 1288. FCCU_STAT field descriptions

Field	Description
26:27 PhysicErrorPin	Physical Error Pin's state PhysicErrorPin[26] corresponds to error pin 0, PhysicErrorPin[27] corresponds to error pin 1. For each of these, the bit values have the following meanings: 0 Error Pin is at logical 0 1 Error Pin is at logical 1 During Fault state, a static or toggling value is observed based on selected protocol.

Table 1288. FCCU_STAT field descriptions (continued)

Field	Description
28 ESTAT	Current Error Pin Status SW can write this bit with a different value. The new value is valid for 1 clock cycle, thereafter it returns to indicate the actual value being driven from FCCU. For example in nonfaulty states, this bit reads 0 and can be set for one cycle to 1. Similarly in faulty state, this bit reads 1 and sw can write it to 0 and after one cycle the bit reads 1 again. 0 System is OPERATIONAL (OK). 1 System is in FAULT state.This state is taken from the FCCU Eout logic (That is, as late as possible).
29:31 STATUS	FCCU Status 000 NORMAL state 001 CONFIG state 010 ALARM state 011 FAULT state Other: UNKNOWN state

65.6.14 FCCU NA Freeze Status Register (N2AF_STATUS)

The N2AF_STATUS register contains a unique code to identify the recoverable source (FCCU recoverable fault x) that caused the state transition from the NORMAL state to the ALARM state. In case of multiple fault conditions the fault source cannot be identified. This register is for test/debug purposes.

The SW application executes the N2AF_STATUS read operation by the following sequence:

1. Set the OP4 operation into the FCCU_CTRL.OPR field
2. Wait for the completion of the operation (FCCU_CTRL.OPS field)
3. Read the N2AF_STATUS register

The SW application executes the N2AF_STATUS clear operation by the following sequence:

1. Set the OP13 operation into the FCCU_CTRL.OPR field
2. Wait for the completion of the operation (FCCU_CTRL.OPS field). (All the freeze registers are cleared by this operation)

Offset: 0x0C4 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0(1)	0(1)	0(1)	0(1)	0(1)	0(1)	0(1)	0(1)	0(1)	0(1)	0(1)	0(1)	0(1)	0(1)	0(1)	0(1)
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0(1)	0(1)	0(1)	0(1)	0(1)	0(1)	0(1)	0(1)	NAFS							
W									w1c							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. These are reserved bits. These bits are always read as ‘0’ and must always be written with ‘0’.

Figure 1344. FCCU NA Freeze Status Register (N2AF_STATUS)

Table 1289. N2AF_STATUS field descriptions

Field	Description
24:31 NAFS	Normal to Alarm Frozen Status 0x00 No transition from NORMAL to ALARM state 0x01 NORMAL to ALARM state transition cause => FCCU recoverable fault[0] 0x02 NORMAL to ALARM state transition cause => FCCU recoverable fault[1] 0x03 NORMAL to ALARM state transition cause => FCCU recoverable fault[2] ... 0x80 NORMAL to ALARM state transition cause => FCCU recoverable fault[127] 0xFF NORMAL to ALARM state transition cause => multiple FCCU recoverable faults

65.6.15 FCCU AF Freeze Status Register (A2FF_STATUS)

The A2FF_STATUS register contains a unique code to identify the timeout trigger (recoverable fault) that caused the state transition from the ALARM state to the FAULT state. In case of multiple fault conditions the fault source cannot be identified. This register is for test/debug purposes.

The SW application executes the A2FF_STATUS read operation by the following sequence:

1. Set the OP5 operation into the FCCU_CTRL.OPR field
2. Wait for the completion of the operation (FCCU_CTRL.OPS field)
3. Read the A2FF_STATUS register

The SW application executes the FCCU_AFFS clear operation by the following sequence:

1. Set the OP13 operation into the FCCU_CTRL.OPR field
2. Wait for the completion of the operation (FCCU_CTRL.OPS field)

All the freeze registers are cleared by this operation.

Offset: 0x0C8 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0(1)	0(1)	0(1)	0(1)	0(1)	0(1)	0(1)	0(1)	0(1)	0(1)	0(1)	0(1)	0(1)	0(1)	0(1)	0(1)
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0(1)	0(1)	0(1)	0(1)	0(1)	0(1)	AF_SRC		AFFS							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. These are reserved bits. These bits are always read as ‘0’ and must always be written with ‘0’.

Figure 1345. FCCU AF Freeze Status Register (A2FF_STATUS)

Table 1290. A2FF_STATUS field descriptions

Field	Description
22:23 AF_SRC	Fault source 00 No fault 01 Reserved 10 Recoverable fault 11 Multiple or recoverable, or both, faults
24:31 AFFS	Alarm to Fault Frozen Status 0x00 No transition from ALARM to FAULT state 0x01 ALARM to FAULT state transition cause => FCCU recoverable fault[0] timeout 0x02 ALARM to FAULT state transition cause => FCCU recoverable fault[1] timeout 0x03 ALARM to FAULT state transition cause => FCCU recoverable fault[2] timeout ... 0x80 ALARM to FAULT state transition cause => FCCU recoverable fault[127] timeout 0xFF ALARM to FAULT state transition cause => multiple FCCU recoverable faults timeout

65.6.16 FCCU NF Freeze Status Register (N2FF_STATUS)

The N2FF_STATUS register contains a unique code to identify the recoverable fault that caused the state transition from the NORMAL state to the FAULT state. In case of multiple fault conditions the fault source cannot be identified. This register is for test/debug purposes. The SW application executes the N2FF_STATUS read operation by the following sequence:

1. Set the OP6 operation into the FCCU_CTRL.OPR field
2. Wait for the completion of the operation (FCCU_CTRL.OPS field)
3. Read the N2FF_STATUS register

The SW application executes the N2FF_STATUS clear operation by the following sequence:

1. Set the OP13 operation into the FCCU_CTRL.OPR field
2. Wait for the completion of the operation (FCCU_CTRL.OPS field). (All the freeze registers are cleared by this operation)

Offset: 0x0CC Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0(1)	0(1)	0(1)	0(1)	0(1)	0(1)	0(1)	0(1)	0(1)	0(1)	0(1)	0(1)	0(1)	0(1)	0(1)	0(1)
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0(1)	0(1)	0(1)	0(1)	0(1)	0(1)	NF_SRC		NFFS							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. These are reserved bits. These bits are always read as ‘0’ and must always be written with ‘0’.

Figure 1346. FCCU NF Freeze Status Register (N2FF_STATUS)

Table 1291. N2FF_STATUS field descriptions

Field	Description
22:23 NF_SRC	Fault source 00 No fault 01 Reserved 10 Recoverable fault 11 Multiple and recoverable faults
24:31 NFFS	Normal to Fault Frozen Status 0x00 No transition from NORMAL to FAULT state 0x01 NORMAL to FAULT state transition cause => FCCU recoverable fault[0] 0x02 NORMAL to FAULT state transition cause => FCCU recoverable fault[1] 0x03 NORMAL to FAULT state transition cause => FCCU recoverable fault[2] ... 0x80 NORMAL to FAULT state transition cause => FCCU recoverable fault[127] 0xFF NORMAL to FAULT state transition cause => multiple FCCU recoverable faults

65.6.17 FCCU FA Freeze Status Register (F2A_STATUS)

The F2AF_STATUS register contains a unique code to identify the recoverable fault source (recoverable fault x) that caused the state transition from the FAULT state to the ALARM state. In case of multiple fault conditions the fault source cannot be identified. This register is for test/debug purposes.

The SW application executes the F2AF_STATUS read operation by the following sequence:

1. Set the OP7 operation into the FCCU_CTRL.OPR field
2. Wait for the completion of the operation (FCCU_CTRL.OPS field)
3. Read the N2FF_STATUS register

The SW application executes the F2AF_STATUS clear operation by the following sequence:

1. Set the OP13 operation into the FCCU_CTRL.OPR field
2. Wait for the completion of the operation (FCCU_CTRL.OPS field). All the freeze registers are cleared by this operation

Offset: 0x0D0 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0(1)	0(1)	0(1)	0(1)	0(1)	0(1)	0(1)	0(1)	0(1)	0(1)	0(1)	0(1)	0(1)	0(1)	0(1)	0(1)
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0(1)	0(1)	0(1)	0(1)	0(1)	0(1)	0(1)	0(1)	FAFS							
W									w1c							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. These are reserved bits. These bits are always read as ‘0’ and must always be written with ‘0’.

Figure 1347. FCCU FA Freeze Status Register (F2A_STATUS)

Table 1292. F2AF_STATUS field descriptions

Field	Description
24:31 FAFS	Fault to Normal Frozen Status 0x00 No transition from FAULT to NORMAL state 0x01 FAULT to ALARM state transition cause => FCCU recoverable fault[0] 0x02 FAULT to ALARM state transition cause => FCCU recoverable fault[1] 0x03 FAULT to ALARM state transition cause => FCCU recoverable fault[2] ... 0x80 FAULT to ALARM state transition cause => FCCU recoverable fault[127] 0xFF FAULT to ALARM state transition cause => multiple FCCU noncritical faults

65.6.18 FCCU RF Fake Register (FCCU_RFF)

The FCCU_RFF register contains a unique code to set a recoverable fault in mutually exclusive mode by the external FAULT interface signal setting. It allows the SW emulation of the recoverable faults, by the injection of the fault directly in the FAULT root, in order to verify the entire path and reaction. The fault injection mechanism is optional and requires a proper management as described in *Section 65.5.7*. The reaction following a fake recoverable fault cannot be masked. The FCCU_RFF is a write-only register with a set of codes corresponding to each recoverable fault injection.

Offset: 0x0DC Access: User Write-only

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0							
W										FRFC						
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1348. FCCU RF Fake Register (FCCU_RFF)

Table 1293. FCCU_RFF field descriptions

Field	Description
25:31 FRFC	Fake recoverable fault code 0x00 Fake recoverable fault injection at recoverable fault source 0 0x01 Fake recoverable fault injectionat recoverable fault source 1 0x02 Fake recoverable fault injection at recoverable fault source 2 … 0x7F Fake recoverable fault injection at recoverable fault source 127 **Note:** Only writing to this register, fake fault injection occurs, writing 00 and default value being zero give different results.

65.6.19 FCCU IRQ Status Register (FCCU_IRQ_STAT)

The FCCU_IRQ_STAT register defines the FCCU interrupt status register related to the following events:

- Configuration timeout error
- Alarm interrupt
- NMI interrupt

The external interrupt line (Interrupt request) is asserted if any interrupt status bit of the FCCU_IRQ_STAT is set and the respective enable bit of the FCCU_IRQ_EN register is also set.

The NMI and ALARM interrupts are asserted and cleared according to the FCCU state. The status bits of the FCCU_IRQ_STAT trace the status of the related interrupt lines.

Offset: 0x0E0 Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	DMA_REQ	IRQ_EOUT(1)(2)	NMI_STAT	ALRM_STAT	CFG_TO_STAT
W													w1c			w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. SPI frame is SW controlled following a HW interrupt trigger (IRQ_EOUT) asserted by the FCCU (EOUT counter saturation) when in NORMAL/ALARM state. For details, please refer to *Figure 1328*.
2. EOUT protocol is SW controlled in NORMAL/ALARM state and HW controlled in FAULT state (dual rail, in phase toggling with programmable delay) a HW interrupt trigger (IRQ_EOUT) asserted by the FCCU (EOUT counter saturation) when in NORMAL/ALARM state in order to generate the EOUT protocol via SW. As alternative an external EOUTMR can be optionally used. EOUT dual rail protocol: NORMAL/ALARM: for example 10 ms toggling in opposite phase FAULT : for example 10 ms toggling in phase starting after a programmable delay T_tgl, slow switching mode (SM=0) FAULT : fast switching mode (SM =1) in phase (no periodic toggling). For details, please refer to *Figure 1329*.

Figure 1349. FCCU IRQ Status Register (FCCU_IRQ_STAT)

Table 1294. FCCU_IRQ_STAT field descriptions

Field	Description
27 DMA_REQ	Status bit asserted when EOUT counter = T_TRG in NORMAL/ALARM mode (DMA_REQ(1) assertion). Cleared on DMA_ACK assertion.
28 IRQ_EOUT	Interrupt status bit asserted when EOUT counter = T_TRG in NORMAL/ALARM mode. Usable to assert an interrupt (if FCCU_IRQ_EN[IRQ_EOUT_EN] = 1) and to control via SW the EOUTx pins in NORMAL/ALARM mode.

Table 1294. FCCU_IRQ_STAT field descriptions (continued)

Field	Description
29 NMI_STAT	NMI Interrupt Status 0 NMI interrupt is OFF 1 NMI interrupt is ON
30 ALRM_STAT	Alarm Interrupt Status 0 Alarm interrupt is OFF 1 Alarm interrupt is ON
31 CFG_TO_STAT	Configuration Timeout Status 0 No configuration timeout error 1 Configuration timeout error

1. SPI frame is DMA controlled following a DMA request (DMARQ_EOUT) asserted by the FCCU (EOUT counter saturation) when in NORMAL/ALARM state. An interrupt by SPI/DMA engine is asserted in order to update the data packet for the next watchdog refresh SPI controlled only via DMA in time division mode. For details, please refer to *Figure 1328*.

65.6.20 FCCU IRQ Enable Register (FCCU_IRQ_EN)

The FCCU_IRQ_EN register defines the FCCU interrupt enable register related to the following events:

- Configuration timeout error

The external interrupt line “Interrupt request (miscellaneous conditions)” is asserted if any interrupt status bit of the FCCU_IRQ_STAT is set and the respective enable bit of the FCCU_IRQ_EN register is also set.

Offset: 0x0E4 Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	IRQ_EOUT_EN	0	0	CFG_TO_IEN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1350. FCCU IRQ Enable Register (FCCU_IRQ_EN)

Table 1295. FCCU_IRQ_EN field descriptions

Field	Description
28 IRQ_EOUT_EN	It enables the FCCU_IRQ_STAT[IRQ_EOUT]. 0 IRQ_EOUT disabled. 1 RQ_EOUT enabled.
31 CFG_TO_IEN	Configuration Timeout Interrupt Enable 0 Configuration timeout interrupt disabled. 1 Configuration timeout interrupt enabled.

65.6.21 FCCU XTMR Register (FCCU_XTMR)

The FCCU_XTMR register contains the read values of the Alarm or Watchdog Timer or EOUT pin low counter. These timers are clocked on the IRCOSC clock.

EOUTMR is the EOUT pin in logic 0 counter. After EOUT has been set to low (by SW or by a real fault), it loads (a value corresponding to T_min = 250 µs + value programmed in FCCU_DELTA_T register) and then starts down-counting and becomes 0 on the expiry of T_min window. It reloads on occurrence of another fault event which has EOUT signaling enabled. T_min feature is available only in Bi-Stable

The SW application executes the timer read operation by the following sequence:

1. Set any of the following operations into the FCCU_CTRL.OPR field:
   a) OP17
   b) OP19
   c) OP20
2. Wait for the completion of the operation (FCCU_CTRL.OPS field)
3. Read the FCCU_XTMR register

Table 1297 shows Timer state/value.

Offset: 0x0E8 Access: User read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	XTMR_VAL															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	XTMR_VAL															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1351. FCCU XTMR Register (FCCU_XTMR)

Table 1296. FCCU_XTMR field descriptions

Field	Description
0:31 XTMR_VAL	Alarm/Watchdog/Safe request timer The current timer value is measured in IRCOSC clock cycles. These bits can be read by the software.

Table 1297. Timer state/value

TIMER	CONFIG state	NORMAL state	ALARM state	FAULT state
ALARM	0x0000_0000	Initial value	Running	Idle/End of count
CFG	Running	0x0001_FFFF	0x0001_FFFF	0x0001_FFFF
EOUTMR	0x0000_0000	0x0000_0000	0x0000_0000	Running

65.6.22 FCCU MCS Register (FCCU_MCS)

The FCCU_MCS register contains a queue of the last four states of the MC. MCS0 is the latest one, while MCS3 is the oldest one. In addition a qualifier indicates if the FCCU is in the FAULT state when the MC state has been captured. The MC state is synchronous to the system clock and provided by a different module while the FCCU state is synchronous to the IRCOSC, therefore some uncertainty must be considered regarding the FAULT state indication.

Offset: 0x0EC Access: User read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	VL3	FS3	0	0	MCS3				VL2	FS2	0	0	MCS2			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	VL1	FS1	0	0	MCS1				VL0	FS0	0	0	MCS0			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1352. FCCU MCS Register (FCCU_MCS)

Table 1298. FCCU_MCS field descriptions

Field	Description
0 VL3	Valid It indicates that the correspondent MCSx and FSx fields are valid. 0 MCS3, FS3 fields are not significative. 1 MCS3, FS3 fields are significative.
1 FS3	Fault status It indicates that the correspondent MCSx field has been captured when the FCCU is in FAULT state. 0 MCS3 field captured in any state different from the FAULT state. 1 MCS3 field captured in FAULT state.
4:7 MCS3	State of MC modules The MCSx is the state of MC modules. MCS0 = latest state MCS3 = oldest state On any MC state change the previous MC states are shifted (MCS3 = MCS2, MCS2= MCS1, MCS1= MCS0) and the latest one is captured in MCS0. For a definition of these states, see the description of the MC modules in the chip reference manual.
8 VL2	Valid It indicates that the correspondent MCSx and FSx fields are valid. 0 MCS2, FS2 fields are not significative 1 MCS2, FS2 fields are significative
9 FS2	Fault status It indicates that the correspondent MCSx field has been captured when the FCCU is in FAULT state. 0 MCS2 field captured in any state different from the FAULT state 1 MCS2 field captured in FAULT state
12:15 MCS2	*State of MC modules* The MCSx is the state of MC modules. MCS0 = latest state MCS3 = oldest state On any MC state change the previous MC states are shifted (MCS3 = MCS2, MCS2= MCS1, MCS1= MCS0) and the latest one is captured in MCS0. For a definition of these states, see the description of the MC modules in the chip reference manual.
16 VL1	Valid It indicates that the correspondent MCSx and FSx fields are valid. 0 MCS1, FS1 fields are not significative. 1 MCS1, FS1 fields are significative.
17 FS1	Fault status It indicates that the correspondent MCSx field has been captured when the FCCU is in FAULT state. 0 MCS1 field captured in any state different from the FAULT state. 1 MCS field captured in FAULT state.

Table 1298. FCCU_MCS field descriptions (continued)

Field	Description
20:23 MCS1	*State of MC modules* The MCSx is the state of MC modules. MCS0 = latest state MCS3 = oldest state On any MC state change the previous MC states are shifted (MCS3 = MCS2, MCS2= MCS1, MCS1= MCS0) and the latest one is captured in MCS0. For a definition of these states, see the description of the MC modules in the chip reference manual.
24 VL0	Valid It indicates that the correspondent MCSx and FSx fields are valid. 0 MCS0, FS0 fields are not significative. 1 MCS0, FS0 fields are significative.
25 FS0	Fault status It indicates that the correspondent MCSx field has been captured when the FCCU is in FAULT state. 0 MCS0 field captured in any state different from the FAULT state. 1 MCS0 field captured in FAULT state.
28:31 MCS0	*State of MC modules* The MCSx is the state of MC modules. MCS0 = latest state MCS3 = oldest state On any MC state change the previous MC states are shifted (MCS3 = MCS2, MCS2= MCS1, MCS1= MCS0) and the latest one is captured in MCS0. For a definition of these states, see the description of the MC modules in the chip reference manual.

65.6.23 FCCU Transient Register (FCCU_TRANS_LOCK)

The FCCU_TRANS register is used for unlocking configuration by writing 0xBC. This register is available only in supervisor mode. This register is write only.

Offset: 0x0F0 Access: Supervisor Write-only

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0								
W									TRANSKEY							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1353. FCCU Transient Register (FCCU_TRANS_LOCK)

Table 1299. FCCU_TRANS_LOCK field descriptions

Field	Description
24:31 TRANSKEY	Transistion Unlocking Key value 0x00 No change ... 0xBB No change 0xBC Configuration unlocked 0xBD No change ... 0xFF No change

65.6.24 FCCU Permanent Lock Register (FCCU_PERMNT_LOCK)

The FCCU_PERMNT_LOCK register is used for locking configuration permanently by writing FFh. This register is available only in supervisor mode. The permanent lock can only be removed by applying a reset (not necessarily power on reset). This register is write only.

Offset: 0x0F4 Access: Supervisor Write-only

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0								
W									PERMNTKEY							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1354. FCCU Permanent Lock Register (FCCU_PERMNT_LOCK)

Table 1300. FCCU_PERMNT_LOCK field descriptions

Field	Description
24:31 PERMNTKEY	Transistion Locking Key value 0xFF Configuration Locked Any other value: No Change

65.6.25 FCCU Delta T Register (FCCU_DELTA_T)

The FCCU_DELTA_T register is used for programming the value of delta_T constant, in microsecounds. This register can be written only in CONFIG state.

Offset: 0x0F8 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	T_TRG(1)													
W																
Reset	0	0	1	0	0	1	1	1	0	0	0	0	1	1	1	1

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	DELTA_T_VAL													
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. EOUT protocol is SW controlled in NORMAL/ALARM state and HW controlled in FAULT state (dual rail, in phase toggling with programmable delay) a HW interrupt trigger (IRQ_EOUT) asserted by the FCCU (EOUT counter saturation) when in NORMAL/ALARM state in order to generate the EOUT protocol via SW. As alternative an external EOUTMR can be optionally used. EOUT dual rail protocol:
-NORMAL/ALARM : for example 10 ms toggling in opposite phase
-FAULT: for example 10 ms toggling in phase starting after a programmable delay T_tgl, slow switching mode (SM=0)
-FAULT: fast switching mode (SM =1) in phase (no periodic toggling). For details, please refer to *Figure 1329*

Figure 1355. FCCU Delta T Register (FCCU_DELTA_T)

Table 1301. FCCU_DELTA_T field descriptions

Field	Description
2:15 T_TRG	IRQ/DMA trigger time EOUT semi-period in FAULT mode if CFG.SM = 0 IRQ_EOUT interrupt time base in NORMAL/ALARM mode if T_TRG > 0 Min = 2 µs, Max = 16.385 µs, Resolution = 1 µs EOUT semi-period = T_TRG + 1 [µs]
18:31 DELTA_T_VAL	Value of Delta_T (in microseconds) Max. value can be 10,000 µs (10 ms)

65.6.26 FCCU IRQ Alarm Enable register *n* (FCCU_IRQ_ALARM_EN*n*)

These registers enable the corresponding IRQ alarm. This register is writable only in CONFIG state.

Offsets: 0x0FC + *n**0x4 (*n*=0 to 3) Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	IRQEN31(1)	IRQEN30(1)	IRQEN29(1)	IRQEN28(1)	IRQEN27(1)	IRQEN26(1)	IRQEN25(1)	IRQEN24(1)	IRQEN23(1)	IRQEN22(1)	IRQEN21(1)	IRQEN20(1)	IRQEN19(1)	IRQEN18(1)	IRQEN17(1)	IRQEN16(1)
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	IRQEN15(1)	IRQEN14(1)	IRQEN13(1)	IRQEN12(1)	IRQEN11(1)	IRQEN10(1)	IRQEN9(1)	IRQEN8(1)	IRQEN7(1)	IRQEN6(1)	IRQEN5(1)	IRQEN4(1)	IRQEN3(1)	IRQEN2(1)	IRQEN1(1)	IRQEN0(1)
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. See *Table 1303* for register offset to channel number relationship.

Figure 1356. FCCU IRQ Alarm Enable register *n* (FCCU_IRQ_ALARM_EN*n*)

Table 1302. FCCU_IRQ_ALARM_EN*n* field descriptions

Field	Description
0:31 IRQEN[31:0]	IRQ alarm enable 0 Alarm is disabled for error source x. 1 Alarm is enabled for error source x.

Table 1303 shows FCCU IRQ alarm enable register channels.

Table 1303. FCCU IRQ alarm enable register channels

Address offset	Register name	Channel range (x) (bit location [0:31])
0x0FC	FCCU_IRQ_ALARM_EN0	IRQEN[31:0]
0x100	FCCU_IRQ_ALARM_EN1	IRQEN[63:32]
0x104	FCCU_IRQ_ALARM_EN2	IRQEN[95:64]
0x108	FCCU_IRQ_ALARM_EN3	IRQEN[127:96]

65.6.27 FCCU NMI Enable registers *n* (FCCU_NMI_EN*n*)

These registers enable the NMI.This register is writable only in CONFIG state.

Offsets: 0x10C + *n**0x4 (*n*=0 to 3) Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	NMIEN31(1)	NMIEN30(1)	NMIEN29(1)	NMIEN28(1)	NMIEN27(1)	NMIEN26(1)	NMIEN25(1)	NMIEN24(1)	NMIEN23(1)	NMIEN22(1)	NMIEN21(1)	NMIEN20(1)	NMIEN19(1)	NMIEN18(1)	NMIEN17(1)	NMIEN16(1)
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	NMIEN15(1)	NMIEN14(1)	NMIEN13(1)	NMIEN12(1)	NMIEN11(1)	NMIEN10(1)	NMIEN9(1)	NMIEN8(1)	NMIEN7(1)	NMIEN6(1)	NMIEN5(1)	NMIEN4(1)	NMIEN3(1)	NMIEN2(1)	NMIEN1(1)	NMIEN0(1)
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. See *Table 1305* for register offset to channel number relationship.

Figure 1357. FCCU NMI Enable register *n* (FCCU_NMI_EN*n*)

Table 1304. FCCU_NMI_EN*n* field descriptions

Field	Description
0:31 NMIEN[31:0]	NMI enable 0 NMI is disabled for error (RF) source x. 1 NMI is enabled for error (RF) source x.

Table 1305 shows FCCU NMI enable register channels.

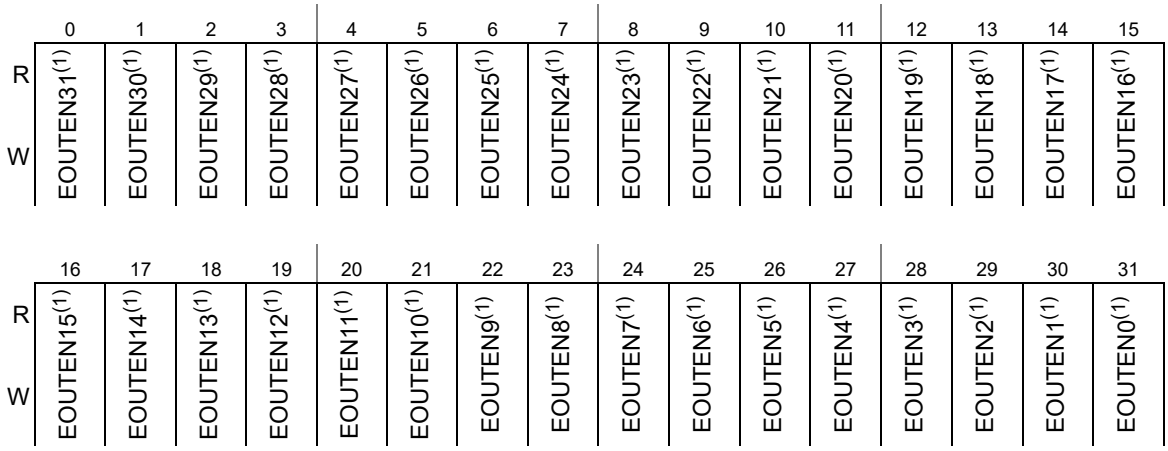
Table 1305. FCCU NMI enable register channels

Address offset	Register name	Channel range (x) (bit location [0:31])
0x10C	FCCU_NMI_EN0	NMIEN[31:0]
0x110	FCCU_NMI_EN1	NMIEN[63:32]
0x114	FCCU_NMI_EN2	NMIEN[95:64]
0x118	FCCU_NMI_EN3	NMIEN[127:96]

65.6.28 FCCU EOUT Signaling Enable registers *n* (FCCU_EOUT_SIG_EN*n*)

These registers enable the error out signaling. Disabling of EOUT signalling is possible for Bi-stable protocol only. This register is writable only in CONFIG state.

Offsets: 0x11C + n*0x4 (n=0 to 3) Access: User read/write



1. See *Table 1307* for register offset to channel number relationship.

Figure 1358. FCCU EOUT Signaling Enable register *n* (FCCU_EOUT_SIG_EN*n*)

Table 1306. FCCU_EOUT_SIG_EN*n* field descriptions

Field	Description
0:31 EOUTEN[31:0]	EOUT signaling enable 0 EOUT signaling is disabled for error (RF) source x for bistable protocol. Error pins are as if in nonfaulty state. 1 EOUT signaling is enabled for error (RF) source x.

Table 1307 shows FCCU EOUT signaling enable register channels.

Table 1307. FCCU EOUT signaling enable register channels

Address offset	Register name	Channel range (x) (bit location [0:31])
0x11C	FCCU_EOUT_SIG_EN0	EOUTEN[31:0]
0x120	FCCU_EOUT_SIG_EN1	EOUTEN[63:32]
0x124	FCCU_EOUT_SIG_EN2	EOUTEN[95:64]
0x128	FCCU_EOUT_SIG_EN3	EOUTEN[127:96]

65.7 FCCU Output Supervision Unit

The FOSU provides a supervision of the primary error notification path by analyzing FCCU behavior for correctness. It waits for any reaction of the FCCU in a fixed time window after an error arrives.

The intention of the FOSU is to provide a secondary error reaction path in most cases the FCCU fails but not to needlessly propagate an error which is already handled by the FCCU into a full chip reset. Only a failed primary error reaction (that is FCCU's failure) is a reason for the secondary reaction to take over (and generate a destructive reset request).

There is a 'do nothing' input coming from the FCCU that indicates that the FCCU is programmed for no reaction. It is a "static" input in the sense that it does not change after FCCU configuration. This signal is automatically de-asserted whenever the FCCU moves to CONFIG state and hence FOSU shall not be able to monitor faults occurring in CONFIG state.The FOSU masks the incoming faults with the 'do nothing' control from the FCCU, meaning that a fault is not captured by the FOSU if the 'do nothing' signal is asserted (that is an enabled fault). There is no minimum pulse width requirement on the fault indication other than what is required by the technology, which is the same as that of the FCCU. FOSU does not monitor FCCU for the case of faults occurring during CONFIG state. Also, the case of a continuously incoming disabled fault being enabled later, is not monitored.

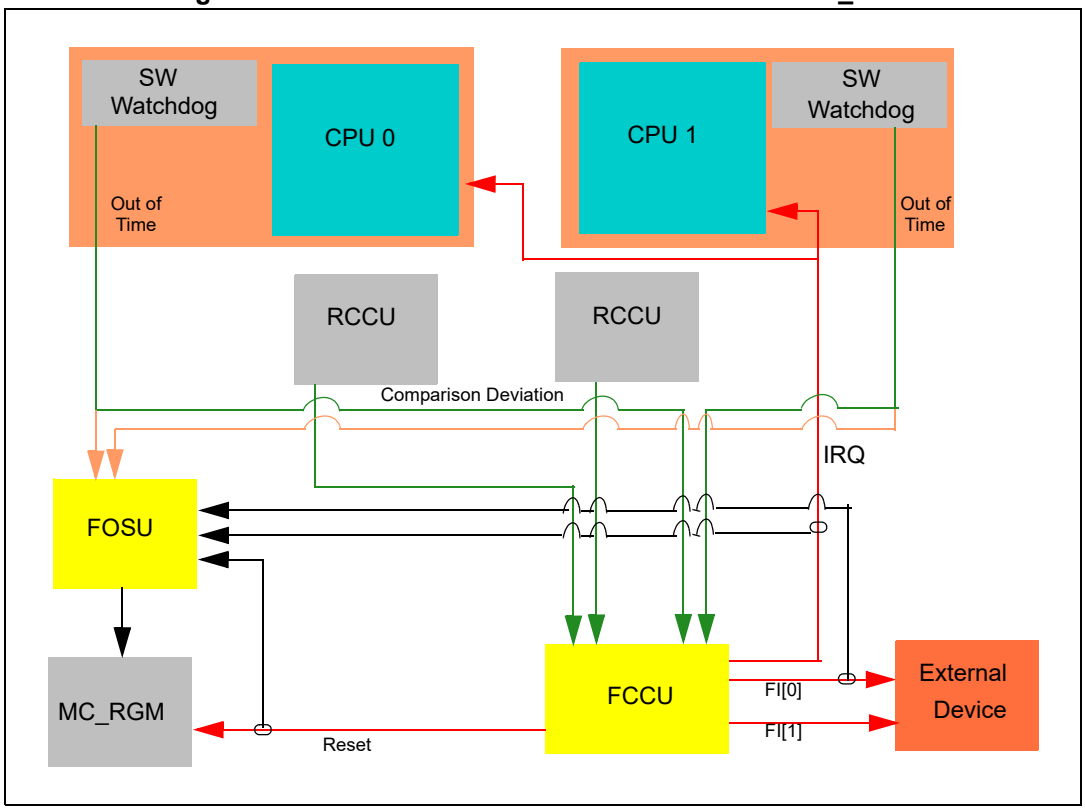
The FOSU contains a timer with a duration of FOSU_COUNT,driven by the IRCOSC.The timer is initialized and started on any captured, enabled fault. While the timer is running, any subsequent captured fault will neither restart nor reinitialize the timer. The timer is stopped when the FCCU shows any of the following reactions (the FCCU does not check whether the reaction is the configured one for the faults which occurred):

- Reset: Long or short functional reset.
- IRQ: NMI or Alarm
- Error out triggered (by FCCU or by SW).

When the timer is stopped, the fault capture logic is cleared in order to ensure that the timer is not restarted due to faults still ‘stuck’ in the capture logic. The timer will then be restarted by the next new failure indication. When the timer expires, the FOSU's failure indicator output is asserted after it ensures that the fault is enabled and the static “fccu program to do nothing” signal is deasserted. This is because FCCU uses settings after it exits CONFIG state, even if fault captured before the exit.
The FOSU's failure indicator output is connected to one of the MC_RGM's ‘destructive’ reset inputs, so its assertion will cause a reset sequence to be initiated starting at PHASE0. The FOSU module is reset with the same reset as is used by the FCCU (System reset), which is asserted on power-on, ‘destructive’, and long external resets. When this reset is asserted, the FOSU's capture logic is cleared, its timer is kept stopped and in a non-expired state, and its failure indicator output is deasserted.

Note: FOSU is triggered on assertion of enabled fault. In case the triggering fault is disabled, FOSU times-out without reaction. All intermediate faults are masked (not monitored) during this timeout cycle.

Figure 1359. FOSU connections to the FCCU and MC_RGM



It is important to remember that there will be no FCCU reaction to an error while the FCCU is in the CONFIG state. For this reason, the FCCU should not be kept in CONFIG for longer than the FOSU_COUNT duration. Otherwise, there is a risk that an incoming error report causing the FOSU to mistakenly see the FCCU as having failed, and then resets the MCU.

65.8 Use cases and limitations

65.8.1 Misconfigurations

The following configurations are appropriate:

1. If at least one reaction to fault is enabled by default then the corresponding fault should also be enabled by default.
2. Enable a fault but disabling all reactions is not a meaningful configuration from safety point of view.
3. Disable the fault and also its reactions. Applicable when a specific fault line is not of interest in a specific application scenario.
   Example: FCCU goes into ALARM state due to a failure, but the IRQ is disabled for that failure.
4. Software reset is optional.
5. The user must program the FCCU for a higher timeout value than the value hardcoded for FOSU. In this way FCCU reacts later than FOSU expects and as a result, FOSU generates a destructive reset request on its timeout, if a fault occurred when the FCCU was in CONFIG state.

65.8.2 Recommendations to configure FCCU

1. After Power on reset, where both system and FCCU get reset, following steps could be followed to configure FCCU:
   a) Check and clear any pending fault status.
   b) Verify FCCU is in NORMAL state, else repeat step(a) above.
   c) Configure FCCU.
2. After functional reset of the system, arising out of reset request from FCCU or other sources, the following steps could be followed to reconfigure FCCU:
   a) If active, wait for the Error out Tmin to expire.
   b) Check and clear fault status.
   c) Error pin moves to “non faulty” state, once fault status is cleared and Tmin expires.
   d) Reset FCCU by register bit in RGM. All FCCU registers and Error pin initialize to default state.
   e) Read and verify default value in FCCU_RF_Ex to ascertain reset of FCCU.
   f) Check and clear any new pending fault status until FCCU is in NORMAL state.
   g) Reconfigure FCCU.

Note: Software reset of FCCU by RGM_PRST5.FCCU_RST bit is optional since FCCU registers are not initialized during functional reset. RGM_PRST5.FCCU_RST has the same effect on FCCU as power on reset. If the bit is activated, it can change the state of the error pin, if the current state of the pin is different from default.

66 Self-Test Control Unit (STCU2)

66.1 Introduction

The self-test control unit (STCU2) is a comprehensive programmable hardware module that controls the self-test sequence applied both during the offline and/or online conditions. The hardware can manage the logic built-in self-test (LBIST) and the SRAM or ROM built-in self-test (MBIST) blocks of the device. The STCU2 includes the SSCM DCF bus to load the self-test parameters (LBIST or MBIST scheduling activity, LBIST setup, unrecoverable faults (UF) or recoverable faults (RF) management, CRC and MISR expected values, PLL management, and so on) from flash memory during the offline self-test phase. This interface is only able to write the configuration parameters and start the self-test execution once the STCU2 global reset has been applied. Register access by software is granted by an IPS interface to check the results of the offline self-test and to load or check the execution of the online self-test.

The STCU2 includes:

- A CRC block that can be optionally enabled to monitor the internal critical signals during the self-test run to increase the STCU2's self-checking capability.
- A programmable watchdog timer (WDG) to check the LBIST and MBIST self-test operations have completed within the assigned time slot or the STCU2 RUN or BYPASS bits have been programmed before watchdog timeout.
- Direct PLL control during the offline self-test sequence.
- Two sets of registers to separately collect the LBIST or MBIST results during offline and online self-test.

To increase the security of this module, a different security key sequence is also required during offline and online self-test to unlock write access to the STCU2 registers. The STCU2 also implements two switch off power-saving mechanisms:

- The global STCU2 clock after self-test sequence is completed.
- The register ITF clock when the WDG timeout is detected after a double security key sequence is applied during the application run (available only when IPS write access is enabled otherwise the clock is off).

66.2 Main features

- Double registers interface (SSCM DCF with priority over IPS).
- SSCM DCF write once register interface.
- IPS read and write (depending on SSCM enable parameters and STCU_CFG[WRP] bit) register interface.
- Programmable scheduler for LBIST or MBIST execution.
- LBIST concurrent/sequential execution.
- MBIST concurrent/sequential execution.
- Programmable LBIST delayed concurrent start.
- 32-bit CRC for STCU2 self-checking capability (enabled by STCU_CFG[CRCEN] bit)
- Online hardware self-test sequence abort capability.
- Programmable internal clock prescaler to reduce internal and MBIST or LBIST TCK clocks.
- Programmable PLL direct control during offline self-test sequence.
- PLL lock signal monitoring during offline and online self-test.
- Programmable WDG timer for self-check MBIST or LBIST time execution.
- Hard-coded WDG timeout to flag offline initialization errors.
- Hard-coded WDG timeout to auto-lock STCU2 write access after double key sequence.
- LBIST and SRAM or ROM MBIST offline status flags.
- STCU2 internal errors offline status flag.
- LBIST and SRAM or ROM MBIST online status flags
- STCU2 internal errors online status flag.
- Online hardware abort status flag
- LBIST and SRAM or ROM MBIST offline or online programmable failure severity (UF/RF).
- STCU2 internal errors offline or online programmable failure severity (UF/RF).
- STCU2 errors (UF/RF) lines for FCCU signaling.
- Redundant UF/RF generation logic to improve safety.
- FCCU UF/RF faults injection mechanism.
- CRCE register access to perform redundant check by software
- LBIST MISRE registers access to perform redundant software checks.
- Self-test bypass capability.
- Protection mechanism of the global write register based on double security key.
- STCU2 global auto power-saving when:
  - Self-test is completed.
  - bypass mode is selected.
  - WDT timeout is detected.
- STCU2 ITF/WDG clock wake-up mechanism when software application writes double security key sequence (available when STCU_CFG[WRP] bit is active).
- STCU2 ITF/WDG auto power-saving when hard-coded WDG timeout is detected after software application writes double security key sequence (available when STCU_CFG[WRP] bit is active).

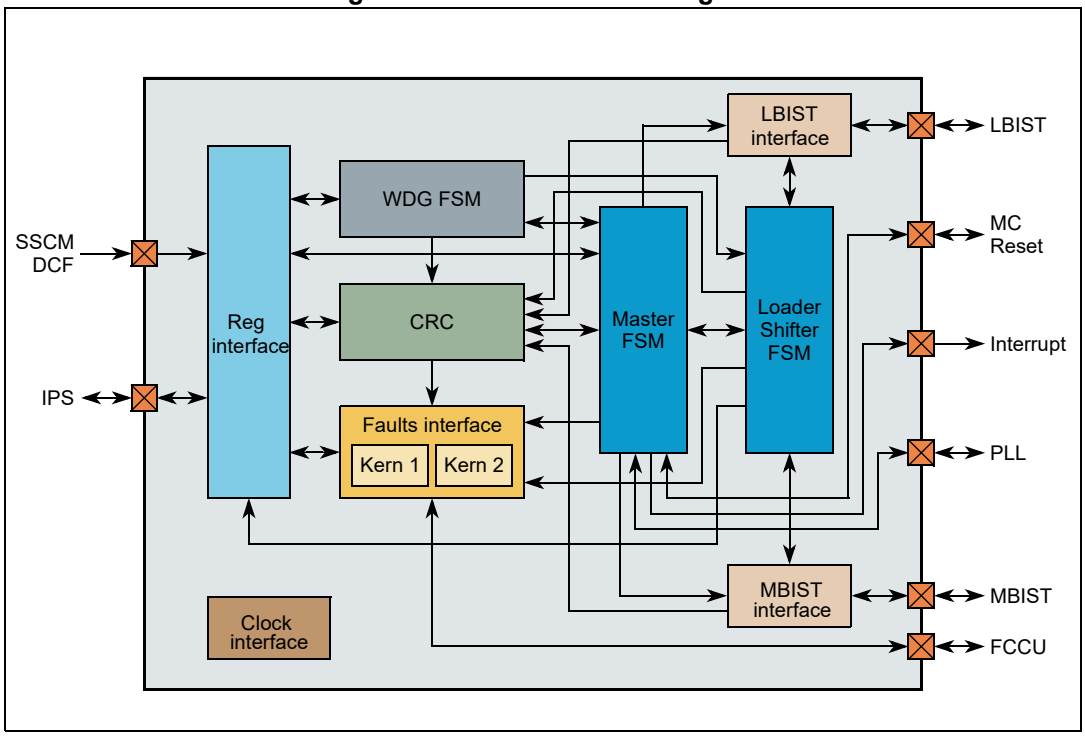
After completion of online self-test (MBIST or LBIST) a functional reset is generated.

So no partition level functional behavior is guaranteed once online self-test is triggered.

66.3 Block diagram

A top level diagram of the STCU2 module is given in *Figure 1360*.

Figure 1360. STCU2 block diagram



The STCU2 module includes the following submodules:

- **Reg interface**: This module includes the registers, the security key logic, the IPS and SSCM DCF bus interface
- **CRC**: This module is the core of the self-checking architecture of the STCU2. It samples a set of selected internal signals when STCU2 is running.
- **Fault interface**: This module collects the error conditions respective to LBIST or MBIST execution and STCU2 internal failures and, depending on the UF/RF configuration of each of them, sets the global STCU2 UF or RF flag. It also manages

the UF/RF fault lines and the set or clear injection mechanism provided by the FCCU. To improve the intrinsic safety of this critical logic, the generation logic is duplicated.

- **Clock Block**: This module manages the internal and the LBIST or MBIST TCK clock prescaler, the internal clock gating power saving and the wake-up clock feature.
- **WDG FSM:** This module includes two different watchdogs:
  - A hard-coded Watchdog used to auto-lock the STCU2 access, forcing a reset condition on the double security key registers.
  - A dual function watchdog:
  - After a reset event initializes the STCU2, it is used as a hard-coded watchdog timeout.
  - During LBIST or MBIST runs, it is used as a programmable watchdog timer to check the LBIST or MBIST have completed in the designated time slot.
- **STCU2 Master FSM**: This module includes the main FSM of the STCU2 used to coordinate and schedule all the operations performed during the self-test sequence.
- **STCU2 Load Shifter FSM:** This module includes the common shifter register and the respective state machine used to program the LBIST or MBIST registers and read back the data to be checked at the end of each self-test operation.
- **LBIST interface**: This module includes the interface between the LBIST engines and the STCU2 controllers of the device.
- **MBIST interface**: This module includes the interface between the MBIST controller and STCU2 controllers.

66.3.1 IPS bus interface

The IPS bus interface is a slave bus used for configuration purposes via CPU. The following bus read operations (contiguous byte enables) are supported:

- Word (32 bits) data read operations to any registers.
- Low and high half-words (16 bits, data[31:16] or data[15:0]) data read operations to any registers.
- Byte (8 bits, data[31:24] or data[23:16] or data[15:8] or data[7:0]) data read operations to any registers.
- Any other operation (free byte enables or other operations) has to be avoided.

The same operations are also supported in write mode under the following circumstances:

- The self-test sequence parameters are loaded before offline self-test is executed.
- The IPS access on some specific registers (see *Section 66.5*) is allowed by software only when the bit STCU_CFG[WRP] is cleared.
- Write operations are not supported and all STCU2 registers can only be read in all other circumstances except for the following R/W bits: STCU_CFG[WRP], STCU_ERR_STAT[UFSF] and STCU_ERR_STAT[RFSF]. See the respective register descriptions for additional details.

The STCU2 module generates a transfer error in the following cases:

- Any write/read access to the register addresses not mapped on the peripheral but included in the address space of the peripheral.
- Any write/read operation different from byte/halfword/word (free byte enables or other operations) on each register.
- Any write operation on double security key register applying a wrong sequence of keys (the two write operations cannot be interleaved with other accesses to STCU2 registers).
- Any write operation performed on a register when the double security keys have not been applied.
- Any write operation performed on registers when the STCU_CFG[WRP] bit is set and the access is performed through software (IPS interface).
- Any write operation performed through software (IPS interface) on row registers.
- Any write operation performed offline on registers where writes are only allowed during the online self-test phase when STCU_CFG[WRP] = 1.
- Any write operation performed on read only registers.

The registers of the STCU2 module are accessible (read/write) in each access mode: user, supervisor or test.

A transfer error is not generated for write operations to bits marked *Reserved*.

For more details regarding register protection features, see *Section 66.5*.

66.4 Functional description

66.4.1 FSM description

The module has three state machines that work together:

- Master State Machine: the core unit of the STCU2 module; it coordinates all the self-test operations and the other state machines.
- Loader/Shifter State Machine: used to program the MBIST and the LBIST parameters and to retrieve the respective results depending on the parameters stored in the STCU2 registers and under the control of the master state machine.
- Watchdog State Machine: evaluates all the schedule time for MBIST and LBIST and the timeouts in case of incorrect STCU2 programming.

66.4.2 Reset management

The asynchronous reset line initializes the STCU2 module status, forcing the offline self-test condition and the cleaning up of all the registers and state machines of the module.

The reset signals generated by the STCU2 module at the end of the self-test execution phase depends on the self-test conditions reported in the following subsections.

66.4.2.1 Offline reset generation

At the end of the offline self-test execution, the STCU2 raises the global functional reset.

66.4.2.2 Online reset generation (only MBIST)

At the end of the execution, the online execution of the MBIST generates a global functional reset.

66.4.2.3 Online Reset generation (LBIST enabled)

At the end of the execution, the online execution of the LBIST generates a global functional reset.

66.4.3 LBIST and MBIST scheduling

The STCU2 module is designed to be very flexible, allowing the programming of the parallel or serial execution of the MBIST or LBIST depending on the power, timing and coverage constraints. The limitation in the programming flexibility is described in *Section 66.7.4*.

The mechanism used to provide this flexibility is a linked list where the starting pointer is the bitfield STCU_CFG[PTR]. The first LBIST is mapped on 0x00, the second on 0x01, and so on. The first MBIST is mapped at the address 0x10, the second on 0x11 and so on. The additional pointers are in the bitfield STCU_LB_CTRL*x*[PTR] for LBIST*x* (where *x* is the selected LBIST) and STCU_MB_CTRL*y*[PTR] for MBIST (where *y* is the selected MBIST), and have to be filled depending on the selected sequence of run. The additional bitfields STCU_LB_CTRL[CSM] and STCU_MB_CTRL[CSM] provide the flexibility to run the chosen set of the LBIST or MBIST concurrently or sequentially, or to close the linked list by setting the NIL pointer. For more details, see *Section 66.5.31* and *Section 66.5.43*.

66.4.4 ABORT management

The STCU2 module provides online self-test execution abort. When the abort is detected, the online self-test operation stops running and the status of the currently running MBIST or LBIST is saved in the respective registers to provide intermediate results that might be useful in case of debug or a concurrent run. For an LBIST run, see *Section 66.4.2*, regarding reset specific management when abort is detected.

66.4.4.1 Hardware ABORT management

The STCU2 enters this abort condition as a consequence of a functional reset that requires the self-test execution to stop.

When hardware abort is detected, the STCU_ERR_STAT[ABORTHW] bit is set. This flag allows the software to diagnose what occurred during the online self-test execution run.

66.4.5 PLL interface

The STCU2 module is able to directly control the PLL during the offline self-test operations depending on the status of the STCU_RUN[MBPLLEN] and STCU_RUN[LBPLLEN] as described in *Section 66.5.2*.

When STCU_RUN[RUN] is set to start the offline self-test operations, the STCU2 takes control of the PLL and changes the PLL clock configurations parameters according to the values programmed in the STCU_PLL_CFG register.

When the STCU_RUN[MBPLLEN] is active and the MBIST is currently selected, the STCU2 waits for the PLL lock. As soon as the lock occurs, the STCU forces the clock to switch to the PLL output clock, allowing the MBIST to proceed at the PLL output frequency. At the end

of the MBIST run, the clock source reverts to the original one. The same behavior applies when STCU_RUN[LBPLLEN] is active and LBIST is currently selected.

66.4.6 FCCU interface

The FCCU interface is the hardware flag mechanism to the system, indicating the occurrence of an unrecoverable failure (UF) or a recoverable failure (RF) during the self-test sequence.

To diagnose physical defects on the two fault signals, a fault injection mechanism is also provided. The FCCU interface allows the user application to check the integrity of the UF and RF connection lines between the STCU2 and the FCCU. Refer to the description of FCCU fault injection mechanism to understand how the UF/RF set and clear mechanism works.

66.4.7 Watchdogs

The STCU2 implements three different watchdogs to ensure that operations are finished on time.

66.4.7.1 Initialization sequence

The STCU2 has two initialization operating modes:

- Program and run the self-test sequence (safety mode).
- Bypass the self-test sequence by setting the STCU_RUN[BYP] bit.

If there are faults during the initialization phase which prevent the selection of one of these two operating modes, a hard-coded watchdog timeout flags the incorrect behavior. See *Section 66.5.6* for more details.

66.4.7.2 LBIST and MBIST scheduling

The LBIST and MBIST execution time is configured as described in *Section 66.5.6* to account for the overall execution time of the self-test sequence. If the selected LBISTs or MBISTs are not yet completed:

- During an offline self-test assigned time, the current LBISTs or MBISTs execution is interrupted and a failure is flagged in STCU_ERR_STAT[WDTO] and STCU_MBEL/M/H or STCU_LBE registers.
- During an online self-test, the current LBISTs or MBISTs execution is interrupted and a failure is flagged in STCU_ERR_STAT[WDTOSW] and STCU_MBELSW/MSW/HSW or STCU_LBESW.

In case of multiple sequential runs in the same online or offline session and the timeout occurs in the middle of a sequential run, the next sequential run is skipped and the execution ends with the current updated status of the registers reported above.

In all cases, the STCU2 reset generation is managed in one of the modes described in *Section 66.4.2*.

66.4.7.3 Write access timeout

The access on the STCU2 registers during the self-test configuration phase (both online or offline) is protected by a key mechanism to prevent any unwanted access, as described in the STCU_SKC register description. In addition to this safety feature, there is a hardware

watchdog that, after 1024 STCU2 clock cycles, locks access and requires reinsertion of the STCU_SKC keys.

This feature is particularly useful if STCU_CFG[WRP] = 0 during the software self-test configuration as the software application may enable write access to the STCU2 registers.

66.4.8 CRC

The CRC block is used to increase the intrinsic coverage of the STCU2 module and implements a 32-bit ethernet protocol polynomial to evaluate the signature of the most important and critical internal signals. This feature is switched off by default but it can be turned on setting the STCU_CFG[CRCEN] bit. Once this bit is set, the expected signature has to be programmed in the STCU_CRCE register. At the end of the self-test execution, the STCU2 updates the content of the STCU_CRCR register with the evaluated signature and compares this value with the expected one previously programmed in STCU_CRCE. In case of mismatches, a UF/RF condition is flagged depending on the fault mapping selected in the STCU_ERR_FM register.

66.4.9 SSCM interface

The SSCM interface is used to program the STCU2's configuration parameters without CPU intervention after a reset trigger event initializes the STCU2. This bus interface has priority over the IPS.

66.5 Register description

The STCU2 registers are listed in *Table 1308*.

Table 1308. STCU2 register set

Address offset	Register Name	Location
0x0000	STCU2 Run Register (STCU_RUN)	*Section 66.5.1*
0x0004	STCU2 Run Software Register (STCU_RUNSW)	*Section 66.5.2*
0x0008	STCU2 SK Code Register (STCU_SKC)	*Section 66.5.3*
0x000C	STCU2 Configuration Register (STCU_CFG)	*Section 66.5.4*
0x0010	STCU2 PLL Configuration Register (STCU_PLL_CFG)	*Section 66.5.5*
0x0014	STCU2 Watchdog Register Granularity (STCU_WDG)	*Section 66.5.6*
0x0018–0x001B	Reserved	
0x001C	STCU2 CRC Expected Status Register (STCU_CRCE)	*Section 66.5.7*
0x0020	STCU2 CRC Read Status Register (STCU_CRCR)	*Section 66.5.8*
0x0024	STCU2 Error Register (STCU_ERR_STAT)	*Section 66.5.9*
0x0028	STCU2 Error FM Register (STCU_ERR_FM)	*Section 66.5.10*
0x002C	STCU2 offline LBIST Status Register (STCU_LBS)	*Section 66.5.11*
0x0030	STCU2 offline LBIST End Flag Register (STCU_LBE)	*Section 66.5.12*
0x0034	STCU2 online LBIST Status Register (STCU_LBSSW)	*Section 66.5.13*
0x0038	STCU2 online LBIST End Flag Register (STCU_LBESW)	*Section 66.5.14*

Table 1308. STCU2 register set (continued)

Address offset	Register Name	Location
0x003C	Reserved	
0x0040	STCU2 LBIST Unrecoverable FM Register (STCU_LBUFM)	*Section 66.5.15*
0x0044	STCU2 offline MBIST Status Low Register (STCU_MBSL)	*Section 66.5.16*
0x0048	STCU2 offline MBIST Status Medium Register (STCU_MBSM)	*Section 66.5.17*
0x004C	STCU2 offline MBIST Status High Register (STCU_MBSH)	*Section 66.5.18*
0x0050	STCU2 offline MBIST End Flag Low Register (STCU_MBEL)	*Section 66.5.19*
0x0054	STCU2 offline MBIST End Flag Medium Register (STCU_MBEM)	*Section 66.5.20*
0x0058	STCU2 offline MBIST End Flag High Register (STCU_MBEH)	*Section 66.5.21*
0x005C	STCU2 online MBIST Status Low Register (STCU_MBSLSW)	*Section 66.5.22*
0x0060	STCU2 online MBIST Status Medium Register (STCU_MBSMSW)	*Section 66.5.23*
0x0064	STCU2 online MBIST Status High Register (STCU_MBSHSW)	*Section 66.5.24*
0x0068	STCU2 online MBIST End Flag Low Register (STCU_MBELSW)	*Section 66.5.25*
0x006C	STCU2 online MBIST End Flag Medium Register (STCU_MBEMSW)	*Section 66.5.26*
0x0070	STCU2 online MBIST End Flag High Register (STCU_MBEHSW)	*Section 66.5.27*
0x0074	STCU2 MBIST Unrecoverable FM Low Register (STCU_MBUFML)	*Section 66.5.28*
0x0078	STCU2 MBIST Unrecoverable FM Medium Register (STCU_MBUFMM)	*Section 66.5.29*
0x007C	STCU2 MBIST Unrecoverable FM High Register (STCU_MBUFMH)	*Section 66.5.30*
LBIST registers (n = 0 to 3)		
0x0100 + *n* * 0x40	STCU2 LBIST Control *n* Register (STCU_LB_CTRL*n*)	*Section 66.5.31*
0x0104 + *n* * 0x40	STCU2 LBIST PC Stop *n* Register (STCU_LB_PCS*n*)	*Section 66.5.32*
0x0108 + *n* * 0x40	STCU2 LBIST PRPG Low *n* Register (STCU_LB_PRPGL*n*)	*Section 66.5.33*
0x010C + *n* * 0x40	STCU2 LBIST PRPG High *n* Register (STCU_LB_PRPGH*n*)	*Section 66.5.34*
0x0110 + *n* * 0x40	STCU2 offline LBIST MISR Expected Low *n* Register (STCU_LB_MISREL*n*)	*Section 66.5.35*
0x0114 + *n* * 0x40	STCU2 offline LBIST MISR Expected High *n* Register (STCU_LB_MISREH*n*)	*Section 66.5.36*
0x0118 + *n* * 0x40	STCU2 offline LBIST MISR Read Low *n* Register (STCU_LB_MISRRL*n*)	*Section 66.5.37*
0x011C + *n* * 0x40	STCU2 offline LBIST MISR Read High *n* Register (STCU_LB_MISRRH*n*)	*Section 66.5.38*
0x0120 + *n* * 0x40	STCU2 online LBIST MISR Expected Low *n* Register (STCU_LB_MISRELSW*n*)	*Section 66.5.39*
0x0124 + *n* * 0x40	STCU2 online LBIST MISR Expected High *n* Register (STCU_LB_MISREHSW*n*)	*Section 66.5.40*
0x0128 + *n* * 0x40	STCU2 online LBIST MISR Read Low *n* Register (STCU_LB_MISRRLSW*n*)	*Section 66.5.41*
0x012C + *n* * 0x40	STCU2 online LBIST MISR Read High *n* Register (STCU_LB_MISRRHSW*n*)	*Section 66.5.42*

Table 1308. STCU2 register set (continued)

Address offset	Register Name	Location
MBIST registers (k = 0 to 21)		
0x0600 + *k* * 0x4	STCU2 MBIST Control *k* Register (STCU_MB_CTRL*k*)	*Section 66.5.43*

66.5.1 STCU2 Run Register (STCU_RUN)

The STCU_RUN register defines the RUN bit to start the offline self-test procedure if some configuration bits are required *only* during this self-test phase.

The R/W fields in this register are readable at any time. However, you can write to these fields only when offline self-test phase is still active.

Offset: 0x0000 Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	BYP	MBPLLEN	LBPLLEN	0	0	0	0	0	0	0	RUN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1361. STCU2 Run Register (STCU_RUN)

Table 1309. STCU_RUN field descriptions

Field	Description
21 BYP	Bypass mode 0 Offline self-test is active. If the STCU_RUN[RUN] bit is not set before the hard-coded WDG timeout, the WDTO error is generated in the STCU_ERR_STAT register. 1 Offline self-test is bypassed and access to STCU2 is locked until the STCU_SCK register is written according to the request access mode.
22 MBPLLEN	Offline MBIST with PLL enabled 0 Offline MBIST is executed without using the on-chip PLL. 1 Offline MBIST is executed enabling the on-chip PLL control interface selecting the parameters defined in STCU_PLL_CFG register.
23 LBPLLEN	Offline LBIST with PLL enabled 0 Offline LBIST is executed without using the on-chip PLL. 1 Offline LBIST is executed enabling the on-chip PLL control interface selecting the parameters defined in STCU_PLL_CFG register.
31 RUN	RUN The RUN bit is automatically cleared by the STCU2 when the offline self-test procedure has been completed. 0 Idle 1 Offline self-test procedure is running

66.5.2 STCU2 Run Software Register (STCU_RUNSW)

The STCU_RUNSW register defines the RUN bit to start the online self-test procedure if some configuration bits are required *only* during this self-test phase.

The R/W fields in this register are readable at any time. However, you can write to these fields only when both of the following conditions are true:

- STCU_CFG[WRP] = 0.
- Online self-test phase is active. The online condition starts when the offline condition finishes. In this condition, the system is alive and the SW can program the STCU.

Offset: 0x0004 Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	MBSWPLLEN	LBSWPLLEN	0	0	0	0	0	0	0	RUNSW
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1362. STCU2 Run Software Register (STCU_RUNSW)

Table 1310. STCU_RUNSW field descriptions

Field	Description
22 MBSWPLLEN	Online MBIST with PLL enabled 0 Online MBIST is executed without using the on-chip PLL. 1 Online MBIST is executed using the PLL configuration provided by software. STCU2 does not take PLL control but monitors the PLL lock signal to check if PLL is working correctly.
23 LBSWPLLEN	Online LBIST with PLL enabled 0 Online LBIST is executed without using the on-chip PLL. 1 Online LBIST is executed using the PLL configuration provided by software. STCU2 does not take the PLL control but monitors the PLL lock signal to check if PLL is working correctly.
31 RUNSW	RUNSW The RUNSW bit is automatically cleared by the STCU2 when the online self-test procedure has been completed. 0 Idle 1 Online self-test procedure is running

66.5.3 STCU2 SK Code Register (STCU_SKC)

The STCU_SKC register implements the security key code mechanism needed for write access to the other STCU2 registers. In order to unlock the STCU2 access after an STCU2

asynchronous reset or the end of the STCU2 run, the software (IPS bus) or the SSCM interfaces must apply the following sequence:

- Write Key1 in the STCU_SKC register.
- Write Key2 in the STCU_SKC register.

Depending on the offline or online test step, the two keys will be different. Byte write operation is not allowed as the full key has to be recognized as one unit.

Note: *Note that after the self-test sequence has been completed or the bypass feature has been enabled (by setting the STCU_RUN[BYP] bit), the SSCM interface is no longer available.*

In case of invalid access or sequence (Key1 and Key2 must be applied consecutively), a transfer error on the IPS or SSCM bus (causing SSCM.CER assertion) is asserted depending on the selected source.

If the STCU2 register access lasts more cycles than defined in the hard-coded WDG timeout, STCU2 write access is locked and the WDG and register ITF clocks are switched off. Also in this case the correct sequence (Key1 and Key2) must be applied in order to re-enable write access to the STCU2 and the WDG and register ITF clocks.

To extend the STCU2 register access cycles before the hard-coded WDG timeout expires, only Key2 needs to be applied. The effect of this write operation is to re-initialize the WDG timeout counter. Key1 must not be applied or a transfer error on the IPS or SSCM bus (causing SSCM[CER] assertion) is asserted depending on the selected source. The STCU2 write access is locked until the correct sequence is applied.

The STCU_SKC register is not readable; any attempts to read this register always returns the value 0x00000000.

Offset: 0x0008 Access: User write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W	SKC															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0															
W	SKC															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1363. STCU2 SK Code Register (STCU_SKC)

Table 1311. STCU_SKC field descriptions

Field	Description
0:31 SKC	STCU2 security key code for offline test 0xD3FEA98B = Key1 to unlock the write access the STCU2 0x2C015674 = Key2 to unlock the write access the STCU2 STCU2 security key code for online test 0x753F924E = Key1 to unlock the write access the STCU2 0x8AC06DB1 = Key2 to unlock the write access the STCU2

66.5.4 STCU2 Configuration Register (STCU_CFG)

The STCU_CFG register includes the global configuration of the STCU2 and can be updated both in the offline and online test steps.

Access to this register is described in *Figure 1364*. It also depends on the state of the WRP bit as follows:

- When WRP = 0, the register can be written during the online self-test case without restrictions
- When WRP = 1:
  - The only bit that can be written without any restriction is WRP.
  - If software operations write to other register bits, a transfer error is raised if the value of the selected byte that is written differs from the current status of the register. This function is implemented to prevent potential compilation behavior that might invalidate this single bit clean capability.

Offset: 0x000C Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	PTR(1)							LB_DELAY(1)							
W																
Reset	The reset value is chip-specific; see the chapter that describes how modules are configured and connected.															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	WRP	0	0	CRCEN(1)	PMOSEN(1)	MBU(1)	CLK_CFG(1)		
W																
Reset	The reset value is chip-specific; see the chapter that describes how modules are configured and connected.															

1. You can always read this field. In the offline self-test phase, you can always write to this field. In the online self-test phase, you can write to this field only when WRP = 0.

Figure 1364. STCU2 Configuration Register (STCU_CFG)

Table 1312. STCU_CFG field descriptions

Field	Description
1:7 PTR	LBIST or MBIST pointer. PTR defines the logical pointer to the first LBIST or MBIST to be scheduled when the self-test procedure is enabled. PTR is the entry pointer to a linked list of BIST descriptors. If PTR = 0, then the LBIST0 is initially scheduled. (0x00 to LBIST - 1) – pointer to LBIST (0x10 to 0x10 + MBIST - 1) – pointer to MBIST 0x7F – pointer to NIL. No BIST execution.
8:15 LB_DELAY	Delay LBIST run. LB_DELAY defines the delay between the LBIST starts when more than a single LBIST is selected to be executed concurrently with the purpose of smoothing the power consumption transient. **Note:** if the frequency of the STCU2 core clock is 16 MHz, the delay parameters are expressed in microseconds. 0x00 No delay 0x01 001 * 16 STCU2 Core clock cycles 0x02 002 * 16 STCU2 Core clock cycles 0x03 003 * 16 STCU2 Core clock cycles … 0xFD 253 * 16 STCU2 Core clock cycles 0xFE 254 * 16 STCU2 Core clock cycles 0xFF 255 * 16 STCU2 Core clock cycles **Note:** STCU2 is being clocked from PBRIDGE clock. Please refer to Clocking chapter for PBRIDGE clock frequency value.
23 WRP	Write Protection. 0 Specific STCU2 registers can be written through IPS bus interface after offline self-test sequence has completed. 1 Specific STCU2 registers cannot be written though IPS even after offline self-test sequence has completed, thus preventing any user application write operation.
26 CRCEN	CRC Enable comparison. 0 The CRC comparison is not performed and the STCU_ERR_STAT[CRCS][SW] and STCU_ERR_STAT[UF/RF] global flags are not updated in case of mismatches between STCU2[CRCE] and STCU2[CRCR] values. 1 The CRC comparison is performed and the status is updated in the respective flags of the STCU_ERR_STAT register
27 PMOSEN	MBIST PMOS Test Enable. 0 MBIST PMOS test is not enabled 1 MBIST PMOS test is enabled

Table 1312. STCU_CFG field descriptions (continued)

Field	Description
28 MBU	MBIST MBU Test Enabled. 0 MBIST full algorithm or reduced one without PMOS test (PMOSEN set to 0) is applied. 1 MBIST simplified multi-bit upset algorithm is used to check the RAM. If a ROM is selected, the applied algorithm will be the standard one without PMOS test. This bit overwrites the PMOSEN feature.
29:31 CLK_CFG	Logic, memory BIST and STCU2 core CLK clock configuration. CLK_CFG defines the ratio between the sys_clk and the TCK used to program both the LBIST and the MBIST and the STCU2 core clock. The punch-out mechanism is used to generate the derived clocks. The allowed configurations are: 000 sys_clk 001 sys_clk/2 010 sys_clk/3 011 sys_clk/4 100 sys_clk/5 101 sys_clk/6 110 sys_clk/7 111 sys_clk/8

66.5.5 STCU2 PLL Configuration Register (STCU_PLL_CFG)

The STCU_PLL_CFG register defines the parameters used to program the PLL only when the offline LBISTs or MBISTs are performed and STCU_RUN[MBPLLEN] = 1 or STCU_RUN[LBPLLEN] = 1.

In the online state, these bits are not effective. It is also possible to set the PLL during active online mode to prevent any potential functionality issues.

The R/W fields in this register are readable at any time. However, you can write to these fields only when offline self-test phase is active.

Offset: 0x0010 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	PLLODF						0	0	0	0	0	PLLIDF		
W																
Reset	The reset value is chip-specific; see the chapter that describes how modules are configured and connected.															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	PLLLDF						
W																
Reset	The reset value is chip-specific; see the chapter that describes how modules are configured and connected.															

Figure 1365. STCU2 PLL Configuration Register (STCU_PLL_CFG)

Table 1313. STCU_PLL_CFG field descriptions

Field	Description
2:7 PLLODF	PLL Output Division Factor. The value of this field drives the output division factor of the PLL embedded in the device. Refer to the RFDPHI field of the PLL0 divider register (PLL0DIV) in *Chapter 25: Dual PLL Digital Interface (PLLDIG)* to define the exact mapping between the PLLODF value and the respective PLL's output division factor.
13:15 PLLIDF	PLL Input Division Factor. The value of this field drives the input division factor of the PLL embedded in the device. Refer to the PREDIV field of the PLL0 divider register (PLL0DIV) in *Chapter 25: Dual PLL Digital Interface (PLLDIG)* to define the exact mapping between the PLLIDF value and the respective PLL's input division factor.
25:31 PLLLDF	PLL Loop Division Factor. The value of this field drives the loop division factor of the PLL embedded in the device. Refer to the MFD field of the PLL0 divider register (PLL0DIV) in *Chapter 25: Dual PLL Digital Interface (PLLDIG)* to define the exact mapping between the PLLLDF value and the respective PLL's loop division factor.

66.5.6 STCU2 Watchdog Register Granularity (STCU_WDG)

The STCU_WDG register defines the time budget for LBIST and MBIST execution, providing protection against dead-locks or infinite loop conditions during the self-test procedure. When the offline self-test sequence is not run and the STCU_RUN[BYP] bit is not set, the bits 15…0 define the timeout before the STCU_ERR[WDTO] bit is set and the STCU2 core clock is switched off.

When the offline or online self-test sequence is run, it defines the timeout of the execution run.

The R/W fields in this register are readable at any time. You can write to these fields when either of the following conditions is true:

- Offline self-test phase is active
- Online self-test phase is active and STCU_CFG[WRP] = 0

Offset: 0x0014 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	WDGEOC															
Reset	The reset value is chip-specific; see the chapter that describes how modules are configured and connected.															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	WDGEOC															
Reset	The reset value is chip-specific; see the chapter that describes how modules are configured and connected.															

Figure 1366. STCU2 Watchdog Register Granularity (STCU_WDG)

Table 1314. STCU_WDG field descriptions

Field	Description
0:31 WDGEOC	Watchdog End of Count Timer. This value must be set in order to define the time budget respective to the offline and online self-test execution and check that everything is working correctly in this timeslot. The allowed time delays are: 0x00000000 1 * 16 STCU2 core clock cycles 0x00000001 2 * 16 STCU2 core clock cycles 0x00000002 3 * 16 STCU2 core clock cycles … 0xFFFFFFFD 4294967294 * 16 STCU2 core clock cycles 0xFFFFFFFE 4294967295 * 16 STCU2 core clock cycles 0xFFFFFFFF 4294967296 * 16 STCU2 core clock cycles **Note:** STCU2 is being clocked from PBRIDGE clock. Please refer to Clocking chapter for PBRIDGE clock frequency value.

66.5.7 STCU2 CRC Expected Status Register (STCU_CRCE)

The STCU_CRCE register includes the expected signature of the CRC-32 (ethernet protocol). The CRC-32 bit engine is initialized at STCU_CRCE_RES and computes the CRC signature of the STCU2 critical signals. When STCU_CFG[CRCEN] = 1, it provides the self check capability of the STCU2 managing the CRCS[SW] bits of the STCU_ERR_STAT register. The polynomial CRC used to evaluate the status of this register is the following:

Equation 77 CRC-32 (ethernet protocol)

$$X^{32} + X^{26} + X^{23} + X^{22} + X^{16} + X^{12} + X^{11} + X^{10} + X^8 + X^7 + X^5 + X^4 + X^2 + X + 1$$

The R/W fields in this register are readable at any time. You can write to these fields when either of the following conditions are true:

- Offline self-test phase is active.
- Online self-test phase is active and STCU_CFG[WRP] = 0.

Offset: 0x001C Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	CRCE															
W																
Reset	The reset value is chip-specific; see the chapter that describes how modules are configured and connected.															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	CRCE															
W																
Reset	The reset value is chip-specific; see the chapter that describes how modules are configured and connected.															

Figure 1367. STCU2 CRC Expected Status Register (STCU_CRCE)

Table 1315. STCU_CRCE field descriptions

Field	Description
0:31 CRCE	CRC expected signature

66.5.8 STCU2 CRC Read Status Register (STCU_CRCR)

The STCU_CRCR register reports the value obtained at the end of the offline and online self-test sequence. It may be used for diagnoses and also for additional checking with respect to the CRCS[SW] bit of the STCU_ERR_STAT register.

The content of this register is initialized to its reset value as soon as STCU_RUNSW[RUNSW] = 1.

Offset: 0x0020 Access: User read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	CRCR															
W																
Reset	The reset value is chip-specific; see the chapter that describes how modules are configured and connected.															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	CRCR															
W																
Reset	The reset value is chip-specific; see the chapter that describes how modules are configured and connected.															

Figure 1368. STCU2 CRC Read Status Register (STCU_CRCR)

Table 1316. STCU_CRCR field descriptions

Field	Description
0:31 CRCR	Read CRC signature

66.5.9 STCU2 Error Register (STCU_ERR_STAT)

The STCU_ERR_STAT register includes the status flags respective to the STCU2 internal error conditions that occurred during configuration or the online and offline self-test execution.

The Unrecoverable Faults Status Flag (UFSF) and Recoverable Faults Status Flag (RFSF) can be set and cleared using the FCCU dedicated channels as described in *Section 66.7.4*.

Access to this register is described in *Figure 1369* and as follows:

- If you select the byte write capability to write only the UFSF and RFSF, then there is no restriction in writing these bits.
- If your software performs the write operations on other bits besides UFSF/RFSF, then a transfer error is generated only if the value you are writing to the other bits differs from their value currently stored in the register. This functionality is implemented to prevent potential compilation behavior that might invalidate the UFSF/RFSF single bit set and clean capability.

Offset: 0x0024 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	ABORTHW(1)	0	0	0	0	LOCKESW(1)	WDTOSW(1)	CRCSSW(1)	ENGESW(1)	INVPSW(1)
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	UFSF	RFSF	0	0	0	LOCKE	WDTO	CRCS	ENGE	INVP
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. You can always read this field. The content of this field is initialized to its reset value as soon as STCU_RUNSW[RUNSW] = 1.

Figure 1369. STCU2 Error Register (STCU_ERR_STAT)

Table 1317. STCU_ERR_STAT field descriptions

Field	Description
6 ABORTHW	Online hardware abort flag. 0 No hardware abort was requested during the online self-test sequence 1 A hardware abort was detected during the online self-test sequence
11 LOCKESW	Online LOCK Error. 0 If PLL is enabled, it is correctly locked during the self-test sequence 1 When the PLL is enabled, this flag highlights an unexpected PLL unlock event during the online self-test sequence execution. The online self-test run is stopped and the status of the current running LBISTs or MBISTs is saved in the respective registers. The LOCK signal is monitored during the LBIST run when STCU_RUNSW[LBSWPLLEN] is set and/or during the MBIST run when STCU_RUNSW[MBSWPLLEN] = 1.

Table 1317. STCU_ERR_STAT field descriptions (continued)

Field	Description
12 WDTOSW	Online Watchdog timeout. 0 LBIST and MBIST time slots have completed within the assigned watchdog time. 1 LBIST and MBIST time slot have not completed in the assigned watchdog time or there are internal mismatches in the end of execution signals. The conditions that flag the failures are the following: – MBIST BEND status flags that at least one of the selected MBIST run is not finished – lbist_done of the selected LBIST flags the LBIST run is/are not finished
13 CRCSSW	Online CRC Status. This flag is activated *only* when online self-test is performed and STCU_CFG[CRCEN] is set to '1', otherwise it is always forced to '0'. This prevents the generation of the respective STCU_ERR_STAT[UFSF/RFSF] bits. In both the cases, the content of the STCU_CRCR register reports the current evaluated CRC. 0 Successful CRC comparison or comparison has been masked (STCU_CFG[CRCEN] set to '0') 1 Failed CRC comparison
14 ENGESW	Online Engine Error. 0 Valid Engine execution 1 Invalid Engine execution. The following conditions set this bit: – STCU2 state machines (watchdog, master and loader/shifter) selects an unused code – Serial pass/fail status of the selected MBIST and the global BBAD flag are not aligned – Serial bend status of the selected MBIST and the global BEND flag are not aligned
15 INVPSW	Online Invalid Pointer. 0 Valid linked pointer list 1 Invalid linked pointer list. The following conditions set this bit: – Initial LBIST or MBIST pointer is out of range – LBIST is selected when MBIST is concurrently running or vice versa – Error in LBIST or MBIST linking (execution generates an infinite loop)
22 UFSF	Unrecoverable Faults Status Flag This flag reports the global status of the UF. 0 No errors that trigger the unrecoverable faults condition 1 There are errors that trigger the unrecoverable faults condition
23 RFSF	Recoverable Faults Status Flag 0 No errors that trigger the recoverable faults condition 1 There are errors that trigger the recoverable faults condition
27 LOCKE	Offline LOCK Error 0 The PLL is correctly locked during the self-test sequence 1 When the PLL is enabled, this flag signals an unexpected PLL unlock event during the offline self-test sequence execution. The online self-test run is stopped and the status of the current running LBISTs or MBISTs is saved in the respective registers. The LOCK signal is monitored during the LBIST run when STCU_RUN[LBPLLEN] = 1 and/or during the MBIST run when STCU_RUN[MBPLLEN] = 1.

Table 1317. STCU_ERR_STAT field descriptions (continued)

Field	Description
28 WDTO	Offline Watchdog timeout. 0 LBIST and MBIST time slot have been completed within the assigned watchdog time. 1 LBIST and MBIST time slot have not been completed within the assigned watchdog time or there are internal mismatches among end of execution signals. The conditions that flag the failures are the following: – The STCU_RUN[RUN] bit or the STCU_RUN[BYP] bit are not set before the watchdog reaches the end of count – MBIST BEND status flags that at least one of the selected MBIST run is not finished – lbist_done of the selected LBIST flags the LBIST run is/are not finished
29 CRCS	Offline CRC status. This flag is activated *only* when offline self-test is performed STCU_CFG[CRCEN] is set to '1', otherwise it is always forced to '0'. This prevents the generation of the respective STCU_ERR_STAT[UFSF/RFSF] bits. In both cases, the content of the STCU_CRCR register reports the current evaluated CRC. 0 Successful CRC comparison or comparison has been masked (STCU_CFG[CRCEN] programmed to '0') 1 Failed CRC comparison
30 ENGE	Offline Engine Error. 0 Valid engine execution 1 Invalid engine execution. The following conditions set this bit: – STCU2 state machines (watchdog, master and loader/shifter) selects an unused code – Serial pass/fail status of the selected MBIST and the global BBAD flag are not aligned – Serial bend status of the selected MBIST and the global BEND flag are not aligned
31 INVP	Offline Invalid pointer. 0 Valid linked pointer list 1 Invalid linked pointer list. The following conditions set this bit: – Initial LBIST or MBIST pointer is out of range – LBIST is selected when MBIST is concurrently running or vice versa – Error in LBIST or MBIST linking (execution generates an infinite loop)

66.5.10 STCU2 Error FM Register (STCU_ERR_FM)

The STCU_ERR_FM register defines the fault mapping of the STCU2 faults described in the STCU_ERR_STAT register in terms of unrecoverable or recoverable faults. All sources of internal faults can be routed to UF and RF.

The R/W fields in this register are readable at any time. You can write to these fields when either of the following conditions are true:

- Offline self-test phase is active.
- Online self-test phase is active and STCU_CFG[WRP] = 0.

Offset: 0x0028 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	The reset value is chip-specific; see the chapter that describes how modules are configured and connected.															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	LOCKEUFM	WDTOUFM	CRCSUFM	ENGEUFM	INVPUFM
W																
Reset	The reset value is chip-specific; see the chapter that describes how modules are configured and connected.															

Figure 1370. STCU2 Error FM Register (STCU_ERR_FM)

Table 1318. STCU_ERR_FM field descriptions

Field	Description
27 LOCKEUFM	PLL LOCK Unrecoverable Fault Mapping 0 Recoverable fault mapping 1 Unrecoverable fault mapping
28 WDTOUFM	WatchDog timeout Unrecoverable Fault Mapping 0 Recoverable fault mapping 1 Unrecoverable fault mapping
29 CRCSUFM	CRC Status Unrecoverable Fault Mapping 0 Recoverable fault mapping 1 Unrecoverable fault mapping
30 ENGEUFM	Engine Error Unrecoverable Fault Mapping 0 Recoverable fault mapping 1 Unrecoverable fault mapping
31 INVPUFM	Invalid Pointer Unrecoverable Fault Mapping 0 Recoverable fault mapping 1 Unrecoverable mapping

66.5.11 STCU2 Offline LBIST Status Register (STCU_LBS)

The STCU_LBS register includes the results corresponding to the execution of the selected offline LBIST. The size of the register depends on the number of LBISTs.

Offset: 0x002C Access: User read

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	LBS15	LBS14	LBS13	LBS12	LBS11	LBS10	LBS9	LBS8	LBS7	LBS6	LBS5	LBS4	LBS3	LBS2	LBS1	LBS0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1371. STCU2 Offline LBIST Status Register (STCU_LBS)

Table 1319. STCU_LBS field descriptions

Field	Description
16:31 LBS[(NLBIST-1):0]	Offline status of the selected LBIST 0 Failed LBIST execution 1Successful LBIST execution

66.5.12 STCU2 Offline LBIST End Flag Register (STCU_LBE)

The STCU_LBE register includes the end flag respective to the execution of the selected offline LBIST. The size of the register depends on the number of LBISTs.

Offset: 0x0030 Access: User read

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	LBE15	LBE14	LBE13	LBE12	LBE11	LBE10	LBE9	LBE8	LBE7	LBE6	LBE5	LBE4	LBE3	LBE2	LBE1	LBE0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1372. STCU2 Offline LBIST End Flag Register (STCU_LBE)

Table 1320. STCU_LBE field descriptions

Field	Description
16:31 LBE[(NLBIST-1):0]	Offline End status of the selected LBIST 0 LBIST execution not yet completed 1LBIST execution finished

66.5.13 STCU2 Online LBIST Status Register (STCU_LBSSW)

The STCU_LBSSW register includes the results corresponding to the execution of the selected online LBIST. The size of the register depends on the number of LBISTs. The content of this register is initialized to its reset value as soon as STCU_RUNSW[RUNSW] = 1.

Offset: 0x0034 Access: User read

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	LBSSW15	LBSSW14	LBSSW13	LBSSW12	LBSSW11	LBSSW10	LBSSW9	LBSSW8	LBSSW7	LBSSW6	LBSSW5	LBSSW4	LBSSW3	LBSSW2	LBSSW1	LBSSW0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1373. STCU2 Online LBIST Status Register (STCU_LBSSW)

Table 1321. STCU_LBSSW field descriptions

Field	Description
16:31 LBSSW[(NLBIST-1):0]	Online status of the selected LBIST 0 Failed LBIST execution 1Successful LBIST execution

66.5.14 STCU2 Online LBIST End Flag Register (STCU_LBESW)

The STCU_LBESW register includes the end flag respective to the execution of the selected online LBIST. The size of the register depends on the number of LBIST (NLBIST parameter). The content of this register is initialized to its reset value as soon as STCU_RUNSW[RUNSW] = 1.

Offset: 0x0038 Access: User read

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	LBESW15	LBESW14	LBESW13	LBESW12	LBESW11	LBESW10	LBESW9	LBESW8	LBESW7	LBESW6	LBESW5	LBESW4	LBESW3	LBESW2	LBESW1	LBESW0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1374. STCU2 Online LBIST End Flag Register (STCU_LBESW)

Table 1322. STCU_LBESW field descriptions

Field	Description
16:31 LBESW[(NLBIST-1):0]	Online LBIST End status. 0 LBIST execution not yet completed 1LBIST execution finished

66.5.15 STCU2 LBIST Unrecoverable FM Register (STCU_LBUFM)

The STCU_LBUFM register defines the fault mapping of each LBIST in terms of unrecoverable or recoverable fault. The size of the register depends on the number of LBIST.

The R/W fields in this register are readable at any time. You can write to these fields when either of the following conditions is true:

- Offline self-test phase is active.
- Online self-test phase is active and STCU_CFG[WRP] = 0.

Offset: 0x0040 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	The reset value is chip-specific; see the chapter that describes how modules are configured and connected.															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	LBUFM15	LBUFM14	LBUFM13	LBUFM12	LBUFM11	LBUFM10	LBUFM9	LBUFM8	LBUFM7	LBUFM6	LBUFM5	LBUFM4	LBUFM3	LBUFM2	LBUFM1	LBUFM0
W																
Reset	The reset value is chip-specific; see the chapter that describes how modules are configured and connected.															

Figure 1375. STCU2 LBIST Unrecoverable FM Register (STCU_LBUFM)

Table 1323. STCU_LBUFM field descriptions

Field	Description
16:31 LBUFM[(NLBIST-1):0]	LBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping

66.5.16 STCU2 Offline MBIST Status Low Register (STCU_MBSL)

The STCU_MBSL register includes the results corresponding to the execution of the selected offline MBIST in the range NMCUT = 0…31. The size of the register depends on the amount of RAM or ROM subject to BIST.

Offset: 0x0044 Access: User read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	MBS31	MBS30	MBS29	MBS28	MBS27	MBS26	MBS25	MBS24	MBS23	MBS22	MBS21	MBS20	MBS19	MBS18	MBS17	MBS16
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	MBS15	MBS14	MBS13	MBS12	MBS11	MBS10	MBS9	MBS8	MBS7	MBS6	MBS5	MBS4	MBS3	MBS2	MBS1	MBS0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1376. STCU2 Offline MBIST Status Low Register (STCU_MBSL)

Table 1324. STCU_MBSL field descriptions

Field	Description
0:31 MBS[(NMCUT-1):0]	Offline status (NMCUT range = 0 to 31) of the selected MBIST **Note:** This bit is meaningful when the respective bit of the STCU_MBEL register reports the MBIST is finished. 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution

66.5.17 STCU2 Offline MBIST Status Medium Register (STCU_MBSM)

The STCU_MBSM register includes the results corresponding to the execution of the selected offline MBIST in the range NMCUT = 32…63. The size of the register depends on the amount of RAM or ROM subject to BIST.

Offset: 0x0048 Access: User read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	MBS63	MBS62	MBS61	MBS60	MBS59	MBS58	MBS57	MBS56	MBS55	MBS54	MBS53	MBS52	MBS51	MBS50	MBS49	MBS48
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	MBS47	MBS46	MBS45	MBS44	MBS43	MBS42	MBS41	MBS40	MBS39	MBS38	MBS37	MBS36	MBS35	MBS34	MBS33	MBS32
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1377. STCU2 Offline MBIST Status Medium Register (STCU_MBSM)

Table 1325. STCU_MBSM field descriptions

Field	Description
0:31 MBS[(NMCUT-1):32]	Offline status (NMCUT range = 32 to 63) of the selected MBIST **Note:** This bit is meaningful when the respective bit of the STCU_MBEL register reports the MBIST is finished). 0 Failed NMCUT BIST execution 1 No fault detected during the NMCUT BIST execution

66.5.18 STCU2 Offline MBIST Status High Register (STCU_MBSH)

The STCU_MBSH register includes the results corresponding to the execution of the selected offline MBIST in the range NMCUT = 64…95. The size of the register depends on the amount of RAM or ROM subject to BIST.

Offset: 0x004C Access: User read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	MBS95	MBS94	MBS93	MBS92	MBS91	MBS90	MBS89	MBS88	MBS87	MBS86	MBS85	MBS84	MBS83	MBS82	MBS81	MBS80
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	MBS79	MBS78	MBS77	MBS76	MBS75	MBS74	MBS73	MBS72	MBS71	MBS70	MBS69	MBS68	MBS67	MBS66	MBS65	MBS64
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1378. STCU2 Offline MBIST Status High Register (STCU_MBSH)

Table 1326. STCU_MBSH field descriptions

Field	Description
0:31 MBS[(NMCUT-1):64]	Offline status (NMCUT range = 64 to 95) of the selected MBIST. **Note:** This bit is meaningful when the respective bit of the STCU_MBEL register reports the MBIST is finished). 0 Failed NMCUT BIST execution 1 No fault detected during the NMCUT BIST execution

66.5.19 STCU2 Offline MBIST End Flag Low Register (STCU_MBEL)

The STCU_MBEL register includes the end flag respective to the execution of the selected offline MBIST in the range NMCUT = 0…31. The size of the register depends on the amount of RAM or ROM subject to BIST.

Offset: 0x0050 Access: User read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	MBE31	MBE30	MBE29	MBE28	MBE27	MBE26	MBE25	MBE24	MBE23	MBE22	MBE21	MBE20	MBE19	MBE18	MBE17	MBE16
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	MBE15	MBE14	MBE13	MBE12	MBE11	MBE10	MBE9	MBE8	MBE7	MBE6	MBE5	MBE4	MBE3	MBE2	MBE1	MBE0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1379. STCU2 Offline MBIST End Flag Low Register (STCU_MBEL)

Table 1327. STCU_MBEL field descriptions

Field	Description
0:31 MBE[(NMCUT-1):0]	Offline End status (0 to 31) of the selected MBIST. 0 MBIST execution still ongoing 1 MBIST execution finished

66.5.20 STCU2 Offline MBIST End Flag Medium Register (STCU_MBEM)

The STCU_MBEM register includes the end flag respective to the execution of the selected offline MBIST in the range NMCUT = 32…63. The size of the register depends on the amount of RAM or ROM subject to BIST.

Offset: 0x0054 Access: User read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	MBE63	MBE62	MBE61	MBE60	MBE59	MBE58	MBE57	MBE56	MBE55	MBE54	MBE53	MBE52	MBE51	MBE50	MBE49	MBE48
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	MBE47	MBE46	MBE45	MBE44	MBE43	MBE42	MBE41	MBE40	MBE39	MBE38	MBE37	MBE36	MBE35	MBE34	MBE33	MBE32
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1380. STCU2 Offline MBIST End Flag Medium Register (STCU_MBEM)

Table 1328. STCU_MBEM field descriptions

Field	Description
0:31 MBE[(NMCUT-1):32]	Offline End status (32 to 63) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished

66.5.21 STCU2 Offline MBIST End Flag High Register (STCU_MBEH)

The STCU_MBEH register includes the end flag respective to the execution of the selected offline MBIST in the range NMCUT = 64…95. The size of the register depends on the amount of RAM or ROM subject to BIST.

Offset: 0x0058 Access: User read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	MBE95	MBE94	MBE93	MBE92	MBE91	MBE90	MBE89	MBE88	MBE87	MBE86	MBE85	MBE84	MBE83	MBE82	MBE81	MBE80
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	MBE79	MBE78	MBE77	MBE76	MBE75	MBE74	MBE73	MBE72	MBE71	MBE70	MBE69	MBE68	MBE67	MBE66	MBE65	MBE64
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1381. STCU2 Offline MBIST End Flag High Register (STCU_MBEH)

Table 1329. STCU_MBEH field descriptions

Field	Description
0:31 MBE[(NMCUT-1):64]	Offline End status (64 to 95) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished

66.5.22 STCU2 Online MBIST Status Low Register (STCU_MBSLSW)

The STCU_MBSLSW register includes the results corresponding to the execution of the selected online MBIST in the range NMCUT = 0…31. The size of the register depends on the amount of RAM or ROM subject to BIST. The content of this register is initialized to its reset value as soon as STCU_RUNSW[RUNSW] = 1.

Offset: 0x005C Access: User read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	MBSSW31	MBSSW30	MBSSW29	MBSSW28	MBSSW27	MBSSW26	MBSSW25	MBSSW24	MBSSW23	MBSSW22	MBSSW21	MBSSW20	MBSSW19	MBSSW18	MBSSW17	MBSSW16
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	MBSSW15	MBSSW14	MBSSW13	MBSSW12	MBSSW11	MBSSW10	MBSSW9	MBSSW8	MBSSW7	MBSSW6	MBSSW5	MBSSW4	MBSSW3	MBSSW2	MBSSW1	MBSSW0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1382. STCU2 Online MBIST Status Low Register (STCU_MBSLSW)

Table 1330. STCU_MBSLSW field descriptions

Field	Description
0:31 MBSSW[(NMCUT-1):0]	Online status (NMCUT range = 0 to 31) of the selected MBIST 0 Failed NMCUT BIST execution 1 No fault detected during the NMCUT BIST execution

66.5.23 STCU2 Online MBIST Status Medium Register (STCU_MBSMSW)

The STCU_MBSMSW register includes the results corresponding to the execution of the selected online MBIST in the range NMCUT = 32…63. The size of the register depends on the amount of RAM or ROM subject to BIST. The content of this register is initialized to its reset value as soon as STCU_RUNSW[RUNSW] = 1.

Offset: 0x0060 Access: User read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	MBSSW63	MBSSW62	MBSSW61	MBSSW60	MBSSW59	MBSSW58	MBSSW57	MBSSW56	MBSSW55	MBSSW54	MBSSW53	MBSSW52	MBSSW51	MBSSW50	MBSSW49	MBSSW48
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	MBSSW47	MBSSW46	MBSSW45	MBSSW44	MBSSW43	MBSSW42	MBSSW41	MBSSW40	MBSSW39	MBSSW38	MBSSW37	MBSSW36	MBSSW35	MBSSW34	MBSSW33	MBSSW32
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1383. STCU2 Online MBIST Status Medium Register (STCU_MBSMSW)

Table 1331. STCU_MBSMSW field descriptions

Field	Description
0:31 MBSSW[(NMCUT-1):32]	Online status (NMCUT range = 32 to 63) of the selected MBIST. 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution

66.5.24 STCU2 Online MBIST Status High Register (STCU_MBSHSW)

The STCU_MBSHSW register includes the results corresponding to the execution of the selected online MBIST in the range NMCUT = 64…95. The size of the register depends on the amount of RAM or ROM subject to BIST. The content of this register is initialized to its reset value as soon as STCU_RUNSW[RUNSW] = 1.

Offset: 0x0064 Access: User read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	MBSSW95	MBSSW94	MBSSW93	MBSSW92	MBSSW91	MBSSW90	MBSSW89	MBSSW88	MBSSW87	MBSSW86	MBSSW85	MBSSW84	MBSSW83	MBSSW82	MBSSW81	MBSSW80
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	MBSSW79	MBSSW78	MBSSW77	MBSSW76	MBSSW75	MBSSW74	MBSSW73	MBSSW72	MBSSW71	MBSSW70	MBSSW69	MBSSW68	MBSSW67	MBSSW66	MBSSW65	MBSSW64
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1384. STCU2 Online MBIST Status High Register (STCU_MBSHSW)

Table 1332. STCU_MBSHSW field descriptions

Field	Description
0:31 MBSSW[(NMCUT-1):64]	Online status (NMCUT range = 64 to 95) of the selected MBIST. 0 Failed NMCUT BIST execution 1 No fault detected during the NMCUT BIST execution

66.5.25 STCU2 Online MBIST End Flag Low Register (STCU_MBELSW)

The STCU_MBELSW register includes the End Flag respective to the execution of the selected online MBIST in the range NMCUT = 0…31. The size of the register depends on the amount of RAM or ROM subject to BIST. The content of this register is initialized to its reset value as soon as STCU_RUNSW[RUNSW] = 1.

Offset: 0x0068 Access: User read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	MBESW31	MBESW30	MBESW29	MBESW28	MBESW27	MBESW26	MBESW25	MBESW24	MBESW23	MBESW22	MBESW21	MBESW20	MBESW19	MBESW18	MBESW17	MBESW16
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	MBESW15	MBESW14	MBESW13	MBESW12	MBESW11	MBESW10	MBESW9	MBESW8	MBESW7	MBESW6	MBESW5	MBESW4	MBESW3	MBESW2	MBESW1	MBESW0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1385. STCU2 Online MBIST End Flag Low Register (STCU_MBELSW)

Table 1333. STCU_MBELSW field descriptions

Field	Description
0:31 MBESW[(NMCUT-1):0]	Online End status (0 to 31) of the selected MBIST. 0 MBIST execution still ongoing 1 MBIST execution finished

66.5.26 STCU2 Online MBIST End Flag Medium Register (STCU_MBEMSW)

The STCU_MBEMSW register includes the end flag respective to the execution of the selected online MBIST in the range NMCUT = 32…63. The size of the register depends on the amount of RAM or ROM subject to BIST. The content of this register is initialized to its reset value as soon as STCU_RUNSW[RUNSW] = 1.

Offset: 0x006C Access: User read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	MBESW63	MBESW62	MBESW61	MBESW60	MBESW59	MBESW58	MBESW57	MBESW56	MBESW55	MBESW54	MBESW53	MBESW52	MBESW51	MBESW50	MBESW49	MBESW48
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	MBESW47	MBESW46	MBESW45	MBESW44	MBESW43	MBESW42	MBESW41	MBESW40	MBESW39	MBESW38	MBESW37	MBESW36	MBESW35	MBESW34	MBESW33	MBESW32
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1386. STCU2 Online MBIST End Flag Medium Register (STCU_MBEMSW)

Table 1334. STCU_MBEMSW field descriptions

Field	Description
0:31 MBESW[(NMCUT-1):32]	Online End status (32 to 63) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished

66.5.27 STCU2 Online MBIST End Flag High Register (STCU_MBEHSW)

The STCU_MBEHSW register includes the end flag respective to the execution of the selected online MBIST in the range NMCUT = 64…95. The size of the register depends on the amount of RAM or ROM subject to BIST. The content of this register is initialized to its reset value as soon as STCU_RUNSW[RUNSW] = 1.

Offset: 0x0070 Access: User read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	MBESW95	MBESW94	MBESW93	MBESW92	MBESW91	MBESW90	MBESW89	MBESW88	MBESW87	MBESW86	MBESW85	MBESW84	MBESW83	MBESW82	MBESW81	MBESW80
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	MBESW79	MBESW78	MBESW77	MBESW76	MBESW75	MBESW74	MBESW73	MBESW72	MBESW71	MBESW70	MBESW69	MBESW68	MBESW67	MBESW66	MBESW65	MBESW64
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1387. STCU2 Online MBIST End Flag High Register (STCU_MBEHSW)

Table 1335. STCU_MBEHSW field descriptions

Field	Description
0:31 MBESW[(NMCUT-1):64]	Online End status (64 to 95) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished

66.5.28 STCU2 MBIST Unrecoverable FM Low Register (STCU_MBUFML)

The STCU_MBUFML register defines the fault mapping, in terms of unrecoverable or recoverable fault, of the MBIST in the range NMCUT = 0…31. The size of the register depends on the amount of RAM or ROM subject to BIST.

The R/W fields in this register are readable at any time. You can write to these fields when either of the following conditions is true:

- Offline self-test phase is active.
- Online self-test phase is active and STCU_CFG[WRP] = 0.

Offset: 0x0074 Access: User read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	MBUFM31	MBUFM30	MBUFM29	MBUFM28	MBUFM27	MBUFM26	MBUFM25	MBUFM24	MBUFM23	MBUFM22	MBUFM21	MBUFM20	MBUFM19	MBUFM18	MBUFM17	MBUFM16
W																
Reset	The reset value is chip-specific; see the chapter that describes how modules are configured and connected.															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	MBUFM15	MBUFM14	MBUFM13	MBUFM12	MBUFM11	MBUFM10	MBUFM9	MBUFM8	MBUFM7	MBUFM6	MBUFM5	MBUFM4	MBUFM3	MBUFM2	MBUFM1	MBUFM0
W																
Reset	The reset value is chip-specific; see the chapter that describes how modules are configured and connected.															

Figure 1388. STCU2 MBIST Unrecoverable FM Low Register (STCU_MBUFML)

Table 1336. STCU_MBUFML field descriptions

Field	Description
0:31 MBUFM[(NMCUT-1):0]	MBIST Unrecoverable Fault Mapping. 0 Recoverable fault mapping 1 Unrecoverable fault mapping

66.5.29 STCU2 MBIST Unrecoverable FM Medium Register (STCU_MBUFMM)

The STCU_MBUFMM register defines the fault mapping, in terms of unrecoverable or recoverable fault, of the MBIST in the range NMCUT = 32…63. The size of the register depends on the amount of RAM or ROM subject to BIST.

The R/W fields in this register are readable at any time. You can write to these fields when either of the following conditions is true:

- Offline self-test phase is active.
- Online self-test phase is active and STCU_CFG[WRP] = 0.

Offset: 0x0078 Access: User read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	MBUFM63	MBUFM62	MBUFM61	MBUFM60	MBUFM59	MBUFM58	MBUFM57	MBUFM56	MBUFM55	MBUFM54	MBUFM53	MBUFM52	MBUFM51	MBUFM50	MBUFM49	MBUFM48
W																
Reset	The reset value is chip-specific; see the chapter that describes how modules are configured and connected.															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	MBUFM47	MBUFM46	MBUFM45	MBUFM44	MBUFM43	MBUFM42	MBUFM41	MBUFM40	MBUFM39	MBUFM38	MBUFM37	MBUFM36	MBUFM35	MBUFM34	MBUFM33	MBUFM32
W																
Reset	The reset value is chip-specific; see the chapter that describes how modules are configured and connected.															

Figure 1389. STCU2 MBIST Unrecoverable FM Medium Register (STCU_MBUFMM)

Table 1337. STCU_MBUFMM field descriptions

Field	Description
0:31 MBUFM[(NMCUT-1):32]	MBIST Unrecoverable Fault Mapping 0 Recoverable fault mapping 1 Unrecoverable fault mapping

66.5.30 STCU2 MBIST Unrecoverable FM High Register (STCU_MBUFMH)

The STCU_MBUFMH register defines the fault mapping, in terms of unrecoverable or recoverable fault, of the MBIST in the range NMCUT = 64…95. The size of the register depends on the amount of RAM or ROM subject to BIST.

The R/W fields in this register are readable at any time. You can write to these fields when either of the following conditions is true:

- Offline self-test phase is active.
- Online self-test phase is active and STCU_CFG[WRP] = 0.

Offset: 0x007C Access: User read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	MBUFM95	MBUFM94	MBUFM93	MBUFM92	MBUFM91	MBUFM90	MBUFM89	MBUFM88	MBUFM87	MBUFM86	MBUFM85	MBUFM84	MBUFM83	MBUFM82	MBUFM81	MBUFM80
W																
Reset	The reset value is chip-specific; see the chapter that describes how modules are configured and connected.															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	MBUFM79	MBUFM78	MBUFM77	MBUFM76	MBUFM75	MBUFM74	MBUFM73	MBUFM72	MBUFM71	MBUFM70	MBUFM69	MBUFM68	MBUFM67	MBUFM66	MBUFM65	MBUFM64
W																
Reset	The reset value is chip-specific; see the chapter that describes how modules are configured and connected.															

Figure 1390. STCU2 MBIST Unrecoverable FM High Register (STCU_MBUFMH)

Table 1338. STCU_MBUFMH field descriptions

Field	Description
0:31 MBUFM[(NMCUT-1):64]	MBIST Unrecoverable Fault Mapping 0 Recoverable fault mapping 1 Unrecoverable fault mapping

66.5.31 STCU2 LBIST Control *n* Register (STCU_LB_CTRL*n*)

The STCU_LB_CTRL register defines the control setting of each LBIST controller.

The R/W fields in this register are readable at any time. You can write to these fields when either of the following conditions is true:

- Offline self-test phase is active.
- Online self-test phase is active and STCU_CFG[WRP] = 0.

Offset: 0x0100 + n * 0x40 (n = 0 to 3) Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	CSM	PTR							0	0	0	0	PRPGEN	SHS		
W																
Reset	The reset value is chip-specific; see the chapter that describes how modules are configured and connected.															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	SCEN_OFF				SCEN_ON				0	0	0	0	PFT	CWS		
W																
Reset	The reset value is chip-specific; see the chapter that describes how modules are configured and connected.															

Figure 1391. STCU2 LBIST Control *n* Register (STCU_LB_CTRL*n*)

Table 1339. STCU_LB_CTRL*n* field descriptions

Field	Description
0 CSM	Concurrent/sequential mode 0 Sequential mode 1 Concurrent mode
1:7 PTR	Next LBIST or MBIST pointer PTR defines the logical pointer to the next LBIST or MBIST to be scheduled. The next LBIST or MBIST is scheduled concurrently to the current one if the CSM bit is set to '1', otherwise it is scheduled sequentially to the completion of the current LBIST execution. In case of NIL pointer, the CSM bit has to be set sequential (0) to define this is the end of the list. The self-test procedure is stopped when the current LBIST has been completed. (0x00 to LBIST - 1) — pointer to LBIST (0x10 to 0x10 + MBIST - 1) — pointer to MBIST 0x7F — pointer to NIL. No BIST execution. others — invalid pointer, an error is set in the STCU_ERR register.
12 PRPGEN	Enable PRPG Loading 0 Default LBIST value of the PRPG is used during LBIST run 1 LBIST PRPG value is initialized with the STCU_LB_PRPGL/H
13:15 SHS	Shift speed 000 Shift at full rate (bist_clk) 001 Shift at 1/2 rate (bist_clk) 010 Shift at 1/3 rate (bist_clk) 011 Shift at 1/4 rate (bist_clk) 100 Shift at 1/5 rate (bist_clk) 101 Shift at 1/6 rate (bist_clk) 110 Shift at 1/7 rate (bist_clk) 111 Shift at 1/8 rate (bist_clk)

Table 1339. STCU_LB_CTRL*n* field descriptions (continued)

Field	Description
16:19 SCEN_OFF	Scan enable OFF SCEN_OFF defines the number of clock cycles OFF following the falling transition on the SCEN 0000 0 delay cycle 0001 1 delay cycle … 1111 15 delay cycles
20:23 SCEN_ON	Scan enable ON SCEN_ON defines the number of clock cycles OFF following the rising transition on the SCEN 0000 0 delay cycle 0001 1 delay cycle … 1111 15 delay cycles
28 PFT	Past Flush Test The LBIST controller, by default apply 32 Flush Test patterns. This bit allows to skip/apply the Flush test pattern sequence. 0 Apply flush test patterns 1 Skip flush test patterns
29:31 CWS	Capture window size 000 controller waits 16 shift cycles for capture to finish 001 controller waits 1 shift cycle for capture to finish 010 controller waits 2 shift cycles for capture to finish 011 controller waits 3 shift cycles for capture to finish 100 controller waits 4 shift cycles for capture to finish 101 controller waits 5 shift cycles for capture to finish 110 controller waits 6 shift cycles for capture to finish 111 controller waits 7 shift cycles for capture to finish

66.5.32 STCU2 LBIST PC Stop *n* Register (STCU_LB_PCS*n*)

The STCU_LB_PCS register defines the pattern counter stop of each LBIST controller.

The R/W fields in this register are readable at any time. You can write to these fields when either of the following conditions is true:

- Offline self-test phase is active.
- Online self-test phase is active and STCU_CFG[WRP] = 0.

Offset: 0x0104 + n * 0x40 (n = 0 to 3) Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	PCS									
W																
Reset	The reset value is chip-specific; see the chapter that describes how modules are configured and connected.															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	PCS															
W																
Reset	The reset value is chip-specific; see the chapter that describes how modules are configured and connected.															

Figure 1392. STCU2 LBIST PC Stop *n* Register (STCU_LB_PCS*n*)

Table 1340. STCU_LB_PCS*n* field descriptions

Field	Description
6:31 PCS	Pattern counter stop PCS defines the pattern counter stop value.

66.5.33 STCU2 LBIST PRPG Low *n* Register (STCU_LB_PRPGL*n*)

The STCU_LB_PRPGL register defines the LSB part (31…0) of the PRPG start value of the LBIST controller. The size of the register depends on the number of PRPG bits of the respective LBIST.

The R/W fields in this register are readable at any time. You can write to these fields when either of the following conditions is true:

- Offline self-test phase is active.
- Online self-test phase is active and STCU_CFG[WRP] = 0.

Offset: 0x0108 + n * 0x40 (n = 0 to 3) Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	PRPG31	PRPG30	PRPG29	PRPG28	PRPG27	PRPG26	PRPG25	PRPG24	PRPG23	PRPG22	PRPG21	PRPG20	PRPG19	PRPG18	PRPG17	PRPG16
W																
Reset	The reset value is chip-specific; see the chapter that describes how modules are configured and connected.															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	PRPG15	PRPG14	PRPG13	PRPG12	PRPG11	PRPG10	PRPG9	PRPG8	PRPG7	PRPG6	PRPG5	PRPG4	PRPG3	PRPG2	PRPG1	PRPG0
W																
Reset	The reset value is chip-specific; see the chapter that describes how modules are configured and connected.															

Figure 1393. STCU2 LBIST PRPG Low *n* Register (STCU_LB_PRPGL*n*)

Table 1341. STCU_LB_PRPGL*n* field descriptions

Field	Description
0:31 PRPG[31:0]	PRPG LBIST low bits This field defines the low part (31…0) of the LBIST PRPG start value.

66.5.34 STCU2 LBIST PRPG High *n* Register (STCU_LB_PRPGH*n*)

The STCU_LB_PRPGH register defines the MSB part (63…32) of the PRPG start value of the LBIST controller. The size of the register depends on the number of PRPG bits of the respective LBIST.

The R/W fields in this register are readable at any time. You can write to these fields when either of the following conditions is true:

- Offline self-test phase is active.
- Online self-test phase is active and STCU_CFG[WRP] = 0.

Offset: 0x010C + n * 0x40 (n = 0 to 3) Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	PRPG63	PRPG62	PRPG61	PRPG60	PRPG59	PRPG58	PRPG57	PRPG56	PRPG55	PRPG54	PRPG53	PRPG52	PRPG51	PRPG50	PRPG49	PRPG48
Reset	The reset value is chip-specific; see the chapter that describes how modules are configured and connected.															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	PRPG47	PRPG46	PRPG45	PRPG44	PRPG43	PRPG42	PRPG41	PRPG40	PRPG39	PRPG38	PRPG37	PRPG36	PRPG35	PRPG34	PRPG33	PRPG32
Reset	The reset value is chip-specific; see the chapter that describes how modules are configured and connected.															

Figure 1394. STCU2 LBIST PRPG High *n* Register (STCU_LB_PRPGH*n*)

Table 1342. STCU_LB_PRPGH*n* field descriptions

Field	Description
0:31 PRPG[63:32]	PRPG LBIST high bits This field is the high part (63…32) of the LBIST PRPG start value.

66.5.35 STCU2 Offline LBIST MISR Expected Low *n* Register (STCU_LB_MISREL*n*)

The STCU_LB_MISREL register defines the LSB part (31…0) of the expected MISR of the offline LBIST controller. The size of the register depends on the number of MISR bits of the respective LBIST.

The R/W fields in this register are readable at any time. However, you can write to these fields only when offline self-test phase is still active.

Offset: 0x0110 + n * 0x40 (n = 0 to 3) Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	MISRE31	MISRE30	MISRE29	MISRE28	MISRE27	MISRE26	MISRE25	MISRE24	MISRE23	MISRE22	MISRE21	MISRE20	MISRE19	MISRE18	MISRE17	MISRE16
W																
Reset	The reset value is chip-specific; see the chapter that describes how modules are configured and connected.															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	MISRE15	MISRE14	MISRE13	MISRE12	MISRE11	MISRE10	MISRE9	MISRE8	MISRE7	MISRE6	MISRE5	MISRE4	MISRE3	MISRE2	MISRE1	MISRE0
W																
Reset	The reset value is chip-specific; see the chapter that describes how modules are configured and connected.															

Figure 1395. STCU2 Offline LBIST MISR Expected Low *n* Register (STCU_LB_MISREL*n*)

Table 1343. STCU_LB_MISREL*n* field descriptions

Field	Description
0:31 MISRE[31:0]	Offline MISR Expected low bits. This field defines the low part (31…0) of the expected MISR.

66.5.36 STCU2 Offline LBIST MISR Expected High *n* Register (STCU_LB_MISREH*n*)

The STCU_LB_MISREH register defines the MSB part (63…32) of the expected MISR of the offline LBIST controller. The size of the register depends on the number of MISR bits of the respective LBIST.

The R/W fields in this register are readable at any time. However, you can write to these fields only when offline self-test phase is still active.

Offset: 0x0114 + n * 0x40 (n = 0 to 3) Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	MISRE63	MISRE62	MISRE61	MISRE60	MISRE59	MISRE58	MISRE57	MISRE56	MISRE55	MISRE54	MISRE53	MISRE52	MISRE51	MISRE50	MISRE49	MISRE48
W																
Reset	The reset value is chip-specific; see the chapter that describes how modules are configured and connected.															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	MISRE47	MISRE46	MISRE45	MISRE44	MISRE43	MISRE42	MISRE41	MISRE40	MISRE39	MISRE38	MISRE37	MISRE36	MISRE35	MISRE34	MISRE33	MISRE32
W																
Reset	The reset value is chip-specific; see the chapter that describes how modules are configured and connected.															

Figure 1396. STCU2 LBIST MISR Expected High *n* Register (STCU_LB_MISREH*n*)

Table 1344. STCU_LB_MISREH*n* field descriptions

Field	Description
0:31 MISRE[63:32]	Offline MISR Expected high bits This field defines the high part (63…32) of the expected MISR.

66.5.37 STCU2 Offline LBIST MISR Read Low *n* Register (STCU_LB_MISRRL*n*)

The STCU_LB_MISRRL register report the LSB part (31…0) of the MISR obtained at the end of the offline LBIST controller execution. The size of the register depends on the number of MISR bits of the respective LBIST.

Offset: 0x0118 + n * 0x40 (n = 0 to 3) Access: User read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	MISRR31	MISRR30	MISRR29	MISRR28	MISRR27	MISRR26	MISRR25	MISRR24	MISRR23	MISRR22	MISRR21	MISRR20	MISRR19	MISRR18	MISRR17	MISRR16
W																
Reset	The reset value is chip-specific; see the chapter that describes how modules are configured and connected.															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	MISRR15	MISRR14	MISRR13	MISRR12	MISRR11	MISRR10	MISRR9	MISRR8	MISRR7	MISRR6	MISRR5	MISRR4	MISRR3	MISRR2	MISRR1	MISRR0
W																
Reset	The reset value is chip-specific; see the chapter that describes how modules are configured and connected.															

Figure 1397. STCU2 Offline LBIST MISR Read Low *n* Register (STCU_LB_MISRRL*n*)

Table 1345. STCU_LB_MISRRL*n* field descriptions

Field	Description
0:31 MISRR[31:0]	Offline MISR Read low bits. This field is equivalent to the low bits (31…0) of the MISR obtained at the end of the offline LBIST execution

66.5.38 STCU2 Offline LBIST MISR Read High *n* Register (STCU_LB_MISRRH*n*)

The STCU_LB_MISRRH register report the MSB part (63…32) of the MISR obtained at the end of each offline LBIST controller execution. The size of the register depends on the number of MISR bits of the respective LBIST.

Offset: 0x011C + n * 0x40 (n = 0 to 3) Access: User read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	MISRR63	MISRR62	MISRR61	MISRR60	MISRR59	MISRR58	MISRR57	MISRR56	MISRR55	MISRR54	MISRR53	MISRR52	MISRR51	MISRR50	MISRR49	MISRR48
W																
Reset	The reset value is chip-specific; see the chapter that describes how modules are configured and connected.															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	MISRR47	MISRR46	MISRR45	MISRR44	MISRR43	MISRR42	MISRR41	MISRR40	MISRR39	MISRR38	MISRR37	MISRR36	MISRR35	MISRR34	MISRR33	MISRR32
W																
Reset	The reset value is chip-specific; see the chapter that describes how modules are configured and connected.															

Figure 1398. STCU2 Offline LBIST MISR Read High *n* Register (STCU_LB_MISRRH*n*)

Table 1346. STCU_LB_MISRRH*n* field descriptions

Field	Description
0:31 MISRR[63:32]	Offline MISR Read high bits. This field is equivalent to the high bits (63…32) of the MISR obtained at the end of the offline LBIST execution.

66.5.39 STCU2 Online LBIST MISR Expected Low *n* Register (STCU_LB_MISRELSW*n*)

The STCU_LB_MISRELSW register defines the LSB part (31…0) of the expected MISR of the online LBIST controller. The size of the register depends on the number of MISR bits of the respective LBIST.

The R/W fields in this register are readable at any time. You can write to these fields when either of the following conditions is true:

- Offline self-test phase is active.
- Online self-test phase is active and STCU_CFG[WRP] = 0.

Offset: 0x0120 + n * 0x40 (n = 0 to 3) Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	MISRESW31	MISRESW30	MISRESW29	MISRESW28	MISRESW27	MISRESW26	MISRESW25	MISRESW24	MISRESW23	MISRESW22	MISRESW21	MISRESW20	MISRESW19	MISRESW18	MISRESW17	MISRESW16
W																
Reset	The reset value is chip-specific; see the chapter that describes how modules are configured and connected.															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	MISRESW15	MISRESW14	MISRESW13	MISRESW12	MISRESW11	MISRESW10	MISRESW9	MISRESW8	MISRESW7	MISRESW6	MISRESW5	MISRESW4	MISRESW3	MISRESW2	MISRESW1	MISRESW0
W																
Reset	The reset value is chip-specific; see the chapter that describes how modules are configured and connected.															

Figure 1399. STCU2 Online LBIST MISR Expected Low *n* Register (STCU_LB_MISRELSW*n*)

Table 1347. STCU_LB_MISRELSW*n* field descriptions

Field	Description
0:31 MISRESW[31:0]	Online MISR expected low bits. This field defines the low part (31…0) of the expected MISR.

66.5.40 STCU2 Online LBIST MISR Expected High *n* Register (STCU_LB_MISREHSW*n*)

The STCU_LB_MISREHSW register defines the MSB part (63…32) of the expected MISR of the online LBIST controller. The size of the register depends on the number of MISR bits of the respective LBIST.

The R/W fields in this register are readable at any time. You can write to these fields when either of the following conditions is true:

- Offline self-test phase is active.
- Online self-test phase is active and STCU_CFG[WRP] = 0.

Offset: 0x0124 + n * 0x40 (n = 0 to 3) Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	MISRESW63	MISRESW62	MISRESW61	MISRESW60	MISRESW59	MISRESW58	MISRESW57	MISRESW56	MISRESW55	MISRESW54	MISRESW53	MISRESW52	MISRESW51	MISRESW50	MISRESW49	MISRESW48
W																
Reset	The reset value is chip-specific; see the chapter that describes how modules are configured and connected.															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	MISRESW47	MISRESW46	MISRESW45	MISRESW44	MISRESW43	MISRESW42	MISRESW41	MISRESW40	MISRESW39	MISRESW38	MISRESW37	MISRESW36	MISRESW35	MISRESW34	MISRESW33	MISRESW32
W																
Reset	The reset value is chip-specific; see the chapter that describes how modules are configured and connected.															

Figure 1400. STCU2 Online LBIST MISR Expected High *n* Register (STCU_LB_MISREHSW*n*)

Table 1348. STCU_LB_MISREHSW*n* field descriptions

Field	Description
0:31 MISRESW[63:32]	Online MISR expected high bits. This field defines the high part (63…32) of the expected MISR.

66.5.41 STCU2 Online LBIST MISR Read Low *n* Register (STCU_LB_MISRRLSW*n*)

The STCU_LB_MISRRLSW register report the LSB part (31…0) of the MISR obtained at the end of the online LBIST controller execution. The size of the register depends on the number of MISR bits of the respective LBIST.

The content of this register is initialized to its reset value as soon as STCU_RUNSW[RUNSW] = 1.

Offset: 0x0128 + n * 0x40 (n = 0 to 3) Access: User read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	MISRRSW31	MISRRSW30	MISRRSW29	MISRRSW28	MISRRSW27	MISRRSW26	MISRRSW25	MISRRSW24	MISRRSW23	MISRRSW22	MISRRSW21	MISRRSW20	MISRRSW19	MISRRSW18	MISRRSW17	MISRRSW16
W																
Reset	The reset value is chip-specific; see the chapter that describes how modules are configured and connected.															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	MISRRSW15	MISRRSW14	MISRRSW13	MISRRSW12	MISRRSW11	MISRRSW10	MISRRSW9	MISRRSW8	MISRRSW7	MISRRSW6	MISRRSW5	MISRRSW4	MISRRSW3	MISRRSW2	MISRRSW1	MISRRSW0
W																
Reset	The reset value is chip-specific; see the chapter that describes how modules are configured and connected.															

Figure 1401. STCU2 Online LBIST MISR Read Low *n* Register (STCU_LB_MISRRLSW*n*)

Table 1349. STCU_LB_MISRRLSW*n* field descriptions

Field	Description
0:31 MISRRSW[31:0]	Online MISR read low bits. This field is equivalent to the low bits (31…0) of the MISR obtained at the end of the offline LBIST execution.

66.5.42 STCU2 Online LBIST MISR Read High *n* Register (STCU_LB_MISRRHSW*n*)

The STCU_LB_MISRRHSW register report the MSB part (63…32) of the MISR obtained at the end of each online LBIST controller execution. The size of the register depends on the number of MISR bits of the respective LBIST.

The content of this register is initialized to its reset value as soon as STCU_RUNSW[RUNSW] = 1.

Offset: 0x012C + n * 0x40 (n = 0 to 3) Access: User read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	MISRRSW63	MISRRSW62	MISRRSW61	MISRRSW60	MISRRSW59	MISRRSW58	MISRRSW57	MISRRSW56	MISRRSW55	MISRRSW54	MISRRSW53	MISRRSW52	MISRRSW51	MISRRSW50	MISRRSW49	MISRRSW48
W																
Reset	The reset value is chip-specific; see the chapter that describes how modules are configured and connected.															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	MISRRSW47	MISRRSW46	MISRRSW45	MISRRSW44	MISRRSW43	MISRRSW42	MISRRSW41	MISRRSW40	MISRRSW39	MISRRSW38	MISRRSW37	MISRRSW36	MISRRSW35	MISRRSW34	MISRRSW33	MISRRSW32
W																
Reset	The reset value is chip-specific; see the chapter that describes how modules are configured and connected.															

Figure 1402. STCU2 Online LBIST MISR Read High *n* Register (STCU_LB_MISRRHSW*n*)

Table 1350. STCU_LB_MISRRHSW*n* field descriptions

Field	Description
0:31 MISRRSW[63:32]	Online MISR Read high bits. This field is equivalent to the high bits (63…32) of the MISR obtained at the end of the offline LBIST execution.

66.5.43 STCU2 MBIST Control *k* Register (STCU_MB_CTRL*k*)

The STCU_MB_CTRL register defines the control setting of MBIST controller.

The R/W fields in this register are readable at any time. You can write to these fields when either of the following conditions is true:

- Offline self-test phase is active.
- Online self-test phase is active and STCU_CFG[WRP] = 0.

Offset: 0x0600 + n * 0x4 (n = 0 to 21) Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	CSM	PTR							0	0	0	0	0	0	0	0
W																
Reset	The reset value is chip-specific; see the chapter that describes how modules are configured and connected.															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	The reset value is chip-specific; see the chapter that describes how modules are configured and connected.															

Figure 1403. STCU2 MBIST Control *k* Register (STCU_MB_CTRL*k*)

Table 1351. STCU_MB_CTRL*k* field descriptions

Field	Description
0 CSM	Concurrent/sequential mode 0 Sequential mode 1 Concurrent mode
1:7 PTR	Next LBIST or MBIST pointer PTR defines the logical pointer to the next LBIST or MBIST to be scheduled. The next LBIST or MBIST is scheduled concurrently to the current one if the CSM bit is set to '1', otherwise it is scheduled sequentially to the completion of the current MBIST execution. In case of NIL pointer, the CSM bit has to be set sequential (0) to define this is the end of the list. The self-test procedure is stopped when the current MBIST has been completed. (0x00 to LBIST - 1) — pointer to LBIST (0x10 to 0x10 + MBIST - 1) — pointer to MBIST 0x7F — pointer to NIL. No BIST execution. others — invalid pointer, an error is set in the STCU_ERR register.

66.6 Register map

The following STCU2 register map shows a full implementation case. If fewer LBISTs or MBISTs are implemented, note the following:

- The size of the STCU_LBS, STCU_LBSSW, STCU_LBE, STCU_LBESW, STCU_LBUFM depends on the number of LBIST.
- The size of STCU_MBSL, STCU_MBSLSW, STCU_MBEL, STCU_MBELSW, STCU_MBUFML depends on the amount of RAM or ROM subject to BIST.
- The size of STCU_MBSM, STCU_MBSMSW, STCU_MBEM, STCU_MBEMSW, STCU_MBUFMM depends on the amount of RAM or ROM subject to BIST only when the respective value is greater than 32, otherwise these registers are not implemented.
- The size of the STCU_MBSH, STCU_MBSHSW, STCU_MBEH, STCU_MBEHSW, STCU_MBUFMH depends on the amount of RAM or ROM subject to BIST only when the respective value is greater than 64, otherwise these registers are not implemented;
- The number of STCU_LB_CTRL, STCU_LB_PCS, STCU_LB_PRPGL, STCU_LB_PRPGH, STCU_LB_MISREL, STCU_LB_MISRELSW,

STCU_LB_MISREH, STCU_LB_MISREHSW, STCU_LB_MISRRL, STCU_LB_MISRRLSW, STCU_LB_MISRRH and STCU_LB_MISRRHSW depends on the number of LBISTs.

- The size of each STCU_LB_MISREL, STCU_LB_MISRELSW and STCU_LB_MISRRL, STCU_LB_MISRRLSW depends on the number of MISR bits of the respective LBIST.
- The size of each STCU_LB_MISREH, STCU_LB_MISREHSW, and STCU_LB_MISRRH, STCU_LB_MISRRHSW, depends on the number of MISR bits of the respective LBIST only when the respective value is greater than 32, otherwise the registers are not implemented.
- The size of STCU_LB_PRPGL and STCU_LB_PRPGH depends on the number of PRPG bits of the respective LBIST.
- The number of STCU_MB_CTRL depends on the amount of RAM or ROM subject to BIST.

Table 1352. STCU2 register map

Addr. Offset (hex code)	Register Name	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0x0000	STCU_RUN	Reserved																					BYP	MBPLLEN	LBPLLEN	Reserved							RUN
0x0004	STCU_RUNSW	Reserved																				Reserved	Reserved	MBSWPLLEN	LBSWPLLEN	Reserved						Reserved	RUNSW
0x0008	STCU_SKC	SKC																															
0x000C	STCU_CFG	Reserved	PTR							LB_DELAY								Reserved							WRP	Reserved		CRCEN	PMOSEN	MBU	CLK_CFG		
0x0010	STCU_PLL_CFG	Reserved	PLLODF5	PLLODF4	PLLODF3	PLLODF2	PLLODF1	PLLODF0		Reserved					PLLIDF2	PLLIDF1	PLLIDF0	Reserved										PLLLDF6	PLLLDF5	PLLLDF4	PLLLDF3	PLLDV2	PLLDV1	PLLDV0
0x0014	STCU_WDG	WDGEOC																															
0x0018	Reserved	Reserved																															
0x001C	STCU_CRCE	CRCE																															
0x0020	STCU_CRCR	CRCR																															
0x0024	STCU_ERR_STAT	Reserved						ABORTHW	Reserved	Reserved				LOCKESW	WDTOSW	CRCSSW	ENGESW	INVPSW	Reserved				UFSF	RFSF	Reserved			LOCKE	WDTO	CRCS	ENGE	INVP
0x0028	STCU_ERR_FM	Reserved																											LOCKEUFM	WDTOUFM	CRCSUFM	ENGEUFM	INVPUFM
0x002C	STCU_LBS	Reserved																LBS15	LBS14	LBS13	LBS12	LBS11	LBS10	LBS9	LBS8	LBS7	LBS6	LBS5	LBS4	LBS3	LBS2	LBS1	LBS0
0x0030	STCU_LBE	Reserved																LBE15	LBE14	LBE13	LBE12	LBE11	LBE10	LBE9	LBE8	LBE7	LBE6	LBE5	LBE4	LBE3	LBE2	LBE1	LBE0
0x0034	STCU_LBSSW	Reserved																LBSSW15	LBSSW14	LBSSW13	LBSSW12	LBSSW11	LBSSW10	LBSSW9	LBSSW8	LBSSW7	LBSSW6	LBSSW5	LBSSW4	LBSSW3	LBSSW2	LBSSW1	LBSSW0

Table 1352. STCU2 register map (continued)

Addr. Offset (hex code)	Register Name	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0x0038	STCU_LBESW	Reserved																LBESW15	LBESW14	LBESW13	LBESW12	LBESW11	LBESW10	LBESW9	LBESW8	LBESW7	LBESW6	LBESW5	LBESW4	LBESW3	LBESW2	LBESW1	LBESW0
0x003C	Reserved	Reserved																															
0x0040	STCU_LBUFM	Reserved																LBUFM15	LBUFM14	LBUFM13	LBUFM12	LBUFM11	LBUFM10	LBUFM9	LBUFM8	LBUFM7	LBUFM6	LBUFM5	LBUFM4	LBUFM3	LBUFM2	LBUFM1	LBUFM0
0x0044	STCU_MBSL	MBS31	MBS30	MBS29	MBS28	MSB27	MBS26	MBS25	MBS24	MBS23	MBS22	MBS21	MBS20	MBS19	MBS18	MBS17	MBS16	MBS15	MBS14	MBS13	MBS12	MBS11	MBS10	MBS9	MBS8	MBS7	MBS6	MBS5	MBS4	MBS3	MBS2	MBS1	MBS0
0x0048	STCU_MBSM	MBS63	MBS62	MBS61	MBS60	MBS59	MBS58	MBS57	MBS56	MBS55	MBS54	MBS53	MBS52	MBS51	MBS50	MBS49	MBS48	MBS47	MBS46	MBS45	MBS44	MBS43	MBS42	MBS41	MBS40	MBS39	MBS38	MBS37	MBS36	MBS35	MBS34	MBS33	MBS32
0x004C	STCU_MBSH	MBS95	MBS94	MBS93	MBS92	MBS91	MBS90	MBS89	MBS88	MBS87	MBS86	MBS85	MBS84	MBS83	MBS82	MBS81	MBS80	MBS79	MBS78	MBS77	MBS76	MBS75	MBS74	MBS73	MBS72	MBS71	MBS70	MBS69	MBS68	MBS67	MBS66	MBS65	MBS64
0x0050	STCU_MBEL	MBE31	MBE30	MBE29	MBE28	MBE27	MBE26	MBE25	MBE24	MBE23	MBE22	MBE21	MBE20	MBE19	MBE18	MBE17	MBE16	MBE15	MBE14	MBE13	MBE12	MBE11	MBE10	MBE9	MBE8	MBE7	MBE6	MBE5	MBE4	MBE3	MBE2	MBE1	MBE0
0x0054	STCU_MBEM	MBE63	MBE62	MBE61	MBE60	MBE59	MBE58	MBE57	MBE56	MBE55	MBE54	MBE53	MBE52	MBE51	MBE50	MBE49	MBE48	MBE47	MBE46	MBE45	MBE44	MBE43	MBE42	MBE41	MBE40	MBE39	MBE38	MBE37	MBE36	MBE35	MBE34	MBE33	MBE32
0x0058	STCU_MBEH	MBE95	MBE94	MBE93	MBE92	MBE91	MBE90	MBE89	MBE88	MBE87	MBE86	MBE85	MBE84	MBE83	MBE82	MBE81	MBE80	MBE79	MBE78	MBE77	MBE76	MBE75	MBE74	MBE73	MBE72	MBE71	MBE70	MBE69	MBE68	MBE67	MBE66	MBE65	MBE64
0x005C	STCU_MBSLSW	MBSSW31	MBSSW30	MBSSW29	MBSSW28	MBSSW27	MBSSW26	MBSSW25	MBSSW24	MBSSW23	MBSSW22	MBSSW21	MBSSW20	MBSSW19	MBSSW18	MBSSW17	MBSSW16	MBSSW15	MBSSW14	MBSSW13	MBSSW12	MBSSW11	MBSSW10	MBSSW9	MBSSW8	MBSSW7	MBSSW6	MBSSW5	MBSSW4	MBSSW3	MBSSW2	MBSSW1	MBSSW0
0x0060	STCU_MBSMSW	MBSSW63	MBSSW62	MBSSW61	MBSSW60	MBSSW59	MBSSW58	MBSSW57	MBSSW56	MBSSW55	MBSSW54	MBSSW53	MBSSW52	MBSSW51	MBSSW50	MBSSW49	MBSSW48	MBSSW47	MBSSW46	MBSSW45	MBSSW44	MBSSW43	MBSSW42	MBSSW41	MBSSW40	MBSSW39	MBSSW38	MBSSW37	MBSSW36	MBSSW35	MBSSW34	MBSSW33	MBSSW32
0x0064	STCU_MBSHSW	MBSSW95	MBSSW94	MBSSW93	MBSSW92	MBSSW91	MBSSW90	MBSSW89	MBSSW88	MBSSW87	MBSSW86	MBSSW85	MBSSW84	MBSSW83	MBSSW82	MBSSW81	MBSSW80	MBSSW79	MBSSW78	MBSSW77	MBSSW76	MBSSW75	MBSSW74	MBSSW73	MBSSW72	MBSSW71	MBSSW70	MBSSW69	MBSSW68	MBSSW67	MBSSW66	MBSSW65	MBSSW64
0x0068	STCU_MBELSW	MBESW31	MBESW30	MBESW29	MBESW28	MBESW27	MBESW26	MBESW25	MBESW24	MBESW23	MBESW22	MBESW21	MBESW20	MBESW19	MBESW18	MBESW17	MBESW16	MBESW15	MBESW14	MBESW13	MBESW12	MBESW11	MBESW10	MBESW9	MBESW8	MBESW7	MBESW6	MBESW5	MBESW4	MBESW3	MBESW2	MBESW1	MBESW0
0x006C	STCU_MBEMSW	MBESW63	MBESW62	MBESW61	MBESW60	MBESW59	MBESW58	MBESW57	MBESW56	MBESW55	MBESW54	MBESW53	MBESW52	MBESW51	MBESW50	MBESW49	MBESW48	MBESW47	MBESW46	MBESW45	MBESW44	MBESW43	MBESW42	MBESW41	MBESW40	MBESW39	MBESW38	MBESW37	MBESW36	MBESW35	MBESW34	MBESW33	MBESW32
0x0070	STCU_MBEHSW	MBESW95	MBESW94	MBESW93	MBESW92	MBESW91	MBESW90	MBESW89	MBESW88	MBESW87	MBESW86	MBESW85	MBESW84	MBESW83	MBESW82	MBESW81	MBESW80	MBESW79	MBESW78	MBESW77	MBESW76	MBESW75	MBESW74	MBESW73	MBESW72	MBESW71	MBESW70	MBESW69	MBESW68	MBESW67	MBESW66	MBESW65	MBESW64
0x0074	STCU_MBUFML	MBUFM 31	MBUFM 30	MBUFM 29	MBUFM 28	MBUFM 27	MBUFM 26	MBUFM 25	MBUFM 24	MBUFM23	MBUFM22	MBUFM21	MBUFM20	MBUFM19	MBUFM18	MBUFM17	MBUFM16	MBUFM15	MBUFM14	MBUFM13	MBUFM12	MBUFM11	MBUFM10	MBUFM9	MBUFM8	MBUFM7	MBUFM6	MBUFM5	MBUFM4	MBUFM3	MBUFM2	MBUFM1	MBUFM0
0x0078	STCU_MBUFMM	MBUFM63	MBUFM62	MBUFM61	MBUFM60	MBUFM59	MBUFM58	MBUFM57	MBUFM56	MBUFM55	MBUFM54	MBUFM53	MBUFM52	MBUFM51	MBUFM50	MBUFM49	MBUFM48	MBUFM47	MBUFM46	MBUFM45	MBUFM44	MBUFM43	MBUFM42	MBUFM41	MBUFM40	MBUFM39	MBUFM38	MBUFM37	MBUFM36	MBUFM35	MBUFM34	MBUFM33	MBUFM32
0x007C	STCU_MBUFMH	MBUFM95	MBUFM94	MBUFM93	MBUFM92	MBUFM91	MBUFM90	MBUFM89	MBUFM88	MBUFM87	MBUFM86	MBUFM85	MBUFM84	MBUFM83	MBUFM82	MBUFM81	MBUFM80	MBUFM79	MBUFM78	MBUFM77	MBUFM76	MBUFM75	MBUFM74	MBUFM73	MBUFM72	MBUFM71	MBUFM70	MBUFM69	MBUFM68	MBUFM67	MBUFM66	MBUFM65	MBUFM64
Address space not used (Available for future expansion)																																	

Table 1352. STCU2 register map (continued)

Addr. Offset (hex code)	Register Name	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0x0100 + n*0x40 n=0 to (NLBIST-1)	STCU_LB_CTRL*n*	CSM	PTR							Reserved				PRPGEN	SHS			SCEN_OFF				SCEN_ON				Reserved				PFT	CWS		
0x0104 + n*0x40 n=0 to (NLBIST-1)	STCU_LB_PCS*n*	Reserved						PCS																									
0x0108 + n*0x40 n=0 to (NLBIST-1)	STCU_LB_PRPGL*n*	PRPG31	PRPG30	PRPG29	PRPG28	PRPG27	PRPG26	PRPG25	PRPG24	PRPG23	PRPG22	PRPG21	PRPG20	PRPG19	PRPG18	PRPG17	PRPG16	PRPG15	PRPG14	PRPG13	PRPG12	PRPG11	PRPG10	PRPG9	PRPG8	PRPG7	PRPG6	PRPG5	PRPG4	PRPG3	PRPG2	PRPG1	PRPG0
0x010C + n*0x40 n=0 to (NLBIST-1)	STCU_LB_PRPGH*n*	PRPG63	PRPG62	PRPG61	PRPG60	PRPG59	PRPG58	PRPG57	PRPG56	PRPG55	PRPG54	PRPG53	PRPG52	PRPG51	PRPG50	PRPG49	PRPG48	PRPG47	PRPG46	PRPG45	PRPG44	PRPG43	PRPG42	PRPG41	PRPG40	PRPG39	PRPG38	PRPG37	PRPG36	PRPG35	PRPG34	PRPG33	PRPG32
0x0110 + n*0x40 n=0 to (NLBIST-1)	STCU_LB_MISREL*n*	MISRE31	MISRE30	MISRE29	MISRE28	MISRE27	MISRE26	MISRE25	MISRE24	MISRE23	MISRE22	MISRE21	MISRE20	MISRE19	MISRE18	MISRE17	MISRE16	MISRE15	MISRE14	MISRE13	MISRE12	MISRE11	MISRE10	MISRE9	MISRE8	MISRE7	MISRE6	MISRE5	MISRE4	MISRE3	MISRE2	MISRE1	MISRE0
0x0114 + n*0x40 n=0 to (NLBIST-1)	STCU_LB_MISREH*n*	MISRE63	MISRE62	MISRE61	MISRE60	MISRE59	MISRE58	MISRE57	MISRE56	MISRE55	MISRE54	MISRE53	MISRE52	MISRE51	MISRE50	MISRE49	MISRE48	MISRE47	MISRE46	MISRE45	MISRE44	MISRE43	MISRE42	MISRE41	MISRE40	MISRE39	MISRE38	MISRE37	MISRE36	MISRE35	MISRE34	MISRE33	MISRE32
0x0118 + n*0x40 n=0 to (NLBIST-1)	STCU_LB_MISRRL*n*	MISRR31	MISRR30	MISRR29	MISRR28	MISRR27	MISRR26	MISRR25	MISRR24	MISRR23	MISRR22	MISRR21	MISRR20	MISRR19	MISRR18	MISRR17	MISRR16	MISRR15	MISRR14	MISRR13	MISRR12	MISRR11	MISRR10	MISRR9	MISRR8	MISRR7	MISRR6	MISRR5	MISRR4	MISRR3	MISRR2	MISRR1	MISRR0
0x011C + n*0x40 n=0 to (NLBIST-1)	STCU_LB_MISRRH*n*	MISRR63	MISRR62	MISRR61	MISRR60	MISRR59	MISRR58	MISRR57	MISRR56	MISRR55	MISRR54	MISRR53	MISRR52	MISRR51	MISRR50	MISRR49	MISRR48	MISRR47	MISRR46	MISRR45	MISRR44	MISRR43	MISRR42	MISRR41	MISRR40	MISRR39	MISRR38	MISRR37	MISRR36	MISRR35	MISRR34	MISRR33	MISRR32
0x0120 + n*0x40 n=0 to (NLBIST-1)	STCU_LB_MISRELSW*n*	MISRESW31	MISRESW30	MISRESW29	MISRESW28	MISRESW27	MISRESW26	MISRESW25	MISRESW24	MISRESW23	MISRESW22	MISRESW21	MISRESW20	MISRESW19	MISRESW18	MISRESW17	MISRESW16	MISRESW15	MISRESW14	MISRESW13	MISRESW12	MISRESW11	MISRESW10	MISRESW9	MISRESW8	MISRESW7	MISRESW6	MISRESW5	MISRESW4	MISRESW3	MISRESW2	MISRESW1	MISRESW0
0x0124 + n*0x40 n=0 to (NLBIST-1)	STCU_LB_MISREHSW*n*	MISRESW63	MISRESW62	MISRESW61	MISRESW60	MISRESW59	MISRESW58	MISRESW57	MISRESW56	MISRESW55	MISRESW54	MISRESW53	MISRESW52	MISRESW51	MISRESW50	MISRESW49	MISRESW48	MISRESW47	MISRESW46	MISRESW45	MISRESW44	MISRESW43	MISRESW42	MISRESW41	MISRESW40	MISRESW39	MISRESW38	MISRESW37	MISRESW36	MISRESW35	MISRESW34	MISRESW33	MISRESW32
0x0128 + n*0x40 n=0 to (NLBIST-1)	STCU_LB_MISRRLSW*n*	MISRRSW31	MISRRSW30	MISRRSW29	MISRRSW28	MISRRSW27	MISRRSW26	MISRRSW25	MISRRSW24	MISRRSW23	MISRRSW22	MISRRSW21	MISRRSW20	MISRRSW19	MISRRSW18	MISRRSW17	MISRRSW16	MISRRSW15	MISRRSW14	MISRRSW13	MISRRSW12	MISRRSW11	MISRRSW10	MISRRSW9	MISRRSW8	MISRRSW7	MISRRSW6	MISRRSW5	MISRRSW4	MISRRSW3	MISRRSW2	MISRRSW1	MISRRSW0
0x012C + n*0x40 n=0 to (NLBIST-1)	STCU_LB_MISRRHSW*n*	MISRRSW63	MISRRSW62	MISRRSW61	MISRRSW60	MISRRSW59	MISRRSW58	MISRRSW57	MISRRSW56	MISRRSW55	MISRRSW54	MISRRSW53	MISRRSW52	MISRRSW51	MISRRSW50	MISRRSW49	MISRRSW48	MISRRSW47	MISRRSW46	MISRRSW45	MISRRSW44	MISRRSW43	MISRRSW42	MISRRSW41	MISRRSW40	MISRRSW39	MISRRSW38	MISRRSW37	MISRRSW36	MISRRSW35	MISRRSW34	MISRRSW33	MISRRSW32
Address space not used (available for future expansion)																																	
0x0600 + n*0x4 n=0 to (NMCUT-1)	STCU_MB_CTRL*n*	CSM	PTR							Reserved																							

66.7 Use cases and limitations

The STCU2 module is designed to give high flexibility during the online and offline self-test. This section shows the programming operations that have to be performed to correctly use this module in three typical use cases:

- Offline self-test sequence after a reset trigger event.
- Bypass mode of the self-test sequence after a reset trigger event.
- Online self-test sequence.

66.7.1 Offline self-test sequence

This is the typical mode of using the STCU2 module after reset trigger event is applied to STCU2. The SSCM DCF bus is used to retrieve the STCU2 schedule and LBIST or MBIST execution parameters stored in the NVM memory. The target is to cover the amount of physical defects in the digital logic and in the system RAM or ROM according to the system specifications. The sequence is the following:

- Unlock the offline STCU2 access writing the Key1/Key2 sequence in the STCU_SKC register.
- Program the LBIST fault reaction conditions (UF/RF) setting the STCU_LBUFM register.
- Program the MBIST fault reaction conditions (UF/RF) setting, depending on the number of memories, the STCU_MBUFML/M/H registers.
- Program the STCU2 internal fault reaction conditions (UF/RF) setting the STCU_ERR_FM register.
- Program the STCU_LB_CTRL, STCU_LB_PCS, STCU_MISREL/H registers of each LBIST to be executed. Optionally, the STCU_LB_PRPGL/H can also be programmed to load a specific seed.
- Program the STCU_MB_CTRL registers of each NMCUT to be executed.
- Program the STCU_WDG register to define the time budget assigned for LBIST or MBIST execution.
- Program the STCU_CFG register to define:
  - the core and LBIST or MBIST TCK clock prescaling factor setting the CLK_CFG bits,
  - the LB_DELAY field to delay the LBIST start when many LBIST are run concurrently to mitigate the potential transient power issue during the start-up phase,
  - the PMOSEN or CHKBRD to define the algorithm to be used during MBIST run,
  - the CRCREN bit to enable the self-checking feature,
  - set the pointer to the first LBIST/NMCUT to be executed.
- If the internal CRC comparison has been enabled, program the CRCE value expected at the end of the offline self-test sequence in the STCU_CRCE register
- If PLL usage IS enabled, program the STCU_PLL_CFG value to set the required system frequency.

- Program the RUN bit in the STCU_RUN register to enable the offline self-test execution and the L/MBPLLEN bits to enable the PLL during LBIST or MBIST execution according to STCU_PLL_CFG register content.
- Start the offline self-test depending on PLL management and LBIST or MBIST execution order.
- Wait the WDGEOC execution run time specified in the STCU_WDG register.
- The STCU2 switches-off the core clock at the end of the self-test run and releases the PLL control if this feature is enabled.
- The STCU2 resets the STCU_RUN[RUN] bit and the device exits the boot sequence. If there are failures, the STCU2 flags these failures to the FCCU by raising one or both of the stcu_ncf and stcu_cf lines.
- The software application reads the STCU_LBS flag register to check the failed LBIST when UF or RF conditions are detected.
- The software application reads the STCU_LBE flag register to check the LBIST still on-going when UF or RF are detected.
- The software application reads the STCU_MBSL/M/H flag registers, depending on the number of memories, to check the failed RAM or ROM memory BIST when UF or RF are detected.
- The software application reads the STCU_MBEL/M/H flag registers, depending on the number of memories, to check the RAM or ROM memory BIST still running when UF or RF are detected.
- The software application reads the bit INV, ENGE, CRCS, WDTO, LOCKE of the STCU_ERR_STAT register to check whether there has been an internal STCU2 engine/parameters failure when UF or RF are detected.
- If the CRCEN bit is enabled, the software application reads the CRCE and CRCR registers to check the coherence with the bit CRCS of the STCU_ERR register.
- For each LBIST, the software application reads the STCU_LBMISREL/H and STCU_LBIST_MISRRL/H registers to check the coherence with the STCU_LBS bits.

Note: The software application reads operations on the STCU2_CRC and LBIST_MISR registers after offline self-test execution has successfully completed are an additional safety layer added to improve the already implemented auto self-test feature included in STCU2.

66.7.2 Online self-test Sequence

This is the typical mode of using the STCU2 module during the application run. The IPS interface is used to program the STCU2 registers and to schedule the LBIST or MBIST execution. Since the online operations clean up the content of the respective digital logic (LBIST) or RAM (MBIST), activation of this mode requires caution. Follow this sequence:

- Unlock the online STCU2 access writing the Key1/Key2 sequence in the STCU_SKC register.
- Check and clear the STCU_CFG[WRP] bit to open the IPS access on the STCU2 online self-test registers.
- The LBIST, MBIST and STCU2 internal fault reaction should already be programmed during the offline self-test sequence. If they have not been programmed or there is a need to change them, they can be overwritten by setting the registers: STCU_LBUFM, STCU_MBUFML/M/H and STCU_ERR_FM.
- The STCU_LB_CTRL, STCU_LB_PCS should already be programmed during offline self-test sequence. If they have not been programmed or there is a need to change

them, they can be overwritten. The registers STCU_LB_MISRELSW/HSW have to be programmed according to the expected signature.

- Overwrite/Program the STCU_MB_CTRL registers of each NMCUT to be executed.
- Overwrite/Program the STCU_WDG register to define the time budget assigned for LBIST or MBIST execution.
- Overwrite/Program the STCU_CFG register to define:
  - the core and LBIST or MBIST TCK clock prescaling factor setting the CLK_CFG bits,
  - the LB_DELAY field to delay the LBIST start when many LBISTs are running simultaneously to mitigate the potential transient power issue during the start-up phase,
  - the PMOSEN or CHKBRD to define the algorithm to be used during MBIST run,
  - the CRCREN bit to enable the self-checking feature
  - set the pointer to the first LBIST/NMCUT to be executed.
- If the internal CRC comparison is enabled, overwrite/program the CRCE value expected at the end of the online self-test sequence in the STCU_CRCE register.
- Program the STCU_RUNSW[RUNSW] bit to enable the online self-test execution and the L/MBSWPLLEN bits to enable the PLL lock signal monitor during LBIST or MBIST execution.
- Start the online self-test depending on PLL management and LBIST or MBIST execution order.
- Wait the WDGEOC execution run time specified in the STCU_WDG register.
- The STCU2 switches off the core clock at the end of the self-test run.
- After online self-test completion, a functional reset is triggered in the SoC. In case core tries to access any peripheral/memory during online self test, access is not guaranteed.
- The STCU2 resets the STCU_RUNSW[RUNSW] bit. In both the reset cases and in case of errors, the STCU2 flags the failure to the FCCU by rising one or both of the stcu_ncf and stcu_cf lines.
- The software application reads the STCU_LBSSW flag register to check the failed LBIST when UF or RF conditions are detected.
- The software application reads the STCU_LBESW flag register to check the LBIST is still running when UFs or RFs are detected.
- The software application reads the STCU_MBSLSW/MSW/HSW flag registers, depending on the number of memories, to check the failed RAM/ROM memory BIST when UFs or RFs are detected.
- The software application reads the STCU_MBELSW/MSW/HSW flag registers, depending on the number of memories, to check the RAM/ROM memory BIST is still running when UFs or RFs are detected.
- The software application reads the bit INVPSW, ENGESW, CRCSSW, WDTOSW, LOCKESW of the STCU_ERR_STAT register to check whether there has been an internal STCU2 engine/parameters failure when UFs or RFs are detected.
- If the CRCEN bit is enabled, the software application reads the CRCE and CRCR registers to check the coherence with the STCU_ERR[CRCSSW] bit.
- For each LBIST, the software application reads the STCU_LBMISRELSW/HSW and STCU_LBIST_MISRRLSW/HSW registers to check the coherence with the STCU_LBS bits

Note: The software application reads operations on the STCU2 CRC and LBIST MISR registers after online self-test execution has successfully completed are an additional safety layer added to improve the already implemented auto self-test feature included in the STCU2.

66.7.3 Bypass USER Mode

In this mode, the user application parameters stored in Flash and read by SSCM are written to skip the self-test procedure after a reset trigger event is applied. It might be useful if the device is not configured for Safety applications. Follow this sequence:

1. Unlock the STCU2 access writing the offline Key1/Key2 sequence in the STCU_SKC register.
2. Set the BYP bit in the STCU_RUN register.

After the BYP bit is set, the core clock is switched off, the self-test sequence is not applied and the system can wake up and start the user application.

66.7.4 Design implementation information

The following list reports the limitations regarding the current implementation:

- Every NMCUT MBIST or LBIST can be run one time during the on/offline self-test procedure.
- It is not possible to run LBIST and MBIST concurrently.
- As the software has full control of the system while applications are running during the online self-test, the STCU2 may not take control of the PLL, but only monitor the lock signal to flag incorrect unlock events.

67 Register Protection (REG_PROT)

67.1 Introduction

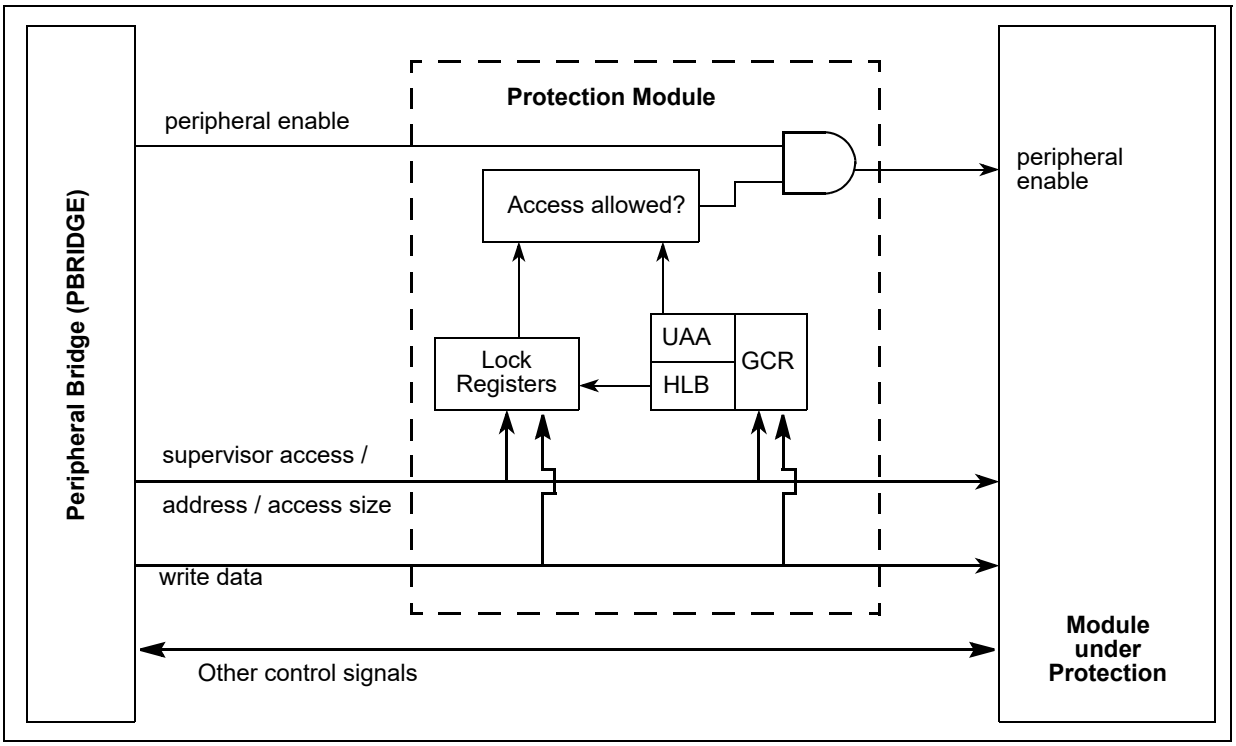
67.1.1 Overview

The Register Protection (REG_PROT) module offers a mechanism to protect defined memory-mapped address locations in a module under protection from being written. The address locations that can be protected are module-specific.

Note: Please see Chapter 7: Device configuration for device-specific details of the protected registers and bytes.

The protection module is located between the module under protection and the peripheral bridge (PBRIDGE). This is shown in *Figure 1404*.

Figure 1404. REG_PROT Block Diagram



67.1.2 Features

Register Protection includes these distinctive features:

- Restrict write accesses for the module under protection to supervisor mode only
- Lock registers for first 6 KB of memory-mapped address space
- Write to address mirror automatically sets corresponding lock bit
- Once configured lock bits can be protected from changes

Also, hard lock bit protection can only be cleared by a system reset once set.

67.1.3 Modes of operation

The Register Protection module is operable when the module under protection is operable.

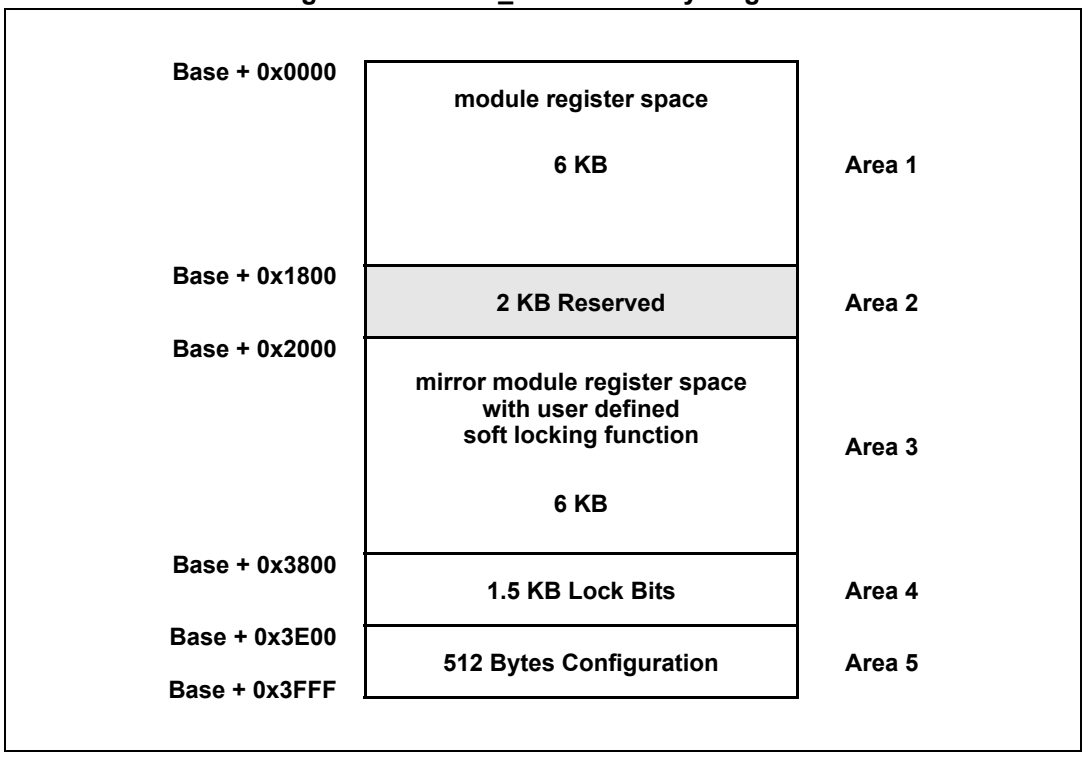
67.2 External signal description

There are no external signals.

67.3 Memory map and register definition

This section provides a detailed description of the memory map of a module with register protection. The original 16 KB module memory space is divided into five areas as shown in *Figure 1405*.

Figure 1405. REG_PROT memory diagram



- Area 1 is 6 KB and holds the normal functional module registers and is transparent for all read/write operations.
- Area 2 is 2 KB starting at offset 0x1800 is a reserved area, which shall not be accessed.
- Area 3 is 6 KB, starting at offset 0x2000 and is a mirror of area 1. A read/write access to an offset 0x2000+X will read/write the register at offset X. However a write access to offset 0x2000+X will additionally set the optional Soft Lock Bits for this offset X in the same cycle as the register at offset X is written. This provides for an atomic write and lock operation. Not all registers in area 1 need to have protection defined by associated

Soft Lock Bits. For unprotected registers at offset Y, accesses to offset 0x2000+Y will be identical to accesses at offset Y.

- Area 4 is 1.5 KB and holds the Soft Lock Bits, one bit per byte in area 1. The four Soft Lock Bits associated with one module register word are arranged at byte boundaries in the memory map. The Soft Lock Bit registers can be directly written using a bit mask.
- Area 5 is 512 bytes large and holds the configuration bits of the protection mode. There is one configuration hard lock bit per module that prevents all further modifications to the Soft Lock Bits and can only be cleared by a system reset once set. The other bits, if set, will allow user access to the protected module.

If any locked byte is accessed with a write transaction, a transfer error will be issued to the system and the write transaction will not be executed. This is true even if not all accessed bytes are locked.

Accessing unimplemented 32-bit registers in Areas 4 and 5 will result in a transfer error.

67.3.1 Memory map

Table 1353 shows the memory map for a module which is protected via a REG_PROT module.

Table 1353. REG_PROT memory map

Offset	Use	Location
0x0000	Module Register 0 (MR0)	*Section 67.3.2.1*
0x0001	Module Register 1 (MR1)	*Section 67.3.2.1*
0x0002	Module Register 2 (MR2)	*Section 67.3.2.1*
0x0003 - 0x17FF	Module Register 3 (MR3) - Module Register 6143(MR6143)	*Section 67.3.2.1*
0x1800 - 0x1FFF	Reserved	
0x2000	Module Register 0 (MR0) + Set Soft Lock Bit 0 (LMR0)	*Section 67.3.2.2*
0x2001	Module Register 1 (MR1) + Set Soft Lock Bit 1 (LMR1)	*Section 67.3.2.2*
0x2002 - 0x37FF	Module Register 2 (MR2) + Set Soft Lock Bit 2 (LMR2) - Module Register 6143 (MR6143) + Set Soft Lock Bit 6143 (LMR6143)	*Section 67.3.2.2*
0x3800	Soft Lock Bit Register 0 (SLBR0): Soft Lock Bits 0-3	*Section 67.3.2.3*
0x3801	Soft Lock Bit Register 1 (SLBR1): Soft Lock Bits 4-7	*Section 67.3.2.3*
0x3802 - 0x3DFF	Soft Lock Bit Register 2 (SLBR2): Soft Lock Bits 8-11 - Soft Lock Bit Register 1535 (SLBR1535): Soft Lock Bits 6140-6143	*Section 67.3.2.3*
0x3E00 - 0x3FFB	Reserved	
0x3FFC	Global Configuration Register (GCR)	*Section 67.3.2.4*

Note: Reserved registers in area #1 will be handled as specified in the protected module's documentation.

67.3.2 Register descriptions

This section describes in address order all the REG_PROT registers. Each description includes a standard register diagram with an associated figure number. Details of register bit and field function follow the register diagrams, in bit order.

67.3.2.1 Module Registers (MR0-6143)

This is the lower 6 KB module memory space which holds all the functional registers of the module that is protected by the REG_PROT module.

67.3.2.2 Module Register and Set Soft Lock Bit (LMR0-6143)

This is memory area #3 that provides mirrored access to the MR0-6143 registers with the side effect of setting Soft Lock Bits in case of a write access to a MR that is defined as protectable by the locking mechanism. Each MR is protectable by one associated bit in a SLBR*n*.SLB*m*, according to the mapping described in *Table 1355*.

67.3.2.3 Soft Lock Bit Register (SLBR0-1535)

The Soft Lock Bit Registers hold the Soft Lock Bits for the protected registers in memory area #1, which is the normal register address space of the protected module. Each SLB register has a four Soft Lock Bits (SLB0-SLB3), each of which controls write access to a byte in memory area #1. Each Soft Lock Bit also has a corresponding Write Enable bit in the same register that controls whether the Soft Lock Bit can be written. The following table shows the mapping between the Soft Lock Bits to the bytes in memory area

Table 1354. Soft lock bits vs. protected address

Soft Lock Bit	Protected address
SLBR0.SLB0	MR0
SLBR0.SLB1	MR1
SLBR0.SLB2	MR2
SLBR0.SLB3	MR3
SLBR1.SLB0	MR4
SLBR1.SLB1	MR5
SLBR1.SLB2	MR6
SLBR1.SLB3	MR7
SLBR2.SLB0	MR8
...	...

Note: As many as 1536 Soft Lock Bit Registers (SLBRn) may be implemented, depending on the number of protected module register bytes.

Access in User Mode is read-only; in Supervisor Mode access is read/write.

Offset: 0x3800-0x3DFF

Access: Read always
Supervisor write

	0	1	2	3	4	5	6	7
R	0	0	0	0	SLB0	SLB1	SLB2	SLB3
W	WE0	WE1	WE2	WE3				
Reset	0	0	0	0	0	0	0	0

Figure 1406. Soft Lock Bit Register (SLBR*n*)

Table 1355. SLBR*n* field descriptions

Field	Description
0 WE0	Write Enable Bits for Soft Lock Bits (SLB): WE0 enables writing to SLB0 1 Value is written to SLB 0 SLB is not modified
1 WE1	Write Enable Bits for Soft Lock Bits (SLB): WE1 enables writing to SLB1 1 Value is written to SLB 0 SLB is not modified
2 WE2	Write Enable Bits for Soft Lock Bits (SLB): WE2 enables writing to SLB2 1 Value is written to SLB 0 SLB is not modified
3 WE3	Write Enable Bits for Soft Lock Bits (SLB): WE3 enables writing to SLB3 1 Value is written to SLB 0 SLB is not modified
4 SLB0	Soft Lock Bits for one MR*n* register: SLB0 can block accesses to MR[*n* *4 + 0] 1 Associated MR*n* byte is locked against write accesses 0 Associated MR*n* byte is unprotected and writable
5 SLB1	Soft Lock Bits for one MR*n* register: SLB1 can block accesses to MR[*n* *4 + 1] 1 Associated MR*n* byte is locked against write accesses 0 Associated MR*n* byte is unprotected and writable
6 SLB2	Soft Lock Bits for one MR*n* register: SLB2 can block accesses to MR[*n* *4 + 2] 1 Associated MR*n* byte is locked against write accesses 0 Associated MR*n* byte is unprotected and writable
7 SLB3	Soft Lock Bits for one MR*n* register: SLB3 can block accesses to MR[*n* *4 + 3] 1 Associated MR*n* byte is locked against write accesses 0 Associated MR*n* byte is unprotected and writable

67.3.2.4 Global Configuration Register (GCR)

This register controls the configuration of the locking of the SLB's and the level of user access allowed to the protected registers.

Note: Access in User Mode is read-only; in Supervisor Mode access is read/write.

Offset: 0x3FFC

Access: Read Always
Supervisor write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	HLB	0	0	0	0	0	0	0	UAA	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

Figure 1407. Global Configuration Register (GCR)

Table 1356. GCR field descriptions

Field	Description
0 HLB	Hard Lock Bit. This register can not be cleared once it is set by software. It can only be cleared by a system reset. 1 All SLB bits are write protected and can not be modified 0 All SLB bits are accessible and can be modified.
8 UAA	User Access Allowed. 1 The registers in the module under protection can be accessed in the mode defined for the module registers without any additional restrictions. 0 The registers in the module under protection can only be written in supervisor mode. All write accesses in non-supervisor mode are not executed and a transfer error is issued. This access restriction is in addition to any access restrictions imposed by the protected IP module.

Note: The GCR.UAA bit has no effect on the allowed access modes for the registers in the REG_PROT module.

67.4 Functional description

The following sections describe the functional characteristics of the Register Protection module.

67.4.1 General

This module provides a generic register (address) write-protection mechanism. The protection size can be:

- 32-bit (address == multiples of 4)
- 16-bit (address == multiples of 2)
- 8-bit (address == multiples of 1)

Which addresses are protected and the protection size is device and module specific, please refer to *Chapter 7: Device configuration for details.*

For all addresses that are protected there are SLBR*n*.SLB*m* bits that specify whether the address is locked. When an address is locked it can only be read but not written in any mode (supervisor/normal). If an address is unprotected the corresponding SLBR*n*.SLB*m* bit is always 0b0 no matter what software is writing to.

67.4.2 Change lock settings

To change the setting whether an address is locked or unlocked the corresponding SLBR*n*.SLB*m* bit needs to be changed. This can be done using the following methods:

- Modify the SLBR*n*.SLB*m* directly by writing to area #4
- Set the SLBR*n*.SLB*m* bit(s) by writing to the mirror module space (area #3)

Both methods are explained in the following sections.

67.4.2.1 Change lock settings directly via Area #4

In memory area #4 the lock bits are located. They can be modified by writing to them. Each SLBR*n*.SLB*m* bit has a mask bit SLBR*n*.WE*m* which protects it from being modified. This masking makes clear-modify-write operations unnecessary.

Figure 1408 shows two modification examples. In the left example there is a write access to the SLBR*n* register specifying a mask value which allows modification of all SLBR*n*.SLB*m* bits. The example on the right specifies a mask which only allows modification of the bits SLBR*n*.SLB[3:1].

Figure 1408. Change lock settings directly via Area #4

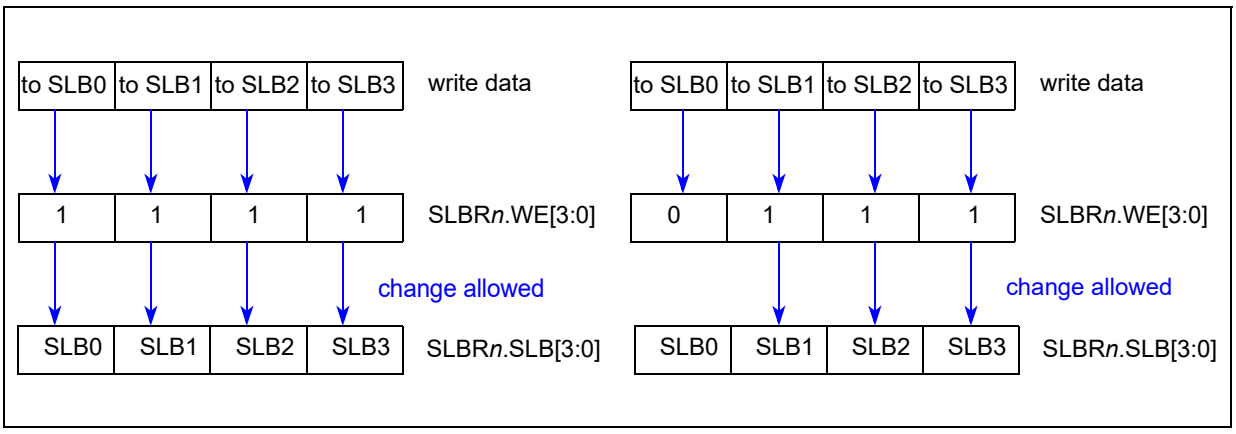
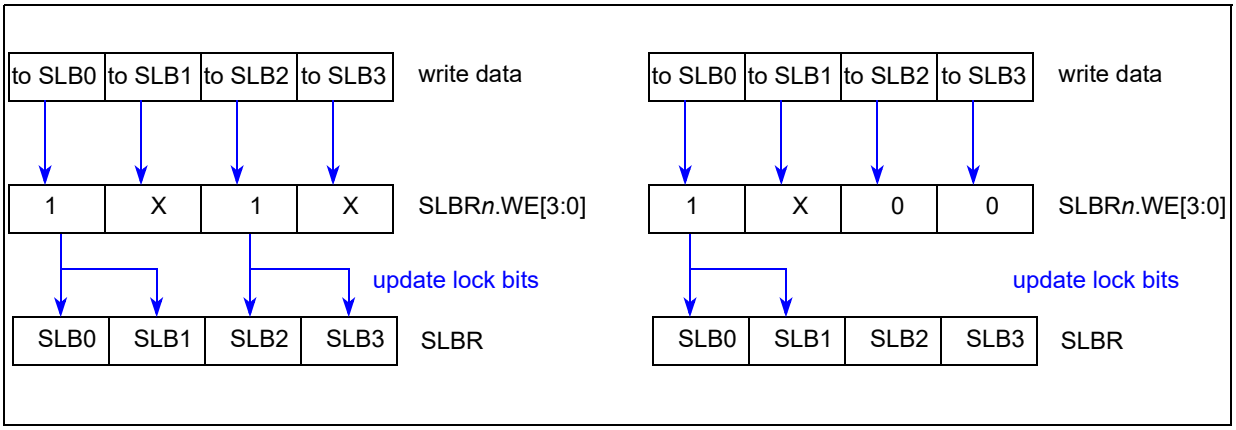


Figure 1408 shows four registers that can be protected 8-bit wise. In *Figure 1409* registers with 16-bit protection and in *Figure 1410* registers with 32-bit protection are shown:

Figure 1409. Change lock settings for 16-bit protected addresses

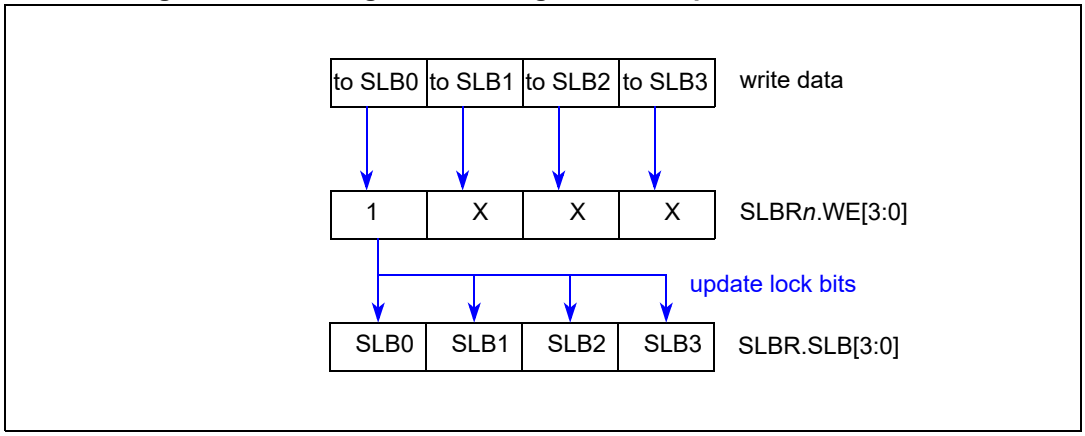


On the right side of *Figure 1409* it is shown that the data written to SLBR*n*.SLB[0] is automatically written to SLBR*n*.SLB[1] also. This is done as the address reflected by SLBR*n*.SLB[0] is protected 16-bit wise. Note that in this case the write enable SLBR*n*.WE[0] must be set while SLBR*n*.WE[1] does not matter. As the enable bits SLBR*n*.WE[3:2] are cleared the lock bits SLBR*n*.SLB[3:2] remain unchanged.

In the example on the left side of *Figure 1409* the data written to SLBR*n*.SLB[0] is mirrored to SLBR*n*.SLB[1] and the data written to SLBR*n*.SLB[2] is mirrored to SLBR*n*.SLB[3] as for both registers the write enables are set.

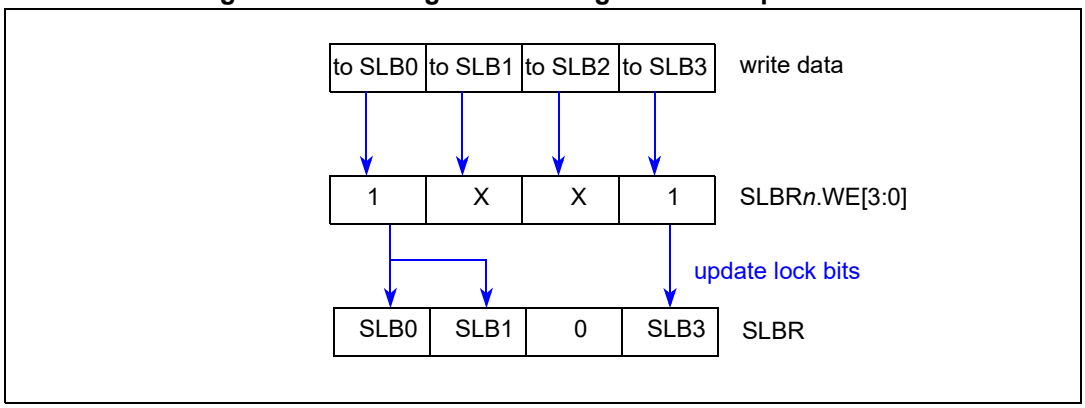
In *Figure 1410* a 32-bit wise protected register is shown. When SLBR*n*.WE[0] is set the data written to SLBR*n*.SLB[0] is automatically written to SLBR*n*.SLB[3:1] also. Otherwise SLBR*n*.SLB[3:0] remains unchanged.

Figure 1410. Change lock settings for 32-bit protected addresses



In *Figure 1411* an example is shown which has a mixed protection size configuration:

Figure 1411. Change lock settings for mixed protection

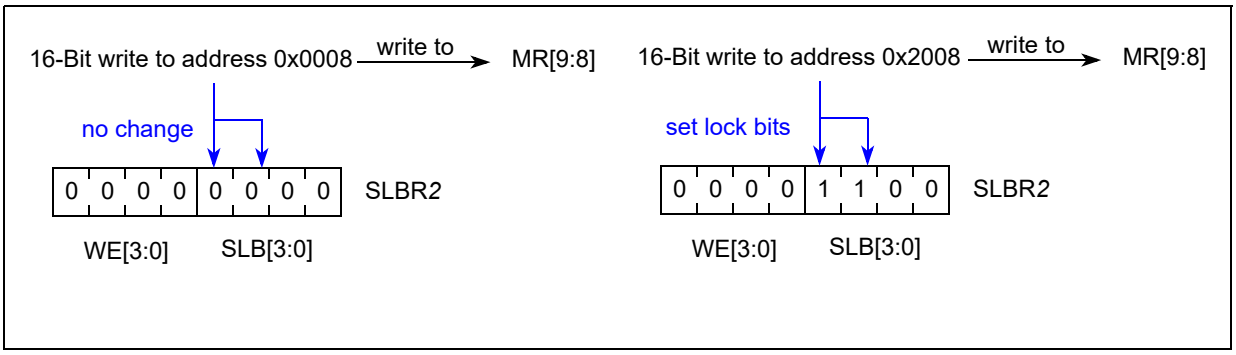


The data written to SLBR*n*.SLB[0] is mirrored to SLBR*n*.SLB[1] as the corresponding register is 16-bit protected. The data written to SLBR*n*.SLB[2] is blocked as the corresponding register is unprotected. The data written to SLBR*n*.SLB[3] is written to SLBR*n*.SLB[3].

67.4.2.2 Enable locking via mirror module space (Area #3)

It is possible to enable locking for a register after writing to it. To do so the mirrored module address space must be used. *Figure 1412* shows one example:

Figure 1412. Enable Locking Via Mirror Module Space (Area #3)

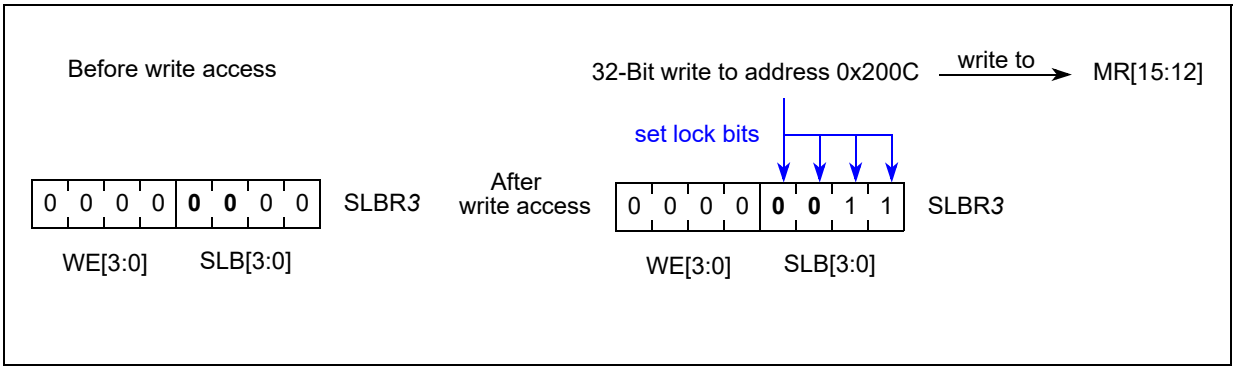


When writing to address 0x0008 the registers MR9 and MR8 in the protected module are updated. The corresponding lock bits remain unchanged (left part of *Figure 1409*).

When writing to address 0x2008 the registers MR9 and MR8 in the protected module are updated. The corresponding lock bits SLBR2.SLB[1:0] are set while the lock bits SLBR2.SLB[3:2] remain unchanged (right part of *Figure 1409*).

Figure 1413 shows an example where some addresses are protected and some are not:

Figure 1413. Enable locking for protected and unprotected addresses



In the example in *Figure 1413* addresses 0x0C and 0x0D are unprotected. Therefore their corresponding lock bits SLBR3.SLB[1:0] are always 0b0 (shown in bold). When doing a 32-bit write access to address 0x200C only lock bits SLBR3.SLB[3:2] are set while bits SLBR3.SLB[1:0] stay 0b0.

Note: Lock bits can only be set via writes to the mirror module space. Reads from the mirror module space will not change the lock bits.

67.4.2.3 Write protection for locking bits

Changing the locking bits through any of the procedures mentioned in *Section 67.4.2.1* and *Section 67.4.2.2* is only possible as long as the bit GCR.HLB is cleared. Once this bit is set the locking bits can no longer be modified until there is a system reset.

67.4.3 Access errors

The protection module generates transfer errors under several circumstances. For the area definition refer to *Figure 1405*

1. If accessing area #1 or area #3, the protection module will pass on any access error from the underlying Module under Protection.
2. If user mode is not allowed, user writes to all areas will assert a transfer error and the writes will be blocked.
3. If accessing the reserved area #2, a transfer error will be asserted.
4. If accessing unimplemented 32-bit registers in area #4 and area #5 a transfer error will be asserted.
5. If writing to a register in area #1 and area #3 with Soft Lock Bit set for any of the affected bytes a transfer error is asserted and the write will be blocked. Also the complete write operation to non-protected bytes in this word is ignored.
6. If writing to a Soft Lock Register in area #4 with the Hard Lock Bit being set a transfer error is asserted.
7. Any write operation in any access mode to area #3 while Hard Lock Bit GCR.HLB is set

67.5 Initialization/application information

67.5.1 Reset

The reset state of each individual bit is shown within the Register Description section (See *Section 67.3.2*). In summary, after reset, locking for all MR*n* registers is disabled. The registers can be accessed in Supervisor Mode only.

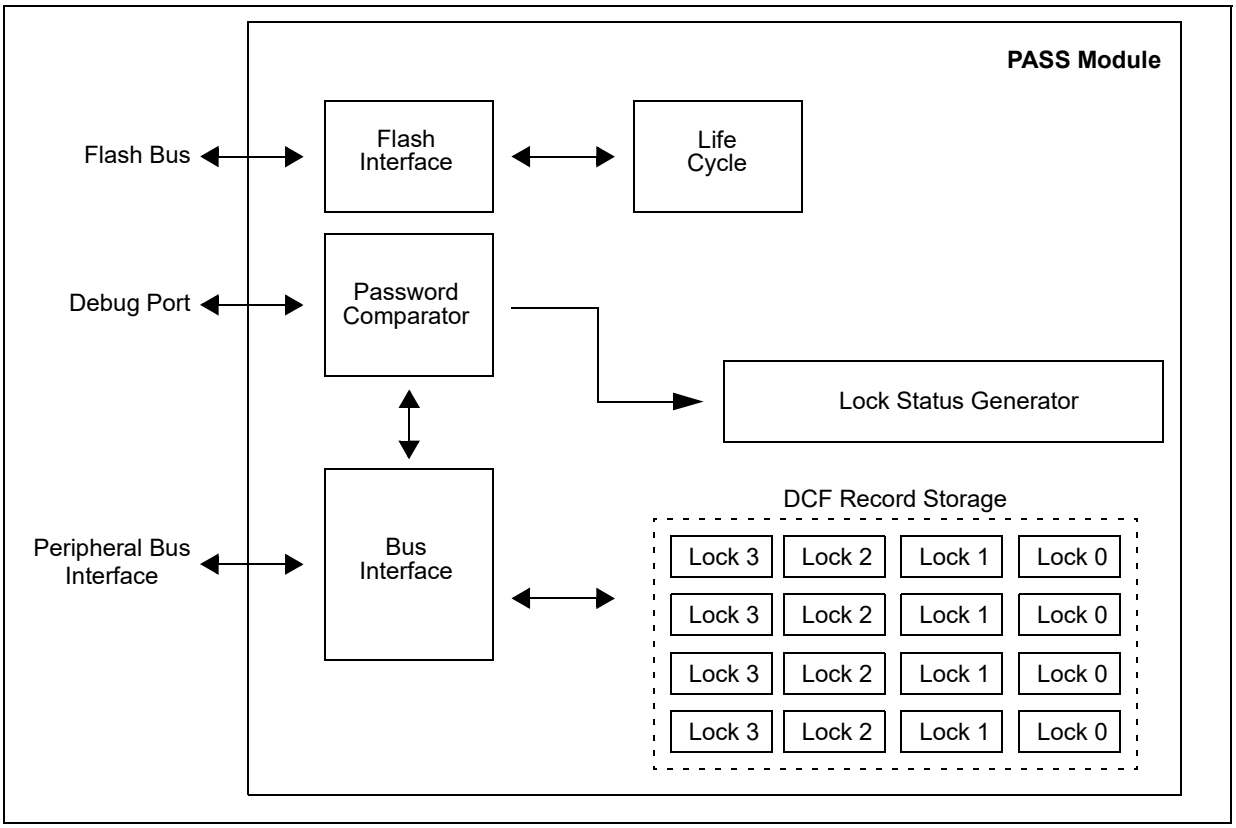
68 Password and Device Security Module (PASS)

68.1 Introduction

68.1.1 Overview

The Password and device security (PASS) module shown in *Figure 1414*, receives password challenges and determines their validity. It also maintains the device security and access states.

Figure 1414. PASS module block diagram



68.1.2 Features

The PASS includes these distinctive features:

- Lifecycle status
- Password comparison
- JTAG password comparison
- Retest Password Comparison

68.1.3 Modes of operation

The PASS operates identically in all system modes.

68.2 Memory map and register definition

This section provides a detailed description of all memory-mapped registers in the PASS. (DCF clients are described in *Section 68.3: DCF clients*.)

Table 1357 shows the memory map for the PASS. Note that all addresses are offsets: the absolute address for registers may be calculated by adding the specified offset to the base address of the PASS. The address of the DCF client is local to the PASS module and must be qualified with the appropriate Client Select (CS[14:0]) for the PASS module, as defined in the DCF Record Section.

Table 1357. PASS memory map

Offset	Register	Access	Reset Value	DCF Client Address(1) (ADDR[14:0])	Section
0x0000	Lifecycle Status Register (LCSTAT)	R	0xX0000007	None	*Section 68.2.1.1*
0x0008	Challenge Selector Register (CHSEL)	R/W	0x00000000	None	*Section 68.2.1.2*
0x0010	Challenge Status Register (CSTAT)	R	0x00000000	None	*Section 68.2.1.3*
0x0014-0x001F	Reserved				
0x0020	Challenge Input Register 0 (CIN0)	W	0x00000000	None	*Section 68.2.1.4*
0x0024	Challenge Input Register 1 (CIN1)	W	0x00000000	None	*Section 68.2.1.4*
0x0028	Challenge Input Register 2 (CIN2)	W	0x00000000	None	*Section 68.2.1.4*
0x002C	Challenge Input Register 3 (CIN3)	W	0x00000000	None	*Section 68.2.1.4*
0x0030	Challenge Input Register 4 (CIN4)	W	0x00000000	None	*Section 68.2.1.4*
0x0034	Challenge Input Register 5 (CIN5)	W	0x00000000	None	*Section 68.2.1.4*
0x0038	Challenge Input Register 6 (CIN6)	W	0x00000000	None	*Section 68.2.1.4*
0x003C	Challenge Input Register 7 (CIN7)	W	0x00000000	None	*Section 68.2.1.4*
0x0040-0x00AF	Reserved				
None	Censorship	None	0x00000000	0x00B0	*Section 68.3.1*
0x00C4-0x00DF	Reserved				
0x00E0-0x00FF	Reserved				
Password Group 0					
0x0100	Lock 0 Status Register (LOCK0_PG0)	R/W	0xXXXXXXXX(2)	0x0100	*Section 68.2.1.5*
0x0104	Lock 1 Status Register (LOCK1_PG0)	R/W	0x0000XXXX(2)	0x0104	*Section 68.2.1.6*
0x0108	Lock 2 Status Register (LOCK2_PG0)	R/W	0xXXXXXXXX(2)	0x0108	*Section 68.2.1.7*
0x010C	Lock 3 Status Register (LOCK3_PG0)	R/W	0xXFXXXXXX(2)	0x010C	*Section 68.2.1.8*

Table 1357. PASS memory map (continued)

Offset	Register	Access	Reset Value	DCF Client Address(1) (ADDR[14:0])	Section
Password Group 1					
0x0110	Lock 0 Status Register (LOCK0_PG1)	R/W	0xXXXXXXXX(2)	0x0110	*Section 68.2.1.5*
0x0114	Lock 1 Status Register (LOCK1_PG1)	R/W	0x0000XXXX(2)	0x0114	*Section 68.2.1.6*
0x0118	Lock 2 Status Register (LOCK2_PG1)	R/W	0xXXXXXXXX(2)	0x0118	*Section 68.2.1.7*
0x011C	Lock 3 Status Register (LOCK3_PG1)	R/W	0xXFXXXXXX(2)	0x011C	*Section 68.2.1.8*
Password Group 2					
0x0120	Lock 0 Status Register (LOCK0_PG2)	R/W	0xXXXXXXXX(2)	0x0120	*Section 68.2.1.5*
0x0124	Lock 1 Status Register (LOCK1_PG2)	R/W	0x0000XXXX(2)	0x0124	*Section 68.2.1.6*
0x0128	Lock 2 Status Register (LOCK2_PG2)	R/W	0xXXXXXXXX(2)	0x0128	*Section 68.2.1.7*
0x012C	Lock 3 Status Register (LOCK3_PG2)	R/W	0xXFXXXXXX(2)	0x012C	*Section 68.2.1.8*
Password Group 3					
0x0130	Lock 0 Status Register (LOCK0_PG3)	R/W	0xXXXXXXXX(2)	0x0130	*Section 68.2.1.5*
0x0134	Lock 1 Status Register (LOCK1_PG3)	R/W	0x0000XXXX(2)	0x0134	*Section 68.2.1.6*
0x0138	Lock 2 Status Register (LOCK2_PG3)	R/W	0xXXXXXXXX(2)	0x0138	*Section 68.2.1.7*
0x013C	Lock 3 Status Register (LOCK3_PG3)	R/W	0xXFXXXXXX(2)	0x013C	*Section 68.2.1.8*

1. DCF Client Address column shows what is the DCF used to pre-load the register value during reset. If the column value is "None", it means that the register value at reset is fixed in hardware.
2. Reset value depends on password group lock state

Note: Non-implemented registers generate bus aborts if enabled by the SOC.

68.2.1 Register descriptions

The following registers are available in the PASS. The shaded bits are reserved for future use. For future compatibility, only write ‘0’ to these bits; they always read ‘0’.

68.2.1.1 Lifecycle Status register (LCSTAT)

The Lifecycle Status Register reflects the production status and censorship status of the device. It is possible to mature the device (move the device lifecycle forward), but never to revert the lifecycle to an earlier state.

The reset value of the LCSTAT register LIFE field derives from an external lifecycle management module (like SSCM). It is a 3-bit lifecycle. This register simply displays the lifecycle input from external blocks.

Offset: 0x0000 Access: Software Read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	CNS	JUN	1	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	x(2)	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	LIFE		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1

2. Reset value depends on the Censorship DCF client

Figure 1415. Lifecycle Status register (LCSTAT)

PASS_LCSTAT register functions are as shown in *Table 1358*.

Table 1358. LCSTAT register field descriptions

Field	Description
0 CNS	This field indicates the state of the Censorship DCF Client. The bit can be set (or cleared) via DCF Record in UTEST Flash. See *Section 68.5.3: Censoring and uncensoring the device*. 0 Device is uncensored 1 Device is censored
1 JUN	This field indicates whether the device has been temporarily uncensored. Successful transmission of the JTAG password sets this bit to 1, otherwise it is 0. Unsuccessful transmission clears this bit. **Note:** Refer to the Security Reference Manual for details.
29:31 LIFE	The lifecycle setting is manipulated externally (SSCM). Pass displays a 3-bit input lifecycle. 000 Failure Analysis mode 001 Reserved 010 OEM Production 011 Customer Delivery 110 ST Production 111 In Field

68.2.1.2 Challenge Selector register (CHSEL)

This register determines which password group is challenged via the Challenge Input Register.

When a master accesses this register, the PASS module stores the number of the master in the LOCK3_PGn register of the selected passgroup and sets the PGL bit in that register.

The ID of the accessing master sets the CMST field of the Challenge Status register.

Offset: 0x0008 Access: Software Read/Write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	GRP	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1416. CHSEL register

CHSEL register functions are as shown in *Table 1359*.

Table 1359. CHSEL register field descriptions

Field	Description
30:31 GRP	Password Group to unlock 00 Password Group 0 01 Password Group 1 10 Password Group 2 11 Password Group 3

68.2.1.3 Challenge Status register (CSTAT)

This register provides status information for the password challenge.

Offset: 0x0010 Access: Software Read

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	CMST			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1417. Challenge Status register (CSTAT)

CSTAT register functions are as shown in *Table 1360*.

Table 1360. CSTAT register field descriptions

Field	Description
28:31 CMST	ID of the master which has last written the CHSEL register

Only the master shown in this register can access the CINn registers—access by other masters result in a comparison fail regardless of the values written.

68.2.1.4 Challenge Input register (CIN*n*)

This register is used to provide the Challenge word *n* for a password group as selected by the Challenge Selector Register.

Valid password criteria is that it should not be all Zeros or all Ones

Offset: 0x0020 + *n**0x4 (*n* = 0 to 7) Access: Software Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W	PW32															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0															
W	PW32															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1418. Challenge Input register (CIN*n*)

CIN register functions are as shown in *Table 1361*.

Table 1361. CIN*n* register field descriptions

Field	Description
0:31 PW32	32 bits of the 256-bit password challenge. Only 32-bit writes may be used.

After programming the Challenge Selector Register, the password challenge needs to be written with eight 32-bit writes from CIN0 to CIN7—in that order. The most significant 32 bits of the password challenge must be written to CIN0. Once CIN7 is written, the password is compared. The write access only completes after the comparison has been executed.

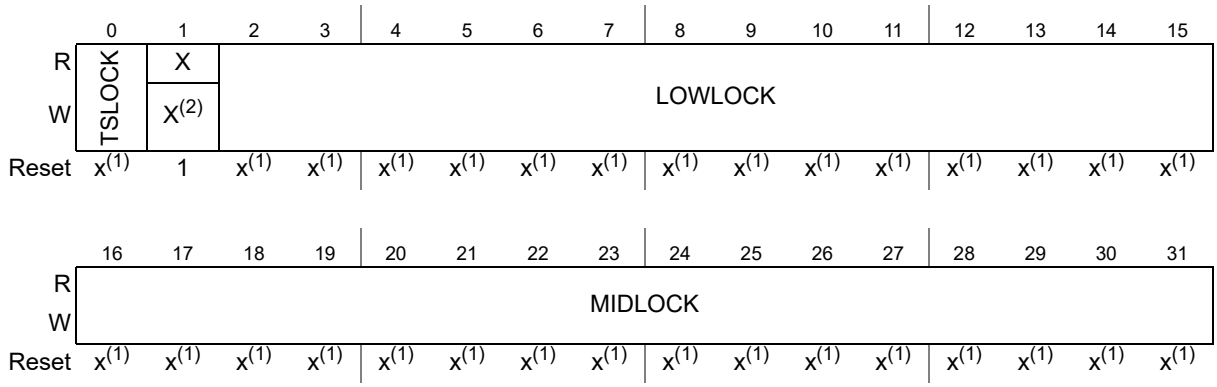
All accesses to these registers need to be made by the bus master which wrote the CHSEL register previously. If another master writes any of the CIN registers before the original master has written CIN7, the comparison fails.

68.2.1.5 Password Group *n* - Lock 0 Status register (LOCK0_PG*n*)

Each Password Group has one Lock 0 Status register. This register shows the current lock state for the Flash blocks. Each time a password group is unlocked the register becomes writable. For additional details of the LOCK bits, refer to the Flash memory description.

The LOCK bits do not come into effect before the lifecycle has matured to *OEM Production* or older.

Offset: 0x0100 + *n**0x10 (*n* = 0 to 3) Access: Software Read/Write & DCF Record Write



1. If no DCF records have been written to change the lock state, the bit will reset to '1'.
2. This bit is not implemented and its value has no impact on the system.

Figure 1419. Password Group *n* - Lock 0 Status register (LOCK0_PG*n*)

LOCK0_PG*n* register functions are as shown in *Table 1362*.

Table 1362. LOCK0_PG*n* register field descriptions

Field	Description
0 TSLOCK	UTEST NVM Lock This bit is used to lock the UTEST NVM block from programs (erase protection not needed since UTEST NVM is OTP and not erasable).
2:15 LOWLOCK	Low Block Lock A value of 1 in a bit of the lock register signifies that the corresponding block is locked for program and erase.
16:31 MIDLOCK	Mid Block Lock A value of 1 in a bit of the lock register signifies that the corresponding block is locked for program and erase.

Note: Lock registers can only be accessed with 32-bit operations.

The initial value of the lock bits is set via DCF records stored in UTEST Flash block and copied over during reset. The default value before any DCF record is written is '1', indicating that a block is locked.

Warning: Failure to program valid passwords or DCF records for all password groups before the lifecycle matures to OEM Production renders the part unusable.

Once a password group is unlocked, software can set or clear any of the lock bits in the group, by writing to the registers.

The resulting lock status of a Flash block is determined by the combination of the lock bits of all password groups. If a block is locked in multiple groups, then all lock bits for the block need to be cleared before program and erase is possible. However, unless the lifecycle is

older than *OEM Production*, the Flash blocks are always unlocked regardless of the values in the password groups.

68.2.1.6 Password Group *n* - Lock 1 status register (LOCK1_PG*n*)

Each Password Group has one Lock 1 Status Register. This register shows the current lock state for the Flash blocks. Each time a password group is unlocked, the register becomes writable. For additional details of the LOCK bits, refer to the Flash description.

The LOCK bits do not come into effect before the lifecycle matures to *OEM Production* or older.

Offset: 0x0104 + *n**0x10 (*n* = 0 to 3) Access: Software Read/Write & DCF Record Write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	HIGHLOCK															
W																
Reset	x(1)	x(1)	x(1)	x(1)	x(1)	x(1)	x(1)	x(1)	x(1)	x(1)	x(1)	x(1)	x(1)	x(1)	x(1)	x(1)

1. If no DCF records have been written to change the lock state, the bit will reset to ‘1’.

Figure 1420. Password Group *n* - Lock 1 status register (LOCK1_PG*n*)

LOCK1_PG*n* register functions are as shown in *Table 1363*.

Table 1363. LOCK1_PG*n* register field descriptions

Field	Description
16:31 HIGHLOCK	High Block Lock A value of 1 in a bit of the lock register signifies that the corresponding block is locked for program and erase.

Note: Lock registers can only be accessed with 32-bit operations.

The initial value of the lock bits is set via DCF records stored in UTEST Flash block and copied over during reset. The default value before any DCF record is written, is ‘1’, which indicates a block is locked.

Warning: Failure to program valid passwords or DCF records for all password groups before the lifecycle matures to OEM Production renders the part unusable. However, additional DCF Records can be written to UTEST to change the initial value of this register.

Once a password group is unlocked, software can set or clear any of the lock bits in the group by writing to the registers.

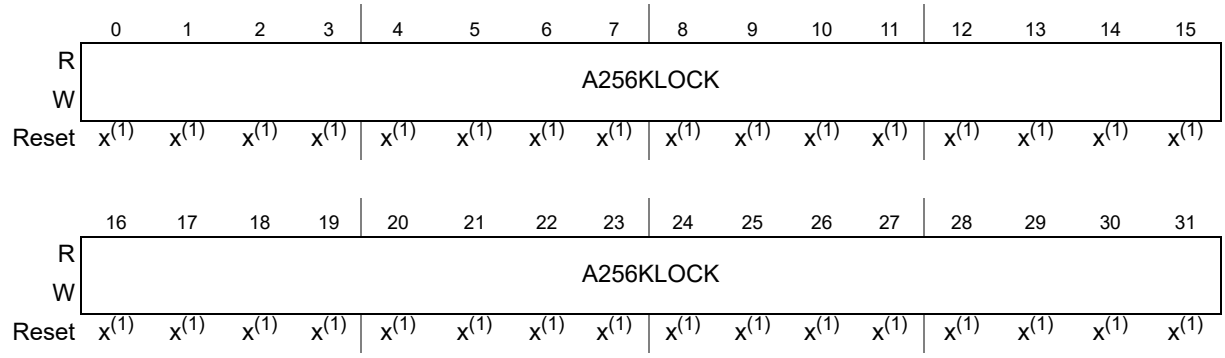
The resulting lock status of a Flash block is determined by the combination of the lock bits of all password groups. If a block is locked in multiple groups, then all lock bits for the block need to be cleared before program and erase is possible. However, unless the lifecycle is older than *OEM Production*, the Flash blocks are always unlocked regardless of the values in the password groups.

68.2.1.7 Password Group *n* - Lock 2 status register (LOCK2_PG*n*)

Each Password Group has one Lock 2 Status Register. This register shows the current lock state for the Flash blocks. Each time a password group is unlocked, the register becomes writable. For additional details of the LOCK bits, refer to the Flash description.

The LOCK bits do not come into effect before the lifecycle is matured to *OEM Production* or older.

Offset: 0x0108 + *n**0x10 (*n* = 0 to 3) Access: Software Read/Write & DCF Record Write



1. If no DCF records have been written to change the lock state, the bit resets to '1'.

Figure 1421. Password Group *n* - Lock 2 status register (LOCK2_PG*n*)

Note: Lock registers can only be accessed with 32-bit operations.

LOCK2_PG*n* register functions are as shown in *Table 1364*.

Table 1364. LOCK2_PG*n* register field descriptions

Field	Description
0:31 A256KLOCK	256K address space block lock A256KLOCK[31:0] A value of 1 in a bit of the lock register signifies that the corresponding block is locked for program and erase.

The initial value of the lock bits is set via DCF records stored in UTEST Flash block and copied over during reset. The default value before any DCF record is written is '1', which indicates a block is locked.

Warning: Failure to program valid passwords or DCF records for all password groups before the lifecycle matures to OEM Production renders the part unusable. However, additional DCF Records can be written to UTEST to change the initial value of this register.

Once a password group is unlocked, software can set or clear any of the lock bits in the group, by writing to the registers.

The resulting lock status of a Flash block is determined by the combination of the lock bits of all password groups. If a block is locked in multiple groups, then all lock bits for the block need to be cleared before program and erase is possible.. However, unless the lifecycle is older than *OEM Production*, the Flash blocks are always unlocked regardless of the values in the password groups.

68.2.1.8 Password Group *n* - Lock 3 status register (LOCK3_PG*n*)

Each Password Group has one Lock 3 Status Register. This register shows the current lock state for the Flash blocks. Each time a password group is unlocked, the PGL bit is cleared and the Lock registers become writable.

The LOCK bits do not come into effect before the lifecycle matures to *OEM Production* or older.

Offset: 0x010C + *n**0x10 (*n* = 0 to 3) Access: Software Read/Write & DCF Record Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	PGL	DBL	MO	0	MSTR				0	0	0	RL4(1)	RL3(1)	RL2(1)	RL1(1)	RL0(1)
W																
Reset	x(2)	x(2)	x(2)	0	1	1	1	1	0	0	0	x(2)	x(2)	x(2)	x(2)	x(2)

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	A256KLOCK(1)															
W																
Reset	x(3)	x(3)	x(3)	x(3)	x(3)	x(3)	x(3)	x(3)	x(3)	x(3)	x(3)	x(3)	x(3)	x(3)	x(3)	x(3)

1. Not all devices include this field. See the device reference manual's chip configuration chapter for details.
2. Reset value depends on lifecycle. If lifecycle is *Customer delivery* or earlier, the value will be 0, otherwise 1.
3. If no DCF records have been written to change the lock state, the bit will reset to '1'.

Figure 1422. Password Group *n* - Lock 3 status register (LOCK3_PG*n*)

LOCK3_PG*n* register functions are as shown in *Table 1365*.

Table 1365. LOCK3_PG*n* register field descriptions

Field	Description
0 PGL	Password Group Lock This bit may be set in software to lock the password group, when the device is older than *Customer Delivery*. Before that the value of the bit is always 0. This bit is only updatable from SW path. It cannot be updated from DCF loading from Flash. This bit can be cleared by writing a valid password to the Password Challenge Input Registers. 0 Password group registers are unlocked. All four registers in the Password group may be read and written without restriction. 1 Password group registers are locked. Writes to the password group registers have no effect. Read accesses work normally. (If lifecycle is *Customer Delivery* or earlier, this bit has no effect and the registers are always unlocked.)
1 DBL	Debug Interface Lock The default value of this bit would be '1' if not updated from flash. 0 Debug interface is unlocked. Accessed to all JTAG clients is allowed. 1 Debug interface is locked. Only access to JTAG password challenge register is possible. Access to all other JTAG clients is blocked.
2 MO	Master Only 0 All masters can modify the passgroup settings if PGL is cleared. 1 Only the Master which unlocked the passgroup can change the passgroup settings.
4:7 MSTR	Master Access Only the Master with this matching master ID can change the passgroup settings. This field is loaded with the ID of the master which unlocked the passgroup once the correct password is provided.
11 RL4	Read Lock Region 4 lock bit The default value of this bit would be '1' if not updated from flash. 0 Region 4 is not protected. 1 Region 4 is protected.
12 RL3	Read Lock Region 3 lock bit The default value of this bit would be '1' if not updated from flash. 0 Region 3 is not protected. 1 Region 3 is protected.
13 RL2	Read Lock Region 2 lock bit The default value of this bit would be '1' if not updated from flash. 0 Region 2 is not protected. 1 Region 2 is protected.
14 RL1	Read Lock Region 1 lock bit The default value of this bit would be '1' if not updated from flash. 0 Region 1 is not protected. 1 Region 1 is protected.

Table 1365. LOCK3_PG*n* register field descriptions (continued)

Field	Description
15 RL0	Read Lock Region 0 lock bit The default value of this bit would be '1' if not updated from flash. 0 Region 0 is not protected. 1 Region 0 is protected.
16:31 A256KLOCK	256K address space block lock A256KLOCK[47:32] A value of 1 in a bit of the lock register signifies that the corresponding block is locked for program and erase.

Note: Lock registers can only be accessed with 32-bit operations.

The initial value of the lock bits is set via DCF records stored in UTEST Flash block and copied over during reset. The default value before any DCF record is written is '1', which indicates a block is locked.

Warning: Failure to program valid passwords or DCF records for all password groups before the lifecycle matures to OEM Production renders the part unusable. However, additional DCF Records can be written to UTEST to change the initial value of this register.

Once a password group is unlocked, software can set or clear any of the lock bits in the group, by writing to the registers.

The resulting lock status of a Flash block is determined by the combination of the lock bits of all password groups. If a block is locked in multiple groups, then all lock bits for the block need to be cleared before program and erase is possible.. However, unless the lifecycle is older than *OEM Production*, the Flash blocks are always unlocked regardless of the values in the password groups.

For operation of the region read protect feature, refer to *Section 68.4.3: Flash Memory Read Protection*.

Once a censored device enters the *Failure Analysis* lifecycle, read protection is active regardless whether the system is in debug mode or not. It is still possible to unlock Flash regions with the correct passwords, however. .

68.3 DCF clients

Where a DCF client is listed, the reset value of that register may automatically be pre-loaded during reset based on the data stored in DCF Records in Flash UTEST region.

The address of the DCF client is local to the PASS module and must be qualified with the appropriate Client Select (CS[14:0]) for the PASS module, as defined in the DCF Record Section.

68.3.1 Censorship

Censorship is a DCF client that is *not* accessible by software. It is automatically pre-loaded during reset with data stored in DCF Record format in the Flash UTEST region. The Censorship DCF client defines whether the MCU is censored or not (see *Table 1357*).

DCF Client Address: 0x00B0 Censorship — Access: DCF Record Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R																
W																
Reset	x(1)	x(1)	x(1)	x(1)	x(1)	x(1)	x(1)	x(1)	x(1)	x(1)	x(1)	x(1)	x(1)	x(1)	x(1)	x(1)

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R																
W	x(2)	x(2)	x(2)	x(2)	x(2)	x(2)	x(2)	x(2)	x(2)	x(2)	x(2)	x(2)	x(2)	x(2)	x(2)	x(2)
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. This bit is not defined.
2. This value is loaded by DCF Record during reset.

Figure 1423. Censorship register (not accessible)

Table 1366. Censorship register field descriptions

Field	Description
16:31	Censorship If this value is 0x55AA the MCU is NOT censored. If this value is anything other that 0x55AA, the MCU is censored.

68.4 Functional description

The primary purpose of the PASS is to provide fine-grained access control to the Flash and debugger interface. It utilizes the DCF mechanism, as described in the SSCM section.

68.4.1 Censorship

Censorship is the overall control that enables and disables two independent security features: Debug Interface access; and flash memory read protection.

The MCU can be Censored by writing a DCF Record to flash that writes to the Censorship DCF client (See *Section 68.3.1: Censorship*).

68.4.2 Debug Interface Access

The debug interface in fully enabled in Customer_Delivery Lifecycle. But from OEM_Production lifecycle, it can be blocked using censorship and selectively by either entering a 256 bits password via JTAG or by software clearing the DBL bit the PASS_LOCK3 register (See *68.2.1.8: Password Group n - Lock 3 status register*

(LOCK3_PGn)). Both of these debug enabling options are further conditioned by lifecycle and Production Disable.

The table below describes all options for enabling/blocking Debug Interface access.

Table 1367. Debug Interface Enable Truth Table

Input Conditions				Outcome
Lifecycle	**Censorship**	**Debug Enable (LOCK3[DBL] bit)**	**JTAG Password**	**Debug Interface Enable**
Customer Delivery	Don't Care	Don't Care	Don't Care	Enabled
OEM Production	Not censored	Don't Care	Don't Care	Enabled
	Censored	Unlocked (DBL=0)	Don't Care	Enabled
		Locked (DBL=1)	Matched	Enabled
			Not Matched	Blocked
In Field	Not censored	Don't Care	Don't Care	Enabled
	Censored	Unlocked (DBL=0)	Don't Care	Enabled
		Locked (DBL=1)	Matched	Enabled
			Not Matched	Blocked
Failure Analysis	Don't Care	Don't Care	Don't Care	Enabled

If the device is censored, then access is possible via JTAG password.

68.4.2.1 JTAG Password Challenge

To enable the Debug Interface Access, the JTAG password challenge register in the JTAGC, can to be written by an external tool. If that password matches the JTAG password stored in UTEST flash, Debug Interface Access is enabled (according to the table above) until the next destructive reset.

68.4.3 Flash Memory Read Protection

Flash Memory Read Protection is a security feature that can selectively make regions of the flash memory unreadable. Any attempt to read a region of flash that is read protected results in the read access terminating with a transfer error. This is consistent for all bus masters including debug bus masters.

The assertion of flash memory region read protection is controlled by Lifecycle, Censorship and whether the Debug Interface access is enabled or blocked. *Table 1368: Flash Memory Read Protection Truth Table* describes the combination of conditions that drive Flash Memory Read Protection.

The intention of the feature is to protect the contents of the flash from being read. The flash memory is readable in normal operation, allowing the application to run, but becomes read protected when a Debug tool is attached and enabled (JTAG password or LOCK3[DBL] bit).

The Flash memory is also read protected when the lifecycle status is *Failure_Analysis*. This lifecycle is used for a module field failure, where the device is returned for failure analysis. This protection prevents the contents of the flash memory from being read.

The flash memory is divided into a number of Regions (refer to *Chapter 7: Device configuration*, *Section 7.11.5: Password and Device Security Module (PASS) configuration*).

Table 1368. Flash Memory Read Protection Truth Table

INPUTS			OUTPUTS	
Lifecycle	**Censorship**	**Lock3[RLx]**	**Read Protected RL[0] (UTEST)**	**Read Protected RL[2:1] (Other Regions)**
STMicroelectronics Production / Customer Delivery	Don't Care	Don't Care	Readable	Readable
OEM Production / IN Field	Uncensored	Don't Care	Readable	Readable
	Censored	Don't Care	Readable	Readable
		Locked (1)	Read Blocked	Read Blocked
		Unlocked (0)	Readable	Readable
Failure Analysis	Uncensored	Don't Care	Readable	Readable

68.4.4 UTEST protection

During the *Customer Delivery* lifecycle, the user must program a valid password or a DCF record that writes LOCKm_PGn registers to unprotect the UTEST for each password group in order to prevent permanent and irreversible locking of Flash blocks.

68.4.5 Retest Password

Retest is the capability for STMicroelectronics to retest devices that have been returned to STMicroelectronics after being shipped to customers. Virgin devices are tested at STMicroelectronics in STMicroelectronics_Production lifecycle, the lifecyle is advanced to Customer_Delivery before shipping to the customer. Retest Password provides additional protection from entering Test mode while in Customer_Delivery lifecycle

68.4.5.1 Retest Password Implementation

The Retest Password is a password entered by the tester during retest to gain access to Flash Test Mode.

Note: This password is not required during normal production test until the lifecycle is advanced to Customer Delivery.

68.4.5.2 Retest Password Storage

The Retest password is 256 bits wide and is stored in Test Flash

68.4.5.3 Retest Password Challenge Entry

The Retest password challenge is entered via JTAG using the password register in the JTAGC.

JTAG password is not used in *Customer Delivery* lifecycle, so when the PASS module receives a request to perform a password compare from JTAGC while lifecycle = *Customer Delivery*, the password challenge is compared with Retest password.

68.4.5.4 Retest Password Compare

The Retest Password challenged, is compared against the password stored in Test Flash in the same way as in the PASS module for JTAG password. If the password matches, access to Flash Test Mode is granted and stays granted until the next destructive reset.

There are no passwords that the hardware detects as illegal. All 0's and all 1's are valid passwords.

68.5 Initialization/application information

68.5.1 Reset

The reset state of each individual bit is shown within the Register Description section (see *Section 68.2.1*). However, many bits receive their values based on DCF records stored in the UTEST Flash memory block.

68.5.2 Setting lock bits in a password group

There is no restriction on setting and clearing lock registers in the UTEST Flash, apart from the available space for DCF records. It is possible to set and clear them multiple times.

Table 1369. CS/ADDR Values for Lock Registers

Register	DCF CS/ADDR
PW Group 0	
Lock0	0xXXXX0100(1)
Lock1	0xXXXX0104(1)
Lock2	0xXXXX0108(1)
Lock3	0xXXXX010C(1)
PW Group 1	
Lock0	0xXXXX0110(1)
Lock1	0xXXXX0114(1)
Lock2	0xXXXX0118(1)
Lock3	0xXXXX011C(1)
PW Group 2	
Lock0	0xXXXX0120(1)
Lock1	0xXXXX0124(1)
Lock2	0xXXXX0128(1)
Lock3	0xXXXX012C(1)

Table 1369. CS/ADDR Values for Lock Registers (continued)

Register	DCF CS/ADDR
PW Group3	
Lock0	0xXXXX0130(1)
Lock1	0xXXXX0134(1)
Lock2	0xXXXX0138(1)
Lock3	0xXXXX013C(1)

1. Value depends on the DCF Client Select assigned to the PASS module in the device.

In order to set the MIDLOCK field in the Lock0 register of password group 1 to the value 0x00FF and other fields to 0, the DCF record has to be added as shown in *Table 1370*.

Table 1370. Setting some bits in the MIDLOCK field

ADDR offset	DATA	Description
0x00	0x05AA55AF	Start Record
0x04	0x00000000	
0x08	0x000000FF	PW Group 1, Lock0 Set MIDLOCK to 0xFF
0x0C	0xXXXX0110(1)	
0x10	0xFFFFFFFF	Stop Record
0x14	0xFFFFFFFF	
...	...	

1. Value depends on the DCF Client Select assigned to the PASS module in the device.

68.5.3 Censoring and uncensoring the device

There is no restriction on setting and clearing the censorship status in the UTEST Flash, apart from the available space for DCF records. It is possible to censor and un-censor multiple times.

The DCF Client Select / Address for the Censorship DCF client, is a concatenation of the DCF Client Select (assigned outside of PASS) and the address 0x00B0. Therefore 0xXXXX00B0. In order to un-censor the device, the data field of the record needs to be written with 0x----55AA (upper 16 bits are not compared). Any other value censors the device.

On delivery the device is censored.

Appendix A Acronyms and abbreviations

Table 1371 lists acronyms and abbreviations used in this document.

Table 1371. Acronyms and abbreviations

Term	Meaning
AC	Alternative current
ACIM	AC induction motor
AHB	Advanced high-performance bus
AIC	AIPS integrity checker
AIPS	AHB to IPS bridge
AMBA	Advanced microcontroller bus architecture
APU	– Advanced protocol unit – Auxiliary processing unit
ASILD	Automotive SIL D
AUTOSAR	Automotive open system architecture
BAF	Boot assist flash
BDC	Brush DC
BIST	Built-in self test
BIU	Bus interface unit
BLDC	Brushless DC
CAN	Controller area network
CF	Critical fault
CGM	Clock generation module
CMU	Clock monitoring unit
CPU	Central processing unit
CQM	Clock quality monitor
CRC	Cyclic redundancy check
CSI	Combined serial interface. DSPI configuration that alternates DSI and SPI frames.
CTU	Cross triggering unit
DC	Direct current
DCF	Device configuration format
DMA	Direct memory access
DMAMUX	Direct memory access controller multiplexer
DSPI	Deserial serial peripheral interface
DSI	Deserial serial interface. DSPI configuration that serializes and deserializes registers or signals to allow for pin reduction.
ECC	Error correction code

Table 1371. Acronyms and abbreviations (continued)

Term	Meaning
ECSM	Error correction status module
EIM	Error injection module
EMI	Electromagnetic interference
EOUT	Error out
ESD	Electromagnetic static discharge
FCCU	Fault collection and control unit
FMPLL	Frequency-modulated phase-locked loop
FOC	Field oriented control
FOSU	FCCU output supervision unit
FPEC	Flash program erase controller
FSM	Finite state machine
FTTI	Fault tolerant time interval
GPIO	General-purpose I/O
HVD	High voltage detector
HW	Hardware
ID	LIN identifier field
IEEE	Institute of electrical and electronics engineers
INTC	Interrupt controller
INTF	Interface
IP	Intellectual propriety
IPS	– Internal peripheral system – IPI slave bus
IRQ	Interrupt request
ISR	Interrupt service routine
JDC	JTAG data communication
JEDEC	Joint electron device engineering council
JTAG	Joint test action group
LBIST	Logic built-in self test
LIN	Local interconnect network
LINFlexD	Local interconnect network controller
LVD	Low voltage detector
LWB	Late-write buffer
MB	Message buffer
MBIST	Memory built-in self test
MBM	Message buffer management

Table 1371. Acronyms and abbreviations (continued)

Term	Meaning
ME	Mode entry module (MC_ME)
MEMU	Memory error management unit
MFD	Multiplication factor divider
Mux	Multiplex
NCF	Non-critical fault
NMI	Non-maskable interrupts
NVM	Non-volatile memory
OPP	Over-program protection
OSC	Oscillator
OTP	One time programmable
PASS	Password and device security module
PBRIDGE	Peripheral bridge
PCM	Platform configuration module
PCU	Power control unit
PFLASH	Platform flash controller
PIT	Periodic interrupt timer
PLL	Phase-locked loop
PMAC	Permanent magnet AC
PMC	Power management controller
PRAM	Platform RAM controller
PWD	Password
PWM	Pulse width modulation / modulator
RC	– Resistance-capacitance – Resistor-capacitor
RCC	Redundancy control checkers
RCCU	Redundancy control checker unit
RCOSC	RC oscillator
RF	Recoverable fault
RFD	Reduced frequency divider
RGM	Reset generation module
RMW	Read-modify-write
RTL	Register transfer language
RWW	Read-while-write
RX	Receive
SAG	Safety application guide

Table 1371. Acronyms and abbreviations (continued)

Term	Meaning
SAR ADC	Successive approximation register analog-to-digital converter
SIL	Safety integrity level
SIUL	System integration unit lite
SMPU	System memory protection unit
SoC	System-on-chip
SOR	Sphere of redundancy
SPI	Serial peripheral interface
SRM	Switched reluctance motor
SSCM	System status and configuration module
STCU	Self-test control unit
STM	System timer module
SW	Software
SWT	Software watchdog timer
TBD	To be defined
TCD	Transfer control descriptor
TSENS	Temperature sensor
TX	Transmit
UART	Universal asynchronous/synchronous receiver transmitter
UF	Unrecoverable fault
VCO	Voltage controlled oscillator
VREG	Voltage regulator
VRM	Variable reluctance motor
WBBR	Write back bus register
WDG	Watchdog timer
WKPU	Wakeup unit
WS	Wait state
XBAR	Crossbar
XBIC	XBAR integrity checker

Revision history

Table 1372. Document revision history

Date	Rev.	Changes
09-Sep-2014	1	Initial release
13-Apr-2015	2	*Chapter 2: Introduction* *Table 2: SPC574Sx device feature summary (superset configuration)*: – For SMPU, description updated to “Yes (8 Regions)” – Footnote 1 “SMPU with process ID support extension” removed Updated the description of LINFlexD from “2” to “4”. *Figure 2: Block Diagram*: INTC updated to INTC_0 *Chapter 3: System architecture* *Section 3.3: Motor control Architecture*: Replaced “PWM unit” by “FlexPWN unit”. *Table 7: ADC channels mapping description*: Added Column “PAD LFBGA244“ same as “PAD QFP144” except for row CADC1 - FCH5 that is “PAD[131]”. *Figure 7: Motor control architecture*:Updated picture at CH[32] input by added (except QFP144). *Chapter 4: Signal description* *Table 9: Power supply and reference pins*: – moved all “Exposed pad” rows at the end of the table. – updated footnote 2 with “PWMODE = 1” – Added column LFBGA244 – Added LFBGA244 pins “B16” and “F17” to pad “VDD_HV_IO1” row – Updated description of VDD_HV_IO0 and VDD_HV_OSC *Table 10: System pins*: row PWMODE: inverted “internal” and “external”. *Table 12: Pin multiplexing description*: Extensive updates throughout table. *Table 13: Peripheral multiplexing description*: row cansubsys_0, MSCR[512]: updated “M_CAN_1_RX” by “M_CAN_0_RX” in column “Functions”. *Table 14: Cross-mapping between pads and package pins*: – Removed rows: PAD_PWMODE, EXTAL, XTAL, PAD_TESTMODE, PAD_PORST – Added column LFBGA244 *Chapter 5: Memory map* Updated table name and content in *Table 17: Peripheral bridge 1 memory map* Replaced AIPS by PBRIDGE, AIPS 1 by PBRIDGE_1, RAM Controller by PRAM_0, MCAN 1 by CAN_SUB_1_M_CAN_1, CMU 2 (AIPS 0/1) by CMU 2 (PBRIDGE 0/1), Updated table name and content in *Table 18: Peripheral bridge 0 memory map* with following changes:

Table 1372. Document revision history (continued)

Date	Rev.	Changes
13-Apr-2015	2	Replaced AIPS 0 by PBRIDGE_0, XBAR by XBAR_0, SMPU by SMPU_0, XBIC by XBIC_0, RAM controller by PRAM_0, Flash Memory Controller by PFLASH_0, INTC by INTC_0, SWT by SWT_0, STM by STM_0, DMA by eDMA_0, ADC 0 by SAR_ADC_12BIT_0, ADC 1 by SAR_ADC_12BIT_1, ADC 2 by SAR_ADC_12BIT_2, ADC Supervisor AIPS0 by SAR_ADC_12BIT_B0, ADC Supervisor AIPS0 seq by ADC Supervisor 0 seq, MCAN 0 by CAN_SUB_0_M_CAN_0, MCAN 2 by CAN_SUB_0_M_CAN_2, CRC by CRC_0, DMA Mux 0 by DMAMUX_0, DMA Mux 1by DMAMUX_1, PIT by PIT_0, Wakeup by WKPU, SUL by SIUL2. *Chapter 6: Functional safety* *Table 24: FCCU failure inputs*: – Re-ordered all FFCU inputs. – Input 41”PMC”: replaced “PMU6” and “PMU” by “PMC”. – Input 44 “DSMC”: removed complete row. – Input 44 “RCCU”: added “No reaction” to this fault. Updated failure sources in *Section 6.4.7: Failure handling* Added *Section 6.4.12: LBIST / MBIST architecture* *Chapter 7: Device configuration*, Extensive updates throughout chapter *Chapter 8: Reset and boot* *Section 8.7: System boot*: Rewrited first paragraph text to “After the reset phase is completed, the CPU starts executing at the start address provide by MC_ME peripheral,”. Removed signal p_rstbase *Section 8.8: HW boot record search (SSCM search)*: Removed list bullet 3. and Note: During ST production, till no valid...7 is programmed. Removed BAR execution paragraph. *Table 133: BAF boot record structure*: Sorted offsets from lowest (0x0) value to biggest value (0xC). *Table 134: BAF boot record search locations* , replaced “Flash 128 KB Code Block 0” with “Flash 256KB Code Block 0” *Section 8.5: ESR0 pin functionality*: Updated latest bullet. Updated *Figure 46: System boot sequence*. *Chapter 9: Device Configuration Format (DCF) Records* *Table 139: DCF client list*: STCU - Updated DCF client description: changed STCU_LBCFM to STCU_LBUFM *Figure 51: UTEST Miscellaneous DCF Client*: updated ‘Reset’ with ‘The reset value is described in *Table 139: DCF client list*.’ Added note “LVD11_C, LVD11_FL & LVD3_C monitors once enabled in DCF cannot be cleared by DCF record.” about *Figure 52: REE_BUS bits DCF Client* Added *Chapter 11: Security*

Table 1372. Document revision history (continued)

Date	Rev.	Changes
13-Apr-2015	2	*Chapter 12: Core e200z420n3 Description* *Figure 63: e200z420n3 SUPERVISOR Mode Programmer's Model SPRs*: Updated figure with Local Memory Configuration registers. *Figure 64: e200z420n3 Supervisor Mode Programmer's Model DCRs and PMRs*: Updated figure with Local Memory registers. *Figure 67: EFPU Status and Control Register (SPEFSCR)*: changed OV bit-field 18 to reserved. *Section 12.7.5.13: System Reset Interrupt*: This section is moved after section "Embedded Floating-point Round Interrupt (offset 0xB0)". *Table 184: DMEMCTL0 field descriptions*: This section moved before section "System Call Interrupt (0x80)". *Table 184: DMEMCTL0 field descriptions*: Added this register table. *Section 12.9.2.1: DMEM Configuration Register 0 (DMEMCFG0)*, *Section 12.9.2.2: DMEM Control Register 0 (DMEMCTL0)* and *Section 12.9.2.3: DMEM Control Register 1 (DMEMCTL1)*: Added text note "These locations are accessible by S/W and will not result in error if accessed.". *Chapter 13: System Integration Unit Lite2 (SIUL2)* *Table 191: SIUL2 memory map*: Added table footnote about implemented MSCR registers. *Table 192: SIUL2_MIDR1 field descriptions*: Changed MAJOR_MASK field description to "Major Mask revision" and MINOR_MASK field description to "Minor Mask revision" (was "Silicon revision" for both). For field "PKG", configuration "11001 QFNmr152" changed to "01110 244 BGA". *Figure 98: SIUL2 MCU ID Register #1 (SIUL2_MIDR1)* and *Table 192: SIUL2_MIDR1 field descriptions*: Changed field bit 0:15 to PARTNUM and rewoked description with PARTNUM[15:8] asTechnology, PARTNUM[7:4] as Core and, PARTNUM[3:0] as Subfamily. *Chapter 14: Crossbar switch (XBAR)* *Table 211: XBAR memory map*: – Set address offsets (Was absolute adresses). – Changed register prefix from XBAR0 to XBAR. – Added subsection row named "SLAVE channel n (n=0 to 4)" *Figure 119: Priority Registers Slave (XBAR_PRSn)*: Changed offset format to "0x0000+n*0x100 (n=0 to 4)" *Figure 120: XBAR Control Register (XBAR_CRSn)*: Changed offset format to "0x0010+n*0x100 (n=0 to 4)" and set bit 29 as Reserved with figure footnote 1. "This bit is RW and must not be used". *Chapter 19: Interrupt Controller (INTC)* Removed "n" from field names in *Figure 149: INTC Priority Select Register n (INTC_PSRn)*, in *Figure 150: INTC Priority Select Register n (INTC_PSRn)*, in *Table 245: INTC_PSRn field descriptions* and in *Table 246: INTC_PSRn field descriptions*. *Chapter 20: Enhanced Direct Memory Access (eDMA)* Updated SSIZE field description in *Table 268: DMA_TCDn_ATTR field descriptions*

Table 1372. Document revision history (continued)

Date	Rev.	Changes
13-Apr-2015	2	*Chapter 21: DMA channel multiplexer (DMACHMUX)* *Section 21.1.3: Modes of operation*: added text “for each DMACHMUX instances.” at the end of “Periodic Trigger Mode” bullet New *Figure 196: DMACHMUX triggered channels* Updated *Section 21.5.2: Enabling and configuring sources* *Chapter 24: Clocking* *Figure 203: Clock generation architecture*: Updated by moving XBAR to @140MHz (was @70MHz) Updated *Figure 204: SPC574Sx dual PLL digital interface block diagram* *Table 299: MC_CGM relationship to clocks*: Added following rows: – PLL0 max 120MHz – PLL0-PHI1 max 100MHz – PLL1 max 140MHz – CCCU max 120MHz Changed System clock from “PWM” to “FlexPWM” and “Etimer” to “eTimer” *Table 301: JTAG frequencies*, changed the “e200z0 max. internal JTAG frequency” from 140 MHz to 70 MHz *Section 24.3: System clock frequency limitations*: Changed “PBRIDGE_0 maximum frequency from 40 MHz to 70 MHz in text note. *Section 24.5.1.2.1: PLL1 input clocks*: Added “, PHI1 output clock from PLL0 divided by 2” after PLL0. *Section 24.6.1.1: System clock*: Changed paragraph text “The output clock from PLL0 and external oscillator (XOSC) can be selected for CLOCKOUT.” by “This is a divided version of device's system clock and can be observed outside the device by programming CGM_SC_DC2 register.” *Chapter 25: Dual PLL Digital Interface (PLLDIG)* *Section 25.3: Features*: Removed text in lastes bullet “User-selectable ability to generate a system reset upon loss of lock.” *Figure 210: PLL0 Control Register (PLL0CR)* and *Table 307: PLL0CR field descriptions*: removed field “LOLRE” at bit 29 *Figure 213: PLL1 Control Register (PLL1CR)* and *Table 310: PLL1CR field descriptions*: removed field “LOLRE” at bit 29 *Section 25.6.2: Clock configuration*: Updated the configuration example. *Chapter 26: Clock Monitor Unit (CMU)* *Section 26.5.3: CLKMN1 supervisor*, Updated the NOTE at end of section for clarity. *Section 26.4.1.5: CMU Interrupt Status Register (CMU_ISR)*:Added NOTE to the OLRI field description, "While entering STOP mode, OLRI may be triggered. To avoid this when the system is running on the PLL, disable the XOSC before entering STOP mode.

Table 1372. Document revision history (continued)

Date	Rev.	Changes
13-Apr-2015	2	*Chapter 27: Clock Generation Module (MC_CGM)* Added CGM_SC_DIV_RC; CGM_DIV_UPD_TYPE; CGM_DIV_UPD_TRIG; CGM_DIV_UPD_STAT registers *Table 323: MC_CGM register description*: Changed all offset addresses from 0x07xx to 0x00xx and from 0x08xx to 0x01xx *Section 27.3.1.16: System Clock Divider 0 Configuration Register (CGM_SC_DC0)*: Changed "...Fast XBAR" by "Core and XBAR." *Figure 241: System Clock Divider 0 Configuration Register (CGM_SC_DC0)* and *Table 339: CGM_SC_DC0 field descriptions*: Changed DIV field dimension to 13:15 (was 10:15), Divider 0 Enable (DE) and Divider 0 Division Value. *Section 27.3.1.17: System Clock Divider 1 Configuration Register (CGM_SC_DC1)*: Changed "...Slow XBAR" by "PBRIDGE." *Figure 242: System Clock Divider 1 Configuration Register (CGM_SC_DC1)* and *Table 340: CGM_SC_DC1 field descriptions*: Changed DIV field dimension to 12:15 (was 10:15). *Section 27.3.1.18: System Clock Divider 2 Configuration Register (CGM_SC_DC2)*: Changed "...PBRIDGE clock" by "...CLKOUT clock.". *Figure 243: System Clock Divider 2 Configuration Register (CGM_SC_DC2)* and *Table 341: CGM_SC_DC2 field descriptions*: Changed DIV field dimension to 7:15 (was 10:15), Divider 2 Enable (DE) field is writable (was read-only). *Section 27.3.1.19: System Clock Divider 3 Configuration Register (CGM_SC_DC3)*: Changed "...PBRIDGE clock" by "...DMA/INTC clock.". *Figure 244: System Clock Divider 3 Configuration Register (CGM_SC_DC3)* and *Table 342: CGM_SC_DC3 field descriptions*: Changed DIV field dimension to 12:15 (was 10:15), Divider 3 Enable (DE) and Divider 3 Division Value (DIV). *Figure 247: Auxiliary Clock 0 Divider 0 Configuration Register (CGM_AC0_DC0)* and *Table 345: CGM_AC0_DC0 field descriptions*: Changed DIV field dimension to 12:15 (was 9:15). *Figure 248: Auxiliary Clock 0 Divider 1 Configuration Register (CGM_AC0_DC1)* and *Table 346: CGM_AC0_DC1 field descriptions*: Changed DIV field dimension to 12:15 (was 9:15). *Figure 252: Auxiliary Clock 1 Divider 0 Configuration Register (CGM_AC1_DC0)* and *Table 350: CGM_AC1_DC0 field descriptions*: Changed DIV field dimension to 10:15 (was 9:15). *Figure 257: Auxiliary Clock 4 Select Control Register (CGM_AC4_SC)* and *Table 355: CGM_AC4_SC field descriptions*: Changed SELCTL field value 0000 COSC to RCOSC. *Figure 259: Auxiliary Clock 4 Divider 0 Configuration Register (CGM_AC4_DC0)* and *Table 357: CGM_AC4_DC0 field descriptions*: Changed DIV field dimension to 12:15 (was 9:15). *Figure 265: Auxiliary Clock 5 Select Control Register (CGM_AC5_SC)* and *Table 363: CGM_AC5_SC field descriptions*: Changed first sentence to "This register is used to select the current clock source for the PLL1 reference clock.". *Figure 267: Auxiliary Clock 5 Divider 0 Configuration Register (CGM_AC5_DC0)* and *Table 365: CGM_AC5_DC0 field descriptions*: Changed DIV field dimension to 14:15 (was 9:15). *Section 27.4.1.2: Progressive System Clock Switching*: Added note. *Figure 275: MC_CGM Auxiliary Clock 4 Generation Overview*: Updated figure with input PLLPHI (was PLLPHI1). Deleted the whole sections: Divider Jitter Injection and Fractional Divider Details.

Table 1372. Document revision history (continued)

Date	Rev.	Changes
13-Apr-2015	2	Corrected some register accesses from “User read/write/write” to “User read/write” (such as *Figure 226* and *Figure 230*. *Chapter 29: IRCOSC digital interface* Added IRCOSC_NT and IRCOSC_TT register. *Section 29.1: Introduction*: added “, as well as an internal temperature sensor and an internal voltage regulator used to compensate external variations in temperature and voltage” *Section 29.3.1.1: IRCOSC control register (IRCOSC_CTL)*: – Added note to register access. – Added bit10 to USER_TRIM bit field and set new description *Chapter 30: Platform RAM controller (PRAM)* *Figure 282: Platform RAM configuration register 1 (PRCR1)*: Changed reset bit 26 and 27 values to undefined (was 0) and add a figure footnote to them *Table 378: PRCR1 field descriptions*: Added note to field FT_DIS. *Table 377: PRAM controller memory map*: changed reset value to 0x000002X0 (was 0x0) with a table footnote to register section. Removed “Read-modify-write datapath” figure. – *Section 30.5.1.1: E2EECC considerations on less-than-64-bit write transactions*: Added “on a 32-bit write” to the first bullet. – Changed text “64-bit read from RAM array” in figure of second bullet (was “16 bit write request”) – Added “The PRAM controller” to the third bullet. *Section 30.5.2: Transaction monitor*, removed “RAM safety feature outputs” table *Chapter 31: Flash memory controller (PFLASH Controller)* Updated *Figure 288: Platform-centric block diagram with flash memory controller* *Chapter 32: Embedded Flash Memory* For registers SEL0, SEL1, and SEL2, updated all field descriptions to read “The select field must not be written when UT0[AIE] is high.“ (was “The select field is not writable...“). *Chapter 33: Successive Approximation Register Analog-to-Digital Converter (SARADC) Digital Interface* *Table 437: CTR0–CTRL3 field descriptions*: Removed text “for most of the SARADC it is greater than 2” in PRECHG field description. *Chapter 34: Enhanced Cross Trigger Unit (ECTU)* *Figure 387: Commands list register n (CLRn)*: changed n values from 0 to 63 (was 1 to 64). *Table 472: CTU interrupt flags*: Added sub-row named “FIFO interrupt flags (i = 1 to 8)” and reworked FIFO_FULL[i], FIFO_EMPTY[i], FIFO_OVERFLOW[i] and FIFO_OVERRUN[i].

Table 1372. Document revision history (continued)

Date	Rev.	Changes
13-Apr-2015	2	Added*Chapter 33: Decorated Storage Memory Controller (DSMC)* *Chapter 39: Software Watchdog Timer (SWT)* Updated reset value in *Figure 551: SWT Counter Output Register (SWT_CO)* *Chapter 40: Periodic Interrupt Timer (PIT)* Modified n value from 1 to 0 into *Table 623: PIT memory map* *Chapter 41: CAN Subsystem* *Section 41.1: Introduction*: added details of implemented CAN subsystems. *Section 41.7: External signal description*: Added 2 bullets - M_CAN_x_RXFD: M_CAN_x Receive Fast Data control signal - M_CAN_x_TXFD: M_CAN_x Transmit Fast Data control signal *Chapter 42: Deserial Serial Peripheral Interface (DSPI)* *Table 698: DSPI memory map*: Added “Reserved space for 0x4C - 0x78” *Figure 965: MC_RGM block diagram* changed ‘FCCU failure to react reset ‘ into ‘FCCU failure detected by FOSU’ *Table 937: Functional Event Status Register (RGM_FES) field descriptions* modified the description of F_SOFT_FUNC bit field *Table 938: RGM_FERD register field descriptions* modified the description of D_SOFT_FUNC bit field *Table 940: Functional Event Status Register (RGM_FBRE) field descriptions* modified the description of BE_SOFT_FUNC bit field *Table 941: RGM_FESS register field descriptions* modified the description of SS_SOFT_FUNC bit field *Chapter 43: LINFlexD* Updated baud rate calculation formula under conditional tag ‘Baud_rate_25=yes’ in *Section 43.2.1: LIN mode features* and *Section 43.2.2: UART mode features* Corrected the naming of LINIER register in the whole chapter Updated register offset of *Section 43.3.2.1: Operating modes* Updated bit field of *Section 43.4.2.25: DMA Tx Enable Register (DMATXE)* and *Section 43.4.2.26: DMA Rx Enable Register (DMARXE)* Removed” Register map” section *Table 738: LINFlexD memory map* : Note “In Master mode, ...would result in ips_xfer_err being flagged” updatd to read “In master mode, ...would result in bus transfer error” *Chapter 46: Reset Generation Module (MC_RGM)* *Table 932: MC_RGM memory map*: updated with new registers RGM_DBRE and RGM_FBRE. *Figure 966: Destructive Event Status Register (RGM_DES)*: moved F_SSCM from position 14 to 17. *Figure 967: Destructive Event Reset Disable Register (RGM_DERD)*: moved D_SSCM from position 14 to 17. *Table 937: Functional Event Status Register (RGM_FES) field descriptions*: updated description for F_SOFT_FUNC (bit 28) *Table 938: RGM_FERD register field descriptions*: updated description for D_SOFT_FUNC (bit 28)

Table 1372. Document revision history (continued)

Date	Rev.	Changes
13-Apr-2015	2	*Table 941: RGM_FESS register field descriptions*: updated description for SS_SOFT_FUNC (bit 28) *Figure 972: Functional Event Alternate Request Register (RGM_FEAR)*: updated AR_FCCU_SOFT(bit 25) to RO and added '1' in AR_VOR_FUNC (bit 7) description. *Figure 979: Peripheral Reset Register 2 (RGM_PRST2)* and *Table 946: RGM_PRST2 register field descriptions*: added MCAN_0_2_RAM_RST (bit 21). *Figure 987: Peripheral Reset Status Register 2 (RGM_PSTAT2)* and *Table 954: RGM_PSTAT2 register field descriptions*: added MCAN_0_2_RAM_STAT (bit 21). *Chapter 47: Boot Assist Flash (BAF)* Removed CAN information *Table 964: BAF image version*: updated field "16:23" with Reserved. *Section 47.4: Resources accessed by BAF code execution*: – updated title (was "Resources") – added "The BAF code execution uses ...the main functions." – PFLASH: added "The two PFLASH are used".- *Table 963: BAF image header*: updated "Field Description" values for "Address offset" "0000h, 0004h, 0008h" *Chapter 48: System Status and Configuration Module (SSCM)* *Table 983: Resulting Life Cycle*: Removed the 3 latest rows including table footnote (1) and (3). Modified RAE bit field definition in *Table 977: ERROR field descriptions* Updated *Section 48.3.2: Boot mode functionality* and *Section 48.3.4.2: Reset Configuration Half-Word* *Chapter 49: Power management controller digital interface (PMC_Dig)* Extensive updates throughout chapter *Chapter 51: Mode Entry Module (MC_ME)* *Table 1031: MC_ME register description* and *Section 51.3.2.25: Peripheral Control Register n (ME_PCTLn)* added ME_PCTL15 register *Chapter 53: Debug and Calibration Interface (DCI)* *Section 53.1.3.1: Software enabled debug and calibration mode*, removed second paragraph in the note: "If both debug methods are used simultaneously, then an external tool may arbitrarily connect when the SoC has already been halted using the software debug mode. In this case, two consecutive assertions of external JCOMP are necessary to allow the external tool to take control of debug session." *Chapter 55: JTAG Controller (JTAGC)* *Table 1106: 6-bit JTAG instructions*: Added row "Reserved"

Table 1372. Document revision history (continued)

Date	Rev.	Changes
13-Apr-2015	2	*Chapter 60: e200z420n3 Nexus 3 Module* *Section 60.4.2: Port Configuration Register - reference only (PCR)*: Added note after first paragraph. *Section 60.4.3: Nexus Development Control Register 1 (DC1)*: Added note after first paragraph. *Figure 1216: Development Status Register (DS)*: Changed register access to Read-only (was Read-write). *Figure 1228: Read/Write Access Control/Status register (RWCS)*: Changed bit field MAP access to Read-only (was Read-write). *Chapter 61: Nexus Crossbar Multi-Master Client (NXMC)* *Section 61.5: Programming considerations (RESET)*: Added text “after negation of TRST and should be programmed before deassertion of system reset” at the end. *Chapter 62: Cyclic Redundancy Check (CRC) Unit* Removed notes like "Refer to the chip configuration chapter for number of contexts used in this application." since the values are specified through the Chapter Specified the index “n” in all register definition section. *Chapter 64: Fault Collection and Control Unit (FCCU)* *Section 64.5.2: FSM description*, The key to lock the FCCU is written to FCCU_PERMNT_LOCK and not FCCU_TRANS_LOCK. Added footnote to “same as NORMAL” entries in *Table 1262: Dual rail encoding*, *Table 1263: Time switching encoding* and *Table 1264: Bi-stable encoding*. Replaced all occurrences of NCF with RF. Added missing bit numbers from field description tables. Added a footnote to the reset value of bit 28 in *Section 64.6.3: FCCU Configuration Register (FCCU_CFG)* *Table 1265: FCCU memory map*: Renamed table title (was “FCCU register set”). Removed Section “FCCU register map” Removed glitch filter related text of *Section 64.6.1: FCCU Control Register (FCCU_CTRL)* *Section 64.6.14: FCCU NA Freeze Status Register (N2AF_STATUS)* : – Footnote updated from “You can read the bits in this field”,.....Writing “1” has no effect” to “These are reserved bits. These bits are always read as ‘0’ and muist always be written with ‘0’. – The reset value of field NAFS[24:31] changed from “0” to “w1c”. *Section 64.6.16: FCCU NF Freeze Status Register (N2FF_STATUS)* : Footnote updated from “You can read the bits in this field”,.....Writing “1” has no effect” to “These are reserved bits. These bits are always read as ‘0’ and muist always be written with ‘0’. *Section 64.6.17: FCCU FA Freeze Status Register (F2A_STATUS)*: – Footnote updated from “You can read the bits in this field”,.....Writing “1” has no effect” to “These are reserved bits. These bits are always read as ‘0’ and muist always be written with ‘0’. – The reset value of field FAFS[24:31] changed from “0” to “w1c”

Table 1372. Document revision history (continued)

Date	Rev.	Changes
13-Apr-2015	2	*Chapter 65: Self-Test Control Unit (STCU2)* *Table 1304: STCU2 register set*, *Section 65.5.31: STCU2 LBIST Control n Register (STCU_LB_CTRLn)* to *Section 65.5.43: STCU2 MBIST Control k Register (STCU_MB_CTRLk)* and *Table 1348: STCU2 register map*: Updated Addresses and Register names for register STCU_LB_CTRL to STCU_MB_CTRL. Address was “0x0100+(NLBIST-1)*0x40h”, is now “0x0100+n*0x40 with n=0 to (NLBIST-1)”, same modification with NMCUT. Added ‘*n*’ to Register names. Updated FIELD*x* with FIELD[high_value:low_value]. *Table 1304: STCU2 register set*: Added table footnote and Reserved spaces at the following addresses: 0x0004-0x0007 0x0034-0x003B 0x005C-0x0073 0x0120-0x05FF *Table 1315: STCU_LBS field descriptions* to *Table 1319: STCU_LBUFM field descriptions*: Updated LBS*x*, LBE*x*, ... LBUFM*x* with LBS[(NLBIST-1):0], LBE[(NLBIST-1):0]... LBUFM[(NLBIST-1):0] *Table 1320: STCU_MBSL field descriptions* to *Table 1320: STCU_MBSL field descriptions*: Updated MBS*x*,... MBUFM*x with* MBS[(NMCUT-1):0]... MBUFM[(NMCUT-1):0] *Table 1321: STCU_MBSM field descriptions* to *Table 1333: STCU_MBUFMM field descriptions*: Updated MBS*x*,... MBUFM*x with* MBS[(NMCUT-1):32]... MBUFM[(NMCUT-1):32] *Table 1322: STCU_MBSH field descriptions* to *Table 1334: STCU_MBUFMH field descriptions*: Updated MBS*x*,... MBUFM*x with* MBS[(NMCUT-1):64]... MBUFM[(NMCUT-1):64] Fixed typos with Off:line and On:line. *Chapter 67: Password and Device Security Module (PASS)* *Table 1353: PASS memory map*: added some reserved spaces and added table footnotes to “DCF Client Address” column. *Figure 1415: Password Group n - Lock 3 status register (LOCK3_PGn)* and *Table 1361: LOCK3_PGn register field descriptions*: – Added RL4 and RL3 with same description as other RLn. – Deleted note in all RLn. – Changed footnote to (1) in reset value bits16:30 of field U_256CLK. *Section 67.2.1.5: Password Group n - Lock 0 Status register (LOCK0_PGn)* to *Section 67.2.1.8: Password Group n - Lock 3 status register (LOCK3_PGn)*: – Removed the sentence “The LOCK bits do not come into effect before the lifecycle matures to OEM Production or older.” – Removed the sentence “Each Password Group has one Lock 2 Status Register. This register ... refer to the Flash description.” and set this paragraph in a new section. Removed “Modifying Lifecycle (maturing the device)” section. Updated *Table 1363: Debug Interface Enable Truth Table*. Updated *Table 1364: Flash Memory Read Protection Truth Table*.

Table 1372. Document revision history (continued)

Date	Rev.	Changes
13-Apr-2015	2	*Figure 1414: Password Group n - Lock 2 status register (LOCK2_PGn)* and *Figure 1415: Password Group n - Lock 3 status register (LOCK3_PGn)*: Replaced field L_256LCK by L_128LCK - Lower 128KByte Block Lock". *Figure 1415: Password Group n - Lock 3 status register (LOCK3_PGn)*: Added figure footnote to RL1-RL4 and U_256LCK. *Chapter 22: Wakeup unit (WKPU)* Renamed *NMI Configuration Register (NCR)* fields: NDSS3 → RDSS3 NWRE3 → RWRE3 NREE3 → RREE3 NFEE3 → RFEE3 NFE3 → RFE3 Updated *Figure 199: Wakeup unit Block Diagram*. *Section 22.1.1: Overview*: Added text "This NMI external source can also generate a functional reset event to the MC_RGM". Renamed *NMI Status Flag Register (NSR)* fields and updated their descriptions: NIF3 → RIF NOVF3 → ROVF Updated the reset conditions of *NMI Configuration Register (NCR)* fields and renamed fields and updated their descriptions as per: NLOCK3 → RLOCK RDSS3 → RDSS RWRE3 → RWRE RREE3 → RREE RFEE3 → RFEE RFE3 → RFE Removed the NMI pad diagram. *Chapter 44: SENT Receiver (SRX)* Extensive updates throughout chapter *Chapter 45: FlexRay Communication Controller (FlexRay)* *Section 45.5.2.66.23: Protocol Configuration Register 22 (FR_PCR22)*: Added footnote for bit 15 "Reserved bit or field. Application must not write any value different from the reset value." *Chapter 35: Motor Control Pulse Width Modulator Module (FlexPWM)* *Table 520: FlexPWM memory map*: – "CAPTCOMPAn" updated to "CAPTCMPAn" – "CAPTCOMPBn" updated to "CAPTCMPBn" – "CAPTCOMPXn" updated to "CAPTCMPXn" – "CVALxCn" updated to "CVALxCYCn", where x = 0 to 5. *Figure 462: Double Switching Output Example*: – "PWMA" updated to "Internal PWMA" – "PWMB" updated to "Internal PWMB" – "DBLPWM" updated to "PWMA & PWMB output"

Table 1372. Document revision history (continued)

Date	Rev.	Changes
12-May-2016	3	*Chapter 2: Introduction* *Section 2.2: Features*: – Modified MPU into SMPU and replaced QFN package data by LFBGA ones – Increased ECC Flash memory size from 1.0 MB to 1.5 MB Code Flash *Table 2: SPC574Sx device feature summary (superset configuration)*: – Increased SRAM size from 64 KB to 96 KB – Increased ECC Flash memory size from 1.0 MB to 1.5 MB Code Flash – Changed the number of LinFlexD from 2 to 4 – Updated the table footnote for the number of MDO pins *Figure 2: Block Diagram*: – Replaced LINFlex by LINFlexD – added missing CMU4 – Added missing modules to block PBRIDGE_0: PASS, JDC, MC_PCU – Added LinFlexD_2 to block PBRIDGE_0, LinFlexD_3 to PBRIDGE_1 – Added a 2nd arrow between e200z420 and top side of XBAR, moved the arrow common to e200z420 & e200z419 to the bottom side of the XBAR. – Renamed AIPS instances by PBRIDGE, Wakeup by WKPU – Added missing modules to block PBRIDGE_0: PWMSync, CCCU_0 – Added missing modules to block PBRIDGE_1: PWMSync, CCCU_1 – Added block NPC next to SPU *Chapter 3: System architecture* *Section 3.1: Platform Architecture*: removed sentence “The blue blocks refer to ...”. *Figure 3: Platform block diagram*: – updated SRAM size by 96KB (was 64). Colored safety related blocks in brown, removed other colors. – updated FLASH size to 1,5MB (was 1MB) *Section 3.1.2.1: CPU exception system*: removed the last paragraph sentence “The figure also shows the HW vector mode...”. *Figure 4: CPU Exceptions vs. Interrupt Controller*: removed Note “Read only: Default to SW Mode” and updated reference to “z0h manual” by ““core manual”. *Section 3.2.1.1: System memory*: updated size by 96KB (was 64). *Figure 5: SPC574Sx flash memory block diagram*: updated Flash Memory Array size by 1632KB (was 1120). *Section 3.2.2.2: Flash memory array*: updated size by 1536KB (was 1024). *Section 3.2.2.2.1: Features*: updated 256KB blocks number by 5 (was 3). *Figure 6: Flash memory segmentation and read-while-write partitioning*: added 2 more blocks of 256KB (blocks 3 and 4). Replaced “AIPS” instances by “PBRIDGE”, “AIC” by “PBRIDGE Checker” through the chapter. *Figure 7: Motor control architecture*: updated figure title from “USE CASE for a two 3-phases motor” to “USE CASE for two 3-phase motors” *Table 7: ADC channels mapping description*, PAD[25], PAD[105] are no longer bonded in any package.

Table 1372. Document revision history (continued)

Date	Rev.	Changes
12-May-2016	3	*Chapter 4: Signal description* Updated *Table 8: Pin startup and reset states* Updated *Table 9: Power supply and reference pins* Updated *Table 12: Pin multiplexing description*: *Table 13: Peripheral multiplexing description*: Renamed MCAN_0 to MCAN_1 Updated *Table 14: Cross-mapping between pads and package pins* *Chapter 5: Memory map* *Table 17: Peripheral bridge 1 memory map*: – in the OPACR19 and OPACR112 entries, replaced AIPS by PBRIDGE – Added LinFlex_3 at start address 0xFBE9_0000 – Changed description “PRAM_1 (MCAN 1 Sub-system)” to “CAN_SUB_1 Message RAM” at start address 0xFBED_4000 *Table 18: Peripheral bridge 0 memory map*: – Added LinFlex_2 at start address 0xFFE9_0000 – Changed description “RAM controller for MCAN 0 and MCAN 2 subsystem” to “CAN_SUB_0 Message RAM” at start address 0xFFED_4000 – Changed description “RCOSC dig” to “RC16M_DIG (IRCOSC)” at start address 0xFFFE_0000 – Changed description “XOSC dig” to “OSC40M_DIG (XOSC)” at start address 0xFFFE_0000 – Changed description “Flash memory (registers) ” to “FLASH_0 (Flash memory registers)” at start address 0xFFFE_0000 *Table 16: Flash memory map*: added Flash 256 KB Code Blocks 3 and 4 *Table 15: Memory map (XBAR slave ports)*: – Changed End address of Flash memory to 0x0113_FFFF (was 0x3FFF_FFFF) – Added Reserved space at address 0x0114_0000-0x3FFF_FFFF – Changed System RAM (64KB) to (96KB) and its Start address to 0x4001_0000 and its End address to 0x4001_7FFF (was 0x4000_0000 / 0x4000_FFFF) *Chapter 6: Functional safety* *Table 24: FCCU failure inputs*: – Added columns “HW fault behaviour” and “Error reaction path check”. – Added footnote to FCCU input #51 (Lockstep Mode) – Added FCCU inputs #62 to #75 *Table 25: MBIST partitioning*: hanged sram end address from 0x4000_FFFF to 0x4001_7FFF *Section 6.4.7: Failure handling*: Updated failure sources – Added *Section 6.5: Register classifications* *Figure 22: ECC protection scheme for CAN subsystems*, exchanged MCAN 1 with MCAN 2

Table 1372. Document revision history (continued)

Date	Rev.	Changes
12-May-2016	3	*Chapter 7: Device configuration*: Extensive updates throughout the chapter *Chapter 8: Reset and boot* *Table 125: Functional reset events*: – Modified signal description of F_SOFT_FUNC module – Modified footnote 5. *Table 127: Module status during the reset phase*: Updated P1, P2 an P3 for FCCU module *Table 134: BAF boot record search locations*: Updated line ‘Search order #6’ from 128KB to 256KB *Section 8.7: System boot*: Updated both text and *Figure 46: System boot sequence* *Section 8.8: HW boot record search (SSCM search)*: Updated both text and *Figure 47: Valid boot header search* Updated *Section 8.9: SW boot record search (BAF search)* *Chapter 9: Device Configuration Format (DCF) Records* *Figure 51: UTEST Miscellaneous DCF Client*: Renamed field “OSC_EN_40MHZ” to “XOSC_EN_40MHZ” *Table 140: UTEST Miscellaneous DCF Client field descriptions*: – In field description “XOSC LOAD CAP SEL”, note “Please refer to....Device Datasheet” added – Renamed field “OSC_EN_40MHZ” to “XOSC_EN_40MHZ” *Table 139: DCF client list*: – REE_BUS_UTEST: added a cross reference to the appropriate register – PNREG_ENB: added a cross reference to the appropriate register in PMC Digital chapter – ESR0_CFG: added a cross reference to the appropriate register PMC Digital chapter – Removed BBCTL – Removed CTL UPDATE – Added (NLBIST) and (NMCUT) to STCU subsection. – Renamed DCF client description “REE_BUS_UTEST” to “REE_BUS”. – Added cross-reference to PMC_DIG chapter to PMC_DIGITAL subsection – PNREG_ENB of DCF client description: added crossreference to Section IPS Regulator Monitor Register (IREGU_MON) – ESR0_CFG of DCF client description: added crossreference to Section ESR0 ConFiGuration Register (ESR0_CFG) *Figure 52: REE_BUS bits DCF Client* and *Table 141: REE_BUS bits DCF client field descriptions*: Renamed “REE_BUS bits DCF client” (was PMC_REE bits DCF client”

Table 1372. Document revision history (continued)

Date	Rev.	Changes
12-May-2016	3	*Chapter 10: Power management* *Section 10.2: Architecture*: Replaced QFN152 by BGA244 in introduction sentence *Figure 54: Power supply scheme BGA244 and QFP144(1)*: – Updated picture – Replaced QFN152 by BGA244 in the figure title – Replaced all EXP by GND – added a footnote to specify that only BGA244 is connected to ground balls (GND) *Figure 55: Power supply scheme QFP100*: Updated picture *Figure 58: Voltage monitor coverage*: Replaced LVD290_IOL0 by LVD290_XOSC *Figure 59: Monitor trigger and system reaction architecture*: Replaced FCCU_NCFS_CFGx by FCCU_RF_CFGx Added *Section 10.2.7: BIST details* *Table 148: Calculating the BIST_TIME values*: Added PBRIDGE clock frequency 70 MHz and BIST_TIME values *Section 10.2.3: Ballast*, replaced 250mA.with 325mA. *Section 10.2.7: BIST details*, replaced 25 MHz with 70 MHz *Figure 62: Monitor checking timing diagram* and *Table 147: Timing descriptions*, updated timing values *Table 148: Calculating the BIST_TIME values*, Updated values *Chapter 11: Security* *Section 11.1: Basic security*: Removed bullet “Secure watchdog timer Updated *Section 11.3: Detailed security information* *Chapter 12: Core e200z420n3 Description* Added *Section 12.8.2: Access Protection Operation* *Chapter 13: System Integration Unit Lite2 (SIUL2)* *Section 13.1.1: Overview*: Updated the number of DMA inputs from 16 to 6. *Section 13.1.2: Features*: Added bullet “Control of analog path switches” *Section 13.3.2: Pad control*: Added bullet “Control of analog path switches” *Table 191: SIUL2 memory map*: Changed field name “PARTNUM” by “FAMILYNUM” at bits 16:23 *Table 191: SIUL2 memory map*: Updated reset value to 0x0080_0000 *Figure 99: SIUL2 MCU ID Register #2 (SIUL2_MIDR2)* and *Table 193: SIUL2_MIDR2 field description*: – Replaced field name “PARTNUM” by “FAMILYNUM” – FAMILYNUM field description updated (SPC574Sxx instead of SPC570Sxx) – Updated footnote (1) to all bits 1:31 – Changed reset value of bits 28:31 to 0xF (was 0x0)

Table 1372. Document revision history (continued)

Date	Rev.	Changes
12-May-2016	3	*Figure 109: I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR0–SIUL2_MSCR511)* and *Table 202: SIUL2_MSCR0–SIUL2_MSCR511 field description*: – Removed APC field – Changed bit dimension of field ODC to bits 6:7 (was 5:7) – Updated OERC field description with “Specifies the output impedance, drive strength, and slew rate of the associated pin.” – Updated reset value to 0x0084_0000 *Table 203: I/O MSCR Reset State Exceptions*: Updated this table. *Chapter 16: Peripheral bridge* Replaced all occurences of AIPS with PBRIDGE *Table 221: Peripheral bridge memory map*: Removed registers PBRIDGE_PACRG and PBRIDGE_PACRH *Table 223: PACR memory map*: – Removed registers PACRG and PACRH – Set register PACRE to Reserved. *Section 16.1.1: Features*: Feature “Supports a pair...memory accesses” updated to “Supports 32 bit transaction”. *Section 16.3.1: Access support*: Removed text “The peripheral bridge performs...single transfer” *Figure 129: Peripheral Access Control Registers (PBRIDGE_PACRx)*: – Removed offset address for registers PACRG and PACRH – Removed PACRE register offset (was 0x100) Corrected offset values from OPACRQ to OPACRAE registers (was 0x140) *Chapter 17: System Memory Protection Unit (SMPU)* *Table 227: SMPU memory map* and *Figure 138: Region Descriptor n, Word 0 (RGDn_WORD0)* to *Figure 141: Region Descriptor n, Word 3 (RGDn_WORD3)*: Corrected all RGD register occurences to n=0 to 11. *Chapter 19: Interrupt Controller (INTC)* Extensive modifications to clarify text and register naming for a single processor device Removed all references to the INTC_MPROT register throughout the chapter *Table 245: INTC_PSRn field descriptions*: Updated bit field description of PRC_SEL0 *Section 19.4: Modes of operation*: removed note about *n* = 0 to avoid confusion. *Section 19.4.1: Software vector mode*: “HVEN_PRCn bit” updated to “HVENn bit”. *Chapter 20: Enhanced Direct Memory Access (eDMA)* *Figure 157: Control Register (DMA_CR)*: – Reset Value of bit “GRP1PRI” (Bit 20-21) changed from “00” to “01”. – For bit “GRP1PRI”, added footnote “Bit 20 is always set to 0 and not writable” – For bit “GRP0PRI”, added footnote “Bit 22 is always set to 0 and not writable” – Bit 31 changed from “Reserved” to “EBW” *Table 250: DMA_CR field descriptions*: Added EBW bit field description

Table 1372. Document revision history (continued)

Date	Rev.	Changes
12-May-2016	3	*Chapter 21: DMA channel multiplexer (DMACHMUX)* Updated register name to CHCFG (was CHCONFIG) throughout the chapter. *Table 292: DMACHMUX memory map*: Corrected register name to CHCFG (was CHCNFG) Updated *Section 21.1.2: Features* *Chapter 22: Wakeup unit (WKPU)* Updated *Figure 199: Wakeup unit Block Diagram* and “AIPS” renamed to “PBRIDGE” *Section 22.1.1: Overview*: Added text “This NMI external source can also generate a functional reset event to the MC_RGM”. *Section 22.2.2.1: NMI Status Flag Register (NSR)*: Renamed fields and updated their descriptions (NIF3 → RIF and NOVF3 → ROVF) *Section 22.2.2.2: NMI Configuration Register (NCR)*: – Renamed fields and updated their descriptions (NLOCK3 → RLOCK, RDSS3 → RDSS, RWRE3 → RWRE, RREE3 → RREE, RFEE3 → RFEE and RFE3 → RFE) – Removed the NMI pad diagram *Table 295: WKPU Memory Map*: – deleted Size column – renamed Use column to Register – deleted the Abbreviation column and moved the abbreviation into the Register column *Chapter 23: Temperature sensor (TSENS)* *Section 23.4: TSENS outputs*: Replaced P1, P2, C1, and C2 with P2, C2, P1, and C1 respectivety in note below the table and appended for Temperature Sensor 0. Added Flash locations for Temperature Sensor 1. *Chapter 24: Clocking* Updated *Figure 204: SPC574Sx dual PLL digital interface block diagram* *Table 300: Maximum system level clock frequencies*: added 2 rows TCK (JTAGM/PAD) *Table 301: JTAG frequencies*: updated “e200z0” by “e200z420n3” *Section 24.3: System clock frequency limitations*: Reworded the sentence in the Note “In order to... dividers” *Chapter 25: Dual PLL Digital Interface (PLLDIG)* *Section 25.3: Features*: – Added “The frequency range after the prescaler is required to be 8-20Mhz” – Removed the information about inout and output clock frequency range. Added a note to consult the Device data sheet for the same *Table 306: Dual PLL Digital Interface memory map*: – Removed “Access” and “Reset Value (hex)” columns. – Reformated all end address of reserved spaces *Figure 212: PLL0 Divider Register (PLL0DV)* and *Figure 215: PLL1 Divider Register (PLL1DV)*: added “default” to figure footnote. *Figure 211: PLL0 Status Register (PLL0SR)*: Updated reset value

Table 1372. Document revision history (continued)

Date	Rev.	Changes
12-May-2016	3	*Chapter 27: Clock Generation Module (MC_CGM)* *Section 27.3.1.18: System Clock Divider 2 Configuration Register (CGM_SC_DC2)*: Changed sentence from “This register relates to PBRIDGE clock.” to “This register relates to CLKOUT clock.” *Section 27.3.1.12: Divider Update Type Register (CGM_DIV_UPD_TYPE)*: added bit fields AUX5_UPD_TYPE and AUX4_UPD_TYPE *Section 27.3.1.14: Divider Update Status Register (CGM_DIV_UPD_STAT)*: added AUX5_UPD_STAT and AUX4_UPD_STAT and set description as AUXn_UPD_STAT *Section 27.4.1.4.2: System Clock Divider Update Triggering*: Removed text in bracket: “(This is applicable to auxilliary mux 4 as well)’ *Chapter 29: IRCOSC digital interface* Updated *Section 29.3.1.2: IRCOSC Native Trimming register (IRCOSC_NT)* and *Section 29.3.1.3: IRCOSC Temperature Trimming register (IRCOSC_TT)* *Chapter 30: Platform RAM controller (PRAM)* *Section 30.1: Introduction*: Reworked the sentence “The read data is the result of a RAM access” *Figure 282: Platform RAM configuration register 1 (PRCR1)*: Reset value changed to 0 for bits 26 and 27 *Table 377: PRAM controller memory map*: – Removed “Width” and “Access” columns. – Changed PRCR1 register reset value to 0x0000_0200 (was 0x000002X0). *Section 30.3.1.2: Optional read wait-state*: Removed note. *Section 30.5.2: Transaction monitor*: Removed text from “As a countermeasure against false misses to the late-write buffer on reads,” to “the RAM or from the late-write buffer.” *Chapter 31: Flash memory controller (PFLASH Controller)* *Table 380: Flash memory controller memory map*: – updated table footnotes – removed Access and Reset value columns *Section 31.5.7: Censorship*: Replaced “TEST flash” by “UTEST flash” *Section 31.5.13.1.1: e2eEDC and flash accesses*: Updated fl_data value *Section 31.5.13.1.2: ECC and data flash accesses*: Replaced “illegal code value” by the value itself, “FFFF” *Section 31.5.13.2: Flash address generation check*: Replaced “fl_enc_addr [23:2]” by “encoded address as received by flash memory” and “fl_enc_error” by “address encoding error provided by flash memory” *Section 31.5.14: Array integrity considerations*: Removed internal naming “fl_aid” *Chapter 32: Embedded Flash Memory* *Section 32.3.2.9: Address register (ADR)*: Added Flash format addresses mapping with Absolute addresses *Table 413: UM9 field descriptions*: Reformated MISR[288] to MISR288 and corrected its bit-field value to 31 (was 0:31)

Table 1372. Document revision history (continued)

Date	Rev.	Changes
12-May-2016	3	*Chapter 34: Enhanced Cross Trigger Unit (ECTU)* *Section 34.2: Main features*: Number of triggers updated to 8 *Table 477: TGS registers*: Removed TCR8–15 registers *Table 478: SU registers* : Commands List Register updated from “CLR1-CLR128” to “CLR0-CLR63” *Table 481: TGSISR field descriptions*: Added note “See figure: Module to module connections in Device Configuration chapter. *Table 510: CTUCR field descriptions*: Added sentence “It is set by the user and automatically reset when it is reloaded in MT_clk domain. The reloaded bit in MT_clk domain gets reset by MRS.“ to FILTER_enable bit field. *Section 34.10.26: Status register 1 (ST1)*: Name updated from “SR1” to “ST1” *Section 34.10.27: Status register 2 (ST2)*: Name updated from “SR2” to “ST2” Added *Section 34.10.40: CTU expected CRC register (CTU_EXP_CRC_VAL)* and *Section 34.10.41: CTU computed CRC register (CTU_COMP_CRC_VAL)* *Section 34.10.42: CTU Mask CRC computation register (CTU_MASK_CRC_VAL)*: added additional bits *Figure 406: CTU_CFGSR2 register* : Footnote added “The reset value is device dependent. Please see the Chapter: Device configuration for details.” *Chapter 35: Motor Control Pulse Width Modulator Module (FlexPWM)* *Section 35.3.31: Capture Value 2 Cycle register n (CVAL2CYCn)*: Updated the width of field “CVAL2CYC” from “3:0” to “2:0” *Section 35.3.33: Capture Value 3 Cycle register n (CVAL3CYCn)*: Updated the width of field “CVAL3CYC” from “3:0” to “2:0” *Section 35.3.38.1: Output Enable register (OUTEN)*: removed index n in the title and in the offset address, now it is 0x0140 instead of BASE + 0x50*n+0x0140(n=0 to 3) *Section 35.3.38.5: Master Control register (MCTRL)*: Added note “The relationship between the fields of MCTRL and the submodules is as follows:...” *Section 35.4.2.6: Enhanced Capture Capabilities (E-Capture)*: Added note “CVAL0-CVAL1 are used for PWMx, CVAL2 and CVAL3 are used for PWMA and finally, VAL4 and CVAL5 are used for PWMB.” *Chapter 36: PWM Synchronizer* *Section 36.5: Functional description*: Added cross-references to *Figure 505: PWM synchronizer (full sync mode)* and *Figure 506: PWM synchronizer (half sync mode)* *Section 36.3.1: Memory map*: Added *Table 571: PWM synchronizer memory map* *Chapter 37: Enhanced Motor Control Timer (eTimer)* *Section 37.2: External signal descriptions*: Removed fault inputs references *Section 37.4.1: Overview*: Added note “The actual availability...” *Section 37.7: Interrupts*: Removed fault inputs references Removed Section 1.2.3: FAULT[n:0] - Fault Inputs *Figure 527: Control Register 1n (CTRL1n)*: Change offset formula to 0x000E + n*0x20 (n=0 to 5) (was n=0 tn 5) *Chapter 38: System Timer Module (STM)* Replaced "system clock" instances by "PBRIDGE clock" throughout the chapter

Table 1372. Document revision history (continued)

Date	Rev.	Changes
12-May-2016	3	*Chapter 39: Software Watchdog Timer (SWT)* *Section 39.1: Introduction*: Added Note “For chip-specific implementation...” *Table 616: SWT_CR field descriptions*: updated description of bit fields MAP[0:7]. *Chapter 40: Periodic Interrupt Timer (PIT)* *Section 40.4: Initialization and application information*: Updated example with device specific values *Chapter 41: CAN Subsystem* *Section 41.3.12: Debug on CAN Support*: Removed Note “To use full ‘Debug on CAN Support’ feature on a M_(TT)CAN instance a DMA channel is required. Please refer to device DMA map to see which M_(TT)CAN instances has got an assigned DMA channel.” *Section 41.3.15: Clock Calibration on M_CAN*: Removed Caution note Removed Section 1.8.1: Shared memory *Figure 566: Data Bit Timing and Prescaler Register (DBTP)* and *Table 633: DBTP Register field descriptions*: Renamed (was FBTP), Moved TCDO bit field to *Section 41.3.5.2.14: Transmitter Delay compensation Register (TDCR)* *Figure 567: Test Register (TEST)* and *Table 634: TEST register field descriptions*: Moved TDCV bit field to *Section 41.3.5.2.13: Protocol Status Register (PSR)*. *Figure 569: CC Control register (CCCR)* and *Table 636: CCCR Register field descriptions*: Added 1 bit field NISO, Removed fields FDBS and FDO, Replaced field FBDS by EFBI, Replaced field FDO by PXHD, Removed CMR, Replaced CME by FDOE and BRSE. *Figure 570: Nominal Bit Timing and Prescaler Register (NBTP)* and *Table 637: NBTP field descriptions*: Renamed (was BTP), renamed field BRP to NBRP, TGSEG1/2 to NTSEG1/2 and SJW to NSJW. *Figure 576: Protocol Status Register (PSR)* and *Table 643: PSR field descriptions*: Moved TDCV bit field to *Section 41.3.5.2.4: Test Register (TEST)* with range increased, added status flag PXE, and renamed field FLEC to DLEC Added *Section 41.3.5.2.14: Transmitter Delay compensation Register (TDCR)* *Figure 578: Interrupt Register (IR)* and *Table 645: IR field descriptions*: Replaced fields STE, FOE, ACKE, BE, CRCE by ARA, PED, PEA *Figure 579: Interrupt Enable Register (IE)* and *Table 646: IE Register field descriptions*: Replaced fields STEE, FOEE, ACKEE, BEE, CRCEE by ARAE, PEDE, PEAE *Figure 580: Interrupt Line Select Register (ILS)* and *Table 647: ILS field descriptions*: Replaced STEL, FOEL, ACKEL, BEL, CRCEL by ARAL, PEDL, PEAL *Figure 611: Rx FIFO Element* and *Table 677: Rx FIFO Element descriptions*: Renamed bit ELD to FDF *Figure 612: Tx Buffer Element* and *Table 678: Tx Buffer Element descriptions*: Added bit ESI (Error State Indicator), bit FDF (FD format) and bit BRS (Bit Rate Switching) Updated text in *Section 41.3.7.3: CAN FD Operation* and in *Section 41.3.7.4: Transceiver Delay Compensation*

Table 1372. Document revision history (continued)

Date	Rev.	Changes
12-May-2016	3	*Figure 631: Calibration Configuration Register (CCFG)* and *Table 690: CCFG Register field descriptions*: Changed SWR field access to RW (was R only). Added values 0x00–0xFF to OPCM field description. Added values 0x04–0x19 to TQBT field description *Section 41.3: Modular CAN (M_CAN) cores*: Added year 2015 to ISO11898-1 *Section 41.8.1: Shared memory map*: Removed note “triggers” in the third sentence *Table 629: IP user manual versions* : Renamed (was “References”) *Figure 572: Timestamp Counter Value Register (TSCV)*, *Figure 574: Timeout Counter Value Register (TOCV)*, *Figure 578: Interrupt Register (IR)*, *Figure 572: Timestamp Counter Value Register (TSCV)* and *Figure 574: Timeout Counter Value Register (TOCV)*: Changed register access to R/W (was w1c) *Figure 578: Interrupt Register (IR)*: Applied w1c to all register fields *Section 41.3.6: Message RAM*: Added note after the first sentence *Table 643: PSR field descriptions*: Updated RFDF field description *Chapter 42: Deserial Serial Peripheral Interface (DSPI)* *Figure 638: DSPI block diagram (1)*: – removed ‘Slave Bus Interface’ and ‘Clock/Reset’ – added footnote “CSI and DSI features are not supported”. *Table 698: DSPI memory map* : – Added Reserved space for ‘0x0004-0x0007’ – Added “Reserved space for 0x4C - 0x7B” *Section 42.3: Register classification for safety*: bullet ‘Safety Latent’ removed footnote “LBIST **not** supported on this device”. *Section 42.4: Functional description*: Added the sentence “SPI — DSPI operates as a basic SPI or a queued SPI.” *Chapter 43: LINFlexD* Updated *Section 43.1: Introduction* with “2.2” version at the end of sentence starting with: “The LINFlexD supports LIN protocol version...”. Updated *Section 43.3.3: Timer* *Table 738: LINFlexD memory map*: removed table footnote (3). *Figure 732: Global Control Register (GCR)*, *Figure 733: UART Preset Timeout Register (UARTPTO)*, *Figure 734: UART Current Timeout Register (UARTCTO)*, *Figure 735: DMA Tx Enable Register (DMATXE)*: Change main address to 0x8C (was 0x4C). *Table 764: DMARXE field descriptions*: Changed TX_CH_NUM to RX_CH_NUM in field column.

Table 1372. Document revision history (continued)

Date	Rev.	Changes
12-May-2016	3	*Chapter 44: SENT Receiver (SRX)* *Section 44.4.2.1: DMA request for Fast Message reading*: Added Note “The data buffer related to the SENT data transferred...” *Table 789: CHn_FMSG_TS field descriptions*: Changed register name to CHn_FMSG_TS (was DMA_FMSG_TS). *Figure 794: Enhanced Serial Message with 8-bit ID field*, *Figure 795: Enhanced Serial Message with 4-bit ID field* and *Figure 796: Short serial message*: Reordered bit numbering from 0 to 31 (was 31 to 0) *Section 44.3.2.20: Channel n Configuration Register (CHn_CONFIG)*: the description for FIL_CNT field has been changed from "This indicates the number" to "This indicates twice the number" *Section 44.4.8: Time stamp logic*: Changed the sentence “For example, if the high frequency receiver clock is 60 MHz, the required prescaler value [...] is 60.” to “For example, if the high frequency receiver clock is 60 MHz, the required prescaler value [...] is 59...” *Section 44.3.2.1: Global Control Register (GBL_CTRL)*: – updated description of Time Stamp Prescaler Value (GBL_CTRL[TSPRSC]). – updated TSPRSC field from value from [1 to 255] to [0 to 255] Added note to *Section 44.3.2.6: Data Control Register 1 (DATA_CTRL1)* and *Section 44.3.2.7: Data Control Register 2 (DATA_CTRL2)* Added two notes in *Section 44.1.2.2: Low power modes* *Section 44.3.2.4: Fast Message Ready Status Register (FMSG_RDY)*: updated note “Reads of bits beyond those for the supported number of channels should be ignored...” *Section 44.4.7.6: Pause pulse diagnostic*: added details as a workaround for unexpected behavior of SENT Calibration equal to pause pulse. Added new paragraphs and tables. *Figure 795: Enhanced Serial Message with 4-bit ID field*: Data [15:12] changed to Data Field [15:12]. *Section 44.1.1: Features*: – changed sentence from "Supports compensation for variation in SENT Tx Clock up to ±25% and adjusts its measurement for drift and jitter in the Transmr clocks per channel" to "Supports compensation for variation in SENT Tx Clock up to ±25% and adjusts its measurement for drift in the Transmitter and Receiver clocks per channel". – changed the maximum supported bit rate from 36µs to 90µs *Section 44.3.2.5: Slow Serial Message Ready Status Register (SMSG_RDY)*: Note changed to "Reads of bits beyond those for the supported number of channels should be ignored. Writes to these bits is ignored by the Receiver". *Table 784: CHn_CLK_CTRL field descriptions*: added a note to the field CM_PRSC. *Section 44.4.6: Input programmable filter*: added value of FIL_CNT in sentence "A timing diagram of the input filter when FIL_CNT is 4 …" *Section 44.3.2.1: Global Control Register (GBL_CTRL)*: NIB_LEN_VAR_LIMIT field is added. *Table 766: SENT memory map*: – updated appearance of register names, and removed “Reserved” rows. – added subtitle rows “CH registers (n = 0 to (CH-1))” and “CH_MSG registers (n = 0 to (CH-1))”

Table 1372. Document revision history (continued)

Date	Rev.	Changes
12-May-2016	3	*Section 44.3.2.4: Fast Message Ready Status Register (FMSG_RDY)*: added note “Under certain conditions, the Single Edge Nibble Transmission (SENT)...“ *Table 770: FMSG_RDY field descriptions*: Added note “These bits are either "write 1 to clear" (w1c) or self-clearing *Table 785: CHn_STATUS field descriptions*: split field description relative to bits 9–15 (MSB=0) into individual field descriptions *Table 786: CHn_CONFIG field descriptions*: split field description relative to bits 9–15 (MSB=0) into individual field descriptions *Section 44.4.1: Initialization sequence*: added notes “When configuring the Single Edge Nibble Transmission (SENT)...“ and “The Successive Calibration Pulse Check Method...” *Section 44.4.5: Adjustment for variation in sensor (Tx) clock*: added note “At ambient temperatures above 114 degree C...”. *Section 44.4.7.5: CRC check*: added note “The Single Edge Nibble Transmitter (SENT) Receiver...” *Section 44.3.2.16: DMA Slow Serial Message Bit2 Read Register (DMA_SMSG_BIT2)*: Replaced field "DATA[11:8]" with "ID3_0_DATA11_8” in register figure and updated field description table accordingly. *Section 44.3.2.25: Channel n Serial Message Bit2 Read Register (CHn_SMSG_BIT2)*: Replaced field "DATA[11:8]" with "ID3_0_DATA11_8” in register figure and updated field description table accordingly. *Figure 788: Channel n Slow Serial Message Bit3 Read Register (CHn_SMSG_BIT3)*: Updated the Offset address. *Figure 767: Global Status Register (GBL_STATUS)* and *Figure 783: Channel n Status Register (CHn_STATUS)*: changed register access to R/W (was w1c). *Section 44.5.3: High frequency receiver clock (protocol clock) requirements*: Added this section. *Figure 765: Global Control Register (GBL_CTRL)*: changed offset to 0x0000 as per value in SENT memory map *Figure 779: DMA Slow Serial Message Bit3 Read Register (DMA_SMSG_BIT3)*: – updated field named “ID7_4_ID3_0” to 2 fields named “ID7_6/ID3_2“ and “ID5_4/ID1_0” – updated field named “ID3_0_DATA15_12” to “ID3_0/DATA15_12” *Figure 780: DMA Slow Serial Message Bit2 Read Register (DMA_SMSG_BIT2)*: – Updated field named “ID[7:6] / ID[3:2]” to “ID3_0/DATA11_8” – Updated field named “DATA[7:0]” to “DATA7_0” *Figure 788: Channel n Slow Serial Message Bit3 Read Register (CHn_SMSG_BIT3)*: – Updated field named “ID[7:6] / ID[3:2]” to “ID7_6/ID3_2” Updated field named “ID[5:4] / ID[1:0]” to “ID5_4/ID1_0” – Updated field named “ID[3:0] / DATA[15:12]” to “ID3_0/DATA15_12” *Figure 789: Channel n Slow Serial Message Bit2 Read Register (CHn_SMSG_BIT2)*: – Updated field named “ID3_0_DATA11_8” to “ID3_0/DATA11_8” – Updated field named “DATA[7:0]” to “DATA7_0” *Figure 778: DMA Fast Message Time Stamp Read Register (DMA_FMSG_TS)* and *Figure 787: Channel n Fast Message Time Stamp Read Register (CHn_FMSG_TS)*: Changed register access from R/W to Read-only.

Table 1372. Document revision history (continued)

Date	Rev.	Changes
12-May-2016	3	*Chapter 45: FlexRay Communication Controller (FlexRay)* Reordered the registers according to the offset address *Section 45.5.2.66.23: Protocol Configuration Register 22 (FR_PCR22)* : Added footnote for bit 15 “Reserved bit or field. Application must not write any value different from the reset value.” *Table 830: FR_SFCNTR field descriptions*: Renamed the bit field Sync Frames Channel A, even cycle from SFEVB to SFEVA *Chapter 46: Reset Generation Module (MC_RGM)* *Figure 965: MC_RGM block diagram*. changed ‘FCCU failure to react reset ‘ into ‘FCCU failure detected by FOSU’ *Table 940: Functional Event Status Register (RGM_FBRE) field descriptions*: modified the description of BE_SOFT_FUNC bit field *Figure 979: Peripheral Reset Register 2 (RGM_PRST2)* and *Table 946: RGM_PRST2 register field descriptions*: Added register field LINFlexD_2_RST at bit 4. *Figure 983: Peripheral Reset Register 6 (RGM_PRST6)* and *Table 950: RGM_PRST6 register field descriptions*: Added register field LINFlexD_3_RST at bit 4. *Figure 987: Peripheral Reset Status Register 2 (RGM_PSTAT2)* and *Table 954: RGM_PSTAT2 register field descriptions*: Added register field LINFlexD_2_STAT at bit 4. *Figure 991: Peripheral Reset Status Register 6 (RGM_PSTAT6)* and *Table 958: RGM_PSTAT6) register field descriptions*: Added register field LINFlexD_3_STAT at bit 4. *Figure 966: Destructive Event Status Register (RGM_DES)* and *Figure 970: Functional Event Status Register (RGM_FES)*: Added footnote “HVD6 occurence will not...”. *Figure 979: Peripheral Reset Register 2 (RGM_PRST2)* and *Figure 983: Peripheral Reset Register 6 (RGM_PRST6)*: Added footnote “RGM is able to reset CAN RAM controller...” *Chapter 47: Boot Assist Flash (BAF)* Inserted CAN information *Section 47.3.4.1: ‘Soft’ DCF clients*: Added Note “*BAF boots with a watchdog timeout window...”* *Section 47.3.4.3: BAF DCF client - DATAX*: added Caution text. Added *Table 967: DATA32 DCF client configuration* Added *Table 968: DATA16 DCF client configuration* and *Table 969: DATA8 DCF client configuration* Added text about “NO_ECHO bit” in sections: – *Section 47.3.5.1: Serial boot mode protocol*:in bullet 2 – *Section 47.3.5.2: LINFlexD download protocol*: in bullet 3 – *Section 47.3.5.5: Download start address, VLE bit, NO_ECHO bit and code size* – *Figure 1001: VLE bit, NO_ECHO bit and code size*: bit NO_ECHO position 30 – *Table 972: LINFlexD serial boot mode download protocol* – *Table 973: CAN serial boot mode download protocol* Updated *Figure 994: Flow of control*

Table 1372. Document revision history (continued)

Date	Rev.	Changes
12-May-2016	3	Updated *Section 47.1: Introduction* *Section 47.4: Resources accessed by BAF code execution*: – Renamed “clocks” to “magic carpet” – Added SSCM to ressource list *Section 47.2: Memory map*: replaced the sentence “The BAF is located in a 16 KB block of flash that is mapped adjacent to the UTEST flash memory block“ by “For the BAF location in flash, refer to the Embedded memories chapter” *Section 47.3.1: Boot modes*: in the bullet “Application code execution”, deleted the sentence “The boot header contains flags that enable/disable individual cores and their reset vectors”. Updated *Section 47.3.2: Device initialization* *Section 47.3.5: Optionally perform a serial boot*: Added the sentence “The BAF is executed via the XOSC as clock source” Replaced all LINFlex occurences by LINFlexD. *Chapter 48: System Status and Configuration Module (SSCM)* Renamed *Table 983: Resulting Life Cycle* and added two rows to this table. *Section 48.3.1: DCF mechanism*: Added Footnote “Refer to the Device Configuration chapter for UTEST_OFFSET value” *Chapter 49: Power management controller digital interface (PMC_Dig)* *Section 49.1: Introduction*: – Removed the following text: “The PMU digital logic.....with the ADC module” – “There is also a custom interface... during initial POR” updated to “There is also a custom interface... to load the trim values” *Section 49.3: IPS bus interface* – “byte(8 bits, data[31:24] or data[23:16] or data[15:8] or data[7:0]) data write/read operations to any registers”, edited to add “except BIST TIME registers (BIST_TIME10, BIST_TIME32, and BIST_TIME65)” – “STA1_PMUCTR_IPS macrocell” updated to “PMC_DIG block” – Instance of “MREG” removed *Section 49.5.1: Event Pending Register (EPR_LV0)*: Removed text “Since the MVD.....disable these UCDs/MVDs/POR” *Figure 1017: Event Pending Register (EPR_LV0)* and *Table 987: EPR_LV0 field descriptions*: bit fields MVD2_FL (bit 11), MVD2_C (bit 15), MVD1_C (bit 23) and PVD0_C (bit 31) made reserved. *Figure 1022: Event Pending Register (EPR_LV1)*: bit fields UVD7_FL (bit 3) and UVD7_C (bit 7) made reserved. *Section 49.5.11: Event Pending Register (EPR_HV0)*: Removed text “Since the MVD.....disable these UCDs/MVDs/POR” *Figure 1027: Event Pending Register (EPR_HV0)*: bit fields MVD10_FL (bit 11), MVD10_C (bit 15), MVD9_C (bit 23) and PVD8_C (bit 31) made reserved. *Section 49.5.16: Event Pending Register (EPR_HV1)*: Removed text “Since the MVD.....disable these UCDs/MVDs/POR” *Figure 1032: Event Pending Register (EPR_HV1)* and *Table 1002: EPR_HV1 field descriptions*: bit field UVD15_C (bit 7) made reserved.

Table 1372. Document revision history (continued)

Date	Rev.	Changes
12-May-2016	3	*Figure 1037: Supply Gauge Status Register (GR_S)* and *Table 1007: GR_S field descriptions*: bit fields VD15_C (bit 0), VD10_FL (bit 12), VD10_C (bit 13), VD9_C (bit 14), VD8_C (bit 15), VD7_FL (bit 16), VD7_C (bit 17), VD2_FL (bit 26), VD2_C (bit 27), VD1_C (bit 29) and VD0_C (bit 31) made reserved. *Table 1007: GR_S field descriptions*: In field “VD6_C” (Bit 19) description, “hvd145_c” updated to “hvd140_c” *Figure 1038: Pending Gauge Status Register (GR_P)* and *Table 1008: GR_P field descriptions*: bit fields VD15_C (bit 0), VD10_FL (bit 12), VD10_C (bit 13), VD9_C (bit 14), VD8_C (bit 15), VD7_FL (bit 16), VD7_C (bit 17), VD2_FL (bit 26), VD2_C (bit 27), VD1_C (bit 29), VD0_C (bit 31) made reserved. *Table 1013: EPR_TD field descriptions* – For field “TEMP1_2” and “TEMP0_2”, Temperature sensor point updated from “150 °C“ to “165 °C” – For field “TEMP1_0”, Temperature sensor point added “-40 °C” *Section 49.7: FCCU Interface*: – “dpmc_fccu_hvd” updated to “HVD_ERROR” – “dpmc_fccu_lvd” updated to “LVD_ERROR” – “dpmc_fccu_tmpsns” updated to “TEMP_ERROR” – “dpmc_ncf” updated to “PMC VD BIST” – “dpmc_safety_error” updated to “DPMC_DCF_SAFETY_ERR” *Figure 1045: Temperature Sensor Configuration register (CTL_TD)*: – bit fields TRIM_ADJ_OVER1[3:0] (bits 4:7), TRIM_ADJ_UNDER1[3:0] (bits 10:13), TRIM_ADJ_OVER0[3:0] (bits 20:23) and TRIM_ADJ_UNDER0[3:0] (bits 26:29) made reserved. – added footnote for bits 19-29: “These bits should not be written” *Table 984: Power Management Controller Memory Map*: – Removed the following registers and their instances, IREGU_CTL_BB, IREGU__DELAY_2_1, IREGU_CTL_SLOPE0....IREGU_CTL_SLOPE10 and IREGU_CTL_UPDATE. – Added register “Voltage selection of IO (VSIO) at offset address 0x0104 – Removed register “IPS Regulator QVF Status register (IREGU_QVF_STATUS) at offset address 0x036C (set to “non_cust” cond. text tag) *Section 49.5.26: Event Pending Register (EPR_TD)*: Removed fields TEMP1_2 and TEMP0_2 and set them to Reserved *Section 49.5.27: Reset Event Enable Register (REE_TD)*: Removed fields TEMP1_2 and TEMP0_2 and set them to Reserved. *Section 49.5.28: Reset Event Select Register (RES_TD)*: Removed fields TEMP1_2 and TEMP0_2 and set them to Reserved. *Section 49.5.30: Temperature Sensor FCCU Event Enable Register (FEE_TD)*: Removed fields FEE1_TS2 and FEE0_TS2 and set them to Reserved. *Section 49.5.34: User BIST Flags Register (BIST_FLAGS)*: Added note with crossreference to Power Managment. *Section 49.5.36: User BIST Time1 and Time0 Register (BIST_TIME10)*, *Section 49.5.37: User BIST Time3 and Time2 Register (BIST_TIME32)* and *Section 49.5.38: User BIST Time6 and Time5 Register (BIST_TIME65)*: – Added a cross-reference to Power Management – Added the sentence “The reset value is calculated based on 70MHz.”

Table 1372. Document revision history (continued)

Date	Rev.	Changes
12-May-2016	3	*Table 1023: BIST_TIME10 field descriptions*, *Table 1024: BIST_TIME32 field descriptions* and *Table 1025: BIST_TIME65 field descriptions*: Added the correspondent timing of reset values for each TIME_*x* field. Added register at *Section 49.5.24: Voltage Selection of IO (VSIO)* with *Figure 1040: Voltage Selection of IO (VSIO)* and *Table 1010: VSIO field descriptions* *Section 49.5.29: Temperature Sensor Configuration register (CTL_TD)*: Removed fields TS2IE1 and TS2IE0 and set bit 0 and bit 16 to Reserved. *Section 49.5.12: Reset Event Enable Register (REE_HV0)*: Removed bit fields REE11_FL and REE11_C. *Section 49.5.13: Reset Event Select Register (RES_HV0)*: Removed bit fields RES11_FL and REE11_C. *Section 49.5.14: Interrupt Enable (IE_HV0)*: Removed bit fields LVD11E_FL and LVD11E_C. *Table 985: 32-bit registers LV0, LV1, HV0, HV1 arrangement table*: Change HV0 / BYTE3 to “VD11=lvd290_adl1_adl0_fl_iol1_osc_c” *Table 986: VD number versus VD name table*: Changed VD11-LVD290_XOSC / VD Meanning to “HV detector asserts when XOSC supply < 2.90V”. *Table 998: REE_HV0 field descriptions*: Removed “PMU” in REE11_OSC field description. *Table 1001: FEE_HV0 field descriptions*: Changed “IO0 PMU” to “XOSC” in REE11_OSC field description. *Figure 1038: Pending Gauge Status Register (GR_P)* and *Table 1008: GR_P field descriptions*: Changed field “VD11_IOL0” to “VD11_OSC” at bit 10. *Figure 1039: Interrupt Enable Global Register (IE_G)* and *Table 1009: IE_G field descriptions*: Changed field “VD11_IOL0” to “VD11_OSC” at bit 11. *Chapter 51: Mode Entry Module (MC_ME)* *Table 1031: MC_ME register description*: added ME_PCTL91 register for LINFlex_2 and ME_PCTL219 register for LINFlex_3 *Figure 1075: Peripheral Status Register 2 (ME_PS2)*, added bit field S_LINFlex_2 *Figure 1079: Peripheral Status Register 6 (ME_PS6)*, added bit field S_LINFlex_3 *Figure 1083: Peripheral Control Register n (ME_PCTLn)*: Address updated to “0x0C0 +n*0x1 (n=0 to 255)”. Updated *Figure 1088: MC_ME transition diagram* *Table 1031: MC_ME register description*: and *Section 51.3.2.26: Core Status Register (ME_CS)*: Renamed register to CS at 0x1C0 (was CS0). *Chapter 52: e200z420n3 Core Debug Support* *Figure 1111: OnCE controller implementation*: added figure title. *Chapter 53: Debug and Calibration Interface (DCI)* Splitted table into *Table 1083: DCI JTAG source selection* and *Table 1083: DCI JTAG source selection* *Section 53.3.2: DCI EVTx pin multiplexing control register (DCI_PINCR)*: Changed the sentence “The DCI_PINCR is a 32-bit data register” into “The DCI_PINCR is a 64-bit data register”.

Table 1372. Document revision history (continued)

Date	Rev.	Changes
12-May-2016	3	*Chapter 56: JTAG Data Communication (JDC)* *Section 56.2: Overview*: removed sentence “The JTAG accessible registers...“ *Section 56.3.1.2: Module status register (JDC_MSR)*: Changed access type of register JDC_MSR to w1c (was RW) *Section 56.3.1.3: JTAG output IPS data register (JDC_JOUT_IPS)*: renamed field “JOUT_IPS Data“ to “DATA“ *Section 56.3.1.4: JTAG input IPS data register (JDC_JIN_IPS)*: renamed field “JIN_IPS Data“ to “DATA“ *Chapter 58: Nexus Port Controller (NPC)* *Section 58.1.2: Features*: Removed bullet “System clock locked status indication via MDO[0] following power-on reset”. *Table 1124: NPC Signal Properties*: Removed table footnote on MDO signal. Removed Section named “System clock locked indication. *Table 1127: PCR field descriptions*: Removed EVT_EN bit field description. *Chapter 61: Nexus Crossbar Multi-Master Client (NXMC)* *Figure 1260: Development control register 2 (DC2)*: Added footnote “This bit is not implemented and it’s value has no impact on the system” to bit 31. *Chapter 62: Cyclic Redundancy Check (CRC) Unit* *Table 1232: CRC unit memory map*: Changed CRC registers per context from (n=1 to 4) to (n=0 to CRC_CNTX_NUM - 1 with CRC_CNTX_NUM = 4). *Figure 1282: CRC computation flow*: Reworked picture (context values) [CRC_COMP_YES] *Section 62.5.1.4: CRC Output Register n (CRC_OUTPn)*: Renamed CRC_OUTP_CHK to CRC_OUTP_CHK*n* *Chapter 63: Memory Error Management Unit (MEMU)* *Table 1251: FLASH_CERR_STSn register field descriptions*: in [BAD_BIT] description, “RAM” replaced by “FLASH”. *Section 63.6.2.14: Flash memory correctable error reporting table status register (FLASH_CERR_STSn)*: Added Note “MEMU captures flash ecc error address on 64bit boundary”. *Chapter 64: Fault Collection and Control Unit (FCCU)* *Section 64.5.2: FSM description*: Updated with “Also, FCCU_CFG_TO ... a configuration register.” *Section 64.7: FCCU Output Supervision Unit*: Updayed with “This signal is automatically de-asserted ...in CONFIG state.” Removed all text referring to SMRT (Safe mode request timer) throughout the chapter. *Section 64.6.21: FCCU XTMR Register (FCCU_XTMR)*: – Updated the second sentence – Updated with “loads (a value corresponding to T_min=250 us + value programmed in FCCU_DELTA_T register) ...which has EOUT signaling enabled.”

Table 1372. Document revision history (continued)

Date	Rev.	Changes
12-May-2016	3	All signal names (such as ipg_clk) are replaced by thier respectives description names (System clock) throughout the chapter. Changed “eTMR” and “ETMR” to “EOUTMR” throughout the chapter. Changed register names from NCF/UF (Noncritical / Unrecovarable Faults) to CF/RF (Critical / Recoverable Faults). *Figure 1329: FCCU RF Status register n (FCCU_RF_Sn)*: Changed reset value from 0x0 to note “The reset value is chip-specific; see the chapter that describes how modules are configured and connected.” *Figure 1330: FCCU RF Key register (FCCU_RFK)* and *Figure 1341: FCCU RF Fake Register (FCCU_RFF)*: Replaced access “Write” by “Write-only”. *Figure 1347: FCCU Permanent Lock Register (FCCU_PERMNT_LOCK)*: Corrected the figure title. *Section 64.5.9: EOUT interface*: Changed sentence “The EOUT frequency is generated ... drives a signal of 61 Hz on EOUT.” by sentence “The EOUT frequency is generated by dividing IRCOSC clock by ((FOP + 1) * 2 * 2048) where FOP vary between 0 and 63. With an IRC clock of 16 MHz, this drives...” *Chapter 65: Self-Test Control Unit (STCU2)* *Table 1335: STCU_LB_CTRLn field descriptions*: For bit CWS, configuration “000”, description changed from “illegal” to “controller waits 16 shift cycles for capture to finish” *Table 1309: STCU_PLL_CFG field descriptions* : – For bit field PLLODF, description updated to add “Refer to the RFDPHI field of the PLL0 Divider Register (PLL0DIV)….” – For bit field PLLIDF, description updated to add “Refer to the PREDIV field of the PLL0 Divider Register (PLL0DIV)….” – For bit field PLLLDF, description updated to add “Refer to the MFD field of the PLL0 Divider Register (PLL0DIV)….” Updated *Section 65.5.3: STCU2 SK Code Register (STCU_SKC)* Renamed all instantiated register headings to the format: STCU2 <description> *n* Register (<reg_abbr>*n*) *Chapter 67: Password and Device Security Module (PASS)* Added new *Table 1363: Debug Interface Enable Truth Table* *Section 67.2.1.7: Password Group n - Lock 2 status register (LOCK2_PGn)*: Renamed bit field into A256KLOCK. *Section 67.2.1.8: Password Group n - Lock 3 status register (LOCK3_PGn)*: Renamed bit field into A256KLOCK. *Table 1353: PASS memory map*: – Changed LOCK3_PG*n* reset values to 0xXFXXXXXX (was 0xXXXXXXXX). – Changed LCSTAT reset value to 0xX0000007 (was 0x000000XX). *Chapter Appendix A: Acronyms and abbreviations* Added Terms: BLDC, FOC, WBBR, AC, ACIM, BDC, DC, PMAC, PWM, SRM and VRM

Table 1372. Document revision history (continued)

Date	Rev.	Changes
12-Jun-2018	4	*Chapter 2: Introduction* *Section 2.2: Features* – “Up to -40 °C to 130 °C ambient .....range” updated to “Up to -40 °C to 150 °C junction....range” – “Supports internal flash...(UART)” updated to “Supports internal flash...(UART and MCAN)” *Table 2: SPC574Sx device feature summary (superset configuration)* – Instances of LFBGA244 conditioned with “non_cust” – Table footnote updated from “Including trace for the crossbar masters (data & instruction trace on core and data trace on eDMA).10 MDO pin Nexus..... and LFBGA244” to “Including trace for the crossbar masters (data & instruction trace on core and data trace on eDMA). 4 MDO pin Nexus..... for QFP144.” *Section 2.4: Packages* – Remove LFBGA244 *Chapter 3: System architecture* *Figure 4: CPU Exceptions vs. Interrupt Controller* – Updated bitline by 14 (was 13). – Added a new note “Note5: IVPR[16:0] + p_voffset[13:0]+2'b00”. Updated the CPU address of HW mode “ IVPR[15:0] + p_voffset[12:0] + 00b” to “ IVPR[23:0] + 0x00001000 + IACKR[31:20] *Section 3.3: Motor control Architecture* – Updated by adding a new point “For each ADC: 4 configurable analog watchdogs.” – Replaced “Resolution: 12 bit, ENOB: 10 bit” by “Resolution: 12 bit.” *Figure 7: Motor control architecture* – DFT, PTCH blocks, and their connections are removed. *Table 7: ADC channels mapping description* – Column PADLBFGA244 deleted. *Chapter 4: Signal description* *Table 9: Power supply and reference pins* – Removed column “LFBGA244”, and remove pin no. 77 from eTQFP144 for VDD_LV pad. *Table 10: System pins* – Removed column “LFBGA244”. *Table 11: Debug and trace pins* – Removed column “LFBGA244”. *Table 12: Pin multiplexing table description* – Table footnote is updated from “ This is..... for eTQFP144 and LFBGA244.” to This is..... for eTQFP144 only”. *Table 13: Peripheral multiplexing table description* – Added linflex_2 and linnflex_3 value in the table. *Table 14: Cross-mapping between pads and package pins* – Removed column “LFBGA244”.

Table 1372. Document revision history (continued)

Date	Rev.	Changes
12-Jun-2018	4 (cont'd)	*Chapter 5: Memory map* *Table 15: Memory map (XBAR slave ports)* – Updated System RAM from 96 KB RAM to 128 MB RAM for 0x4000_0000 (Start address) and 0x4001_7FFF (End address). – Updated table footnote, “In order to avoid ECC errors.....address within the upper DMEM window” to “In order to avoid ECC errors..... whenever there is an access to the DMEM mirrored space, the address feedback alarm gets set. It will be reported to FCCU as ncf[70].” *Table 18: Peripheral bridge 0 memory map* – Updated End address from 0xFFED_FFFF to 0xFFED_3FFF. *Chapter 6: Functional safety* *Table 22: ECC Engine Summary* – Updated the value of FlexRay CHI RAM I/F for Correction and Detection column. *Table 26: LBIST / MBIST configurations* – Updated the table. *Chapter 7: Device configuration* Created *Section 7.5.2: Decorated Storage Memory Controller (DSMC)*. – Extensive changes are made throughout the chapter. *Chapter 9: Device Configuration Format (DCF) Records* – Editorial and formatting changes. – *Figure 51: Appending DCF records*, editting done in the diagram from”Data Record - CS2, Ad = 1” to “Data Record - CS2, Ad = 0” *Chapter 14: System Integration Unit Lite2 (SIUL2)* *Figure 98: SIUL2 MCU ID Register #1 (SIUL2_MIDR1)*: – In the footnote removed the default value of PKG bit field. *Table 240: SIUL input trigger connections to the INTC channels*: – Updated external interrupt number column from “0 to 3, 4 to 7, 8 to 11, 12 to 15, 16 to 19, 20 to 23, 24 to 27, 28 to 31” to “0 to 7, 8 to 15, 16 to 23, 24 to 31”. *Chapter 21: DMA channel multiplexer (DMACHMUX)* – Updated *Figure 194: DMACHMUX triggered channels* *Chapter 22: Wakeup unit (WKPU)* – Updated *Figure 197: Wakeup unit Block Diagram* – *Section 22.3.2.1: NMI Management*: Removed Note “An edge enable for a given input must be asserted before the NMI/external interrupt edge in order for a system wakeup or interrupt to be generated.”

Table 1372. Document revision history (continued)

Date	Rev.	Changes
12-Jun-2018	4 (cont'd)	*Chapter 24: Clocking* *Figure 201: Clock generation architecture*: – Removed the block of Re-sync stage. – Added two blocks linflex2_clk and linflex3_clk in periph clock for DIV1-16 *Table 299: Maximum system level clock frequencies*: – Added the value of Max protocol clock 40 MHz. – Added the value of Max host clock 70 MHz. *Section 24.5.2: External oscillator (XOSC)*: – “The XOSC provides support for 8 – 44 MHz crystal....resistor.” updated to “The XOSC provides support for 8 – 40 MHz crystal....resistor.” – Added a note, “The user must take care when using 16 MHz crystals or lower, because the S_XOSC flag in the ME_GS register will not be set if the CMU_PLL (CMU 0) default configuration is used. Indeed CMU_PLL (CMU 0) is programmed by default to use crystal frequencies greater than IRC (16 MHz) + 0.5 MHz and this condition is no longer valid with XOSC <= 16 MHz. In this case, to use an XOSC frequency less than or equal to 16 MHz, an RCDIV value (CMU_0.CSR register) that respects the previous formula (CMU_CSR.B.RCDIV = 3) must be used. This setting has to be done before enabling XOSC in the user code.” *Section 24.5.2.1: XOSC startup*: – “The XOSC.....or 40 MHz to 44 MHz crystal. The XOSC EN_40 MHz.....crystal (UTEST Misc bit 21 = 0).” updated to “The XOSC.....or 40 MHz crystal. The XOSC EN_40 MHz.....crystal (UTEST Misc bit 21 = 0).” *Figure 205: CMU_0 block diagram* and *Figure 206: CMU[1:4] block diagram*: – Deleted “$f_{CLKMN1} < f_{CLKMT0_RMN} / 4$” from CLKMN1 Supervisor part. *Chapter 25: Dual PLL Digital Interface (PLLDIG)* *Table 308: PLL0DV field descriptions* and *Table 311: PLL1DV field descriptions*: – Updated division factors of RFDPHI. *Table 312: PLL1FM field descriptions*: – Added a cross-reference to Frequency modulation section for MODPRD and INCSTP field descriptions. *Table 313: PLL1FD field descriptions*: – Added a cross-reference to Clock configuration section for FRCDIV field description. – Updated DTHDIS field description. – Added note to FRCDIV field description. *Section 25.6.2: Clock configuration*: – Added text “refer to the device datasheet” – Added two notes. *Section 25.6.4: Frequency modulation*: – Changed feedback divider f_{ref} by $f_{pll1ref}$ – Move text “PLLFM[MODPRD] and PLLFM[INCSTP] are subject to the following restriction:” to next equation. – Replaced PLLnCR[MFD] with PLL1DV[MFD].

Table 1372. Document revision history (continued)

Date	Rev.	Changes
12-Jun-2018	4 (cont'd)	*Chapter 26: Clock Monitor Unit (CMU)* *Table 316: CMU_CSR field descriptions* – SFM: added sentences “MDR[MD] ... CSR[SFM]=1” and “Software ... bit” – RCDIV: added sentence “Ensure ... [RCDIV]” – CME: added sentence “CMU_HFREF ... (CSR[CME=1])” *Section 26.5.2: CLKMN0_RMT supervisor*: – Deleted Note “f_{CLKMN0_RMT} ... *monitoring.*” *Chapter 27: Clock Generation Module (MC_CGM)* *Table 322: MC_CGM register description*: – Updated offset address for PCS Divider Start for XOSC (CGM_PCS_DIVE1), PCS Divider Start for XOSC (CGM_PCS_DIVS1), PCS Divider Start for XOSC (CGM_PCS_DIVE2), PCS Divider Start for XOSC (CGM_PCS_DIVS2), PCS Divider Start for XOSC (CGM_PCS_DIVE4), PCS Divider Start for XOSC (CGM_PCS_DIVS4). – Updated offset address in *Section 27.3.1.3: PCS Divider End 1 Register (CGM_PCS_DIVE1)*, *Section 27.3.1.4: PCS Divider Start 1 Register (CGM_PCS_DIVS1)*, *Section 27.3.1.6: PCS Divider End 2 Register (CGM_PCS_DIVE2)*, *Section 27.3.1.7: PCS Divider Start 2 Register (CGM_PCS_DIVS2)*, *Section 27.3.1.9: PCS Divider End 4 Register (CGM_PCS_DIVE4)*, *Section 27.3.1.10: PCS Divider Start 4 Register (CGM_PCS_DIVS4)*. *Chapter 32: Embedded Flash Memory* *Section 32.3.2.13: Test Mode Disable Password Check register (TMD)*: – Deleted text between brackets “Test Mode Disable Override Passcode at address offset 0x3DC8” *Table 416: TDM field descriptions*: – Added Note “PWD bitfield is byte-reversed...” Added *Chapter 33: Decorated Storage Memory Controller (DSMC)*. *Chapter 34: Successive Approximation Register Analog-to-Digital Converter (SARADC)* *Section 34.4.5.1: CTU trigger mode*: – Replaced JABORT by JABORTCHAIN (was a typo) *Table 423: MCR field descriptions*: – Added note in the description of CTUEN – Removed note in CTUEN bitfield Replaced SDADC by SARADC in FRZ bitfield description (was a typo)

Table 1372. Document revision history (continued)

Date	Rev.	Changes
12-Jun-2018	4 (cont'd)	*Chapter 35: Enhanced Cross Trigger Unit (ECTU)* *Section 35.1: Introduction*: – "The ECTU...command to the ADC, trigger signals to other IPs and external trigger...lost." updated to "The ECTU ...commands to the ADCs, start/stop trigger signals to other IPs and external trigger to the pad...lost. *Section 35.2: Main features*: – Added "Receives up to 8 trigger events from other IPs such as PWM, Timers, external pins" to "Receives up to 16 input trigger events from other IPs such as PWM, Timers, external pins . For mapping details, refer to *Figure 33: Module to module connections* and *Table 13: Peripheral multiplexing table description*" – "Generates.....8 trigger events" updated to "Generates..8 internal events to trigger ADC sequences" – "Each trigger...conversion" updated to "Each internal trigger...conversion" Multiple Conversion Mode of *Section 35.6.1: ADC commands list format* : – Updated the description from "Maximum...CLR_0 will.....CLR_1 and so on..FIFO." to "Maximum...CLR_1 will.....CLR_2 and so on...FIFO." Updated *Section 35.10.18: Commands list register n (CLRn)*: – Offset: 0x120 +n*0x4 (n=0 to 63)" to " Offset: 0x11C +n*0x4 (n=1 to 64)." *Table 477: SU registers*: – Updated Register section of Address offset 0x0120- 0x02AC from "Commands List Register 0–63 (CLR0–CLR63)" to "Commands List Register 1–64 (CLR1–CLR64)". *Table 481: TGSCR field descriptions*: – Updated the descriptive section of MRS_SM field from " MRS...(5 bits to select one of 32 inputs)." to "MRS...(4-bits to select one of 16 input trigger signals)." *Figure 414: Cross triggering unit expected value n (CTU_EXPECTEDn)*: – Changed constant offset address to 0x051C (was 0x0520) *Figure 415: Cross triggering unit counter range (CTU_CNT_RANGEn)*: – Changed constant offset address to 0x052C (was 0x0530)

Table 1372. Document revision history (continued)

Date	Rev.	Changes
12-Jun-2018	4 (cont'd)	*Chapter 42: CAN Subsystem* *Table 637: CCCR field descriptions*: – Removed note in NISO register field and added notes to PXHD and BRSE register fields. *Figure 569: CAN subsystem generic block diagram*: – Rewrite figure note 1. *Figure 579: Timestamp Counter Value Register (TSCV)* and *Figure 581: Timeout Counter Value Register (TOCV)*: – Changed TSC/TOC field access to R/W (was R/w1c). *Section 42.3.15: Clock Calibration on CAN Unit (CCCU)*: – Added note. *Section 42.3.6: Message RAM* and *Figure 617: Message RAM Configuration*: – Set Message RAM to maximum value of 4352 words and added reference to specific device configuration. *Section 42.5: SRAM interface and memory organization*: – Added cross- reference to Shared memory map section. *Chapter 48: Boot Assist Flash (BAF)* *Section 48.2: Memory map*: – Added a note (removed from the PASS chapter's Clock Jitter activation section and copied here) Updated *Table 964: BAF memory organization*: – Updated the value of BAF_data_area_and_stack. Updated *Table 965: BAF image header* – Updated the value of address offset from 0010h to 0x0008 for the field BAF version number. *Chapter 49: System Status and Configuration Module (SSCM)* *Section 49.3.5: Life Cycle*: – Changed the sentence starting with "The LC is written into 5 slots, 128 bits each,..." by "The LC is written into 5 slots, 256 bits each,..." – Changed the sentence ending with "each LC slot can have one of the possible 4 status:" by "each LC slot can have one of the possible 3 status:" – Updated *Table 983: Life Cycle slots* and *Table 984: Life Cycle Slots in Memory* – Updated *Figure 1020: Appending DCF records* – *Table 985: Resulting Life Cycle*, changed back the title.

Table 1372. Document revision history (continued)

Date	Rev.	Changes
12-Jun-2018	4 (cont'd)	*Chapter 55: Debug and Trace* *Table 1100: Nexus signals*: – Updated the footnote. *Chapter 59: Nexus Port Controller (NPC)* Updated *Section 59.1.3.4: Reduced-Port Mode* – Added “All MSEO signals need to be connected” to the content. *Chapter 65: Fault Collection and Control Unit (FCCU)* *Table 1272: FCCU_CFG field descriptions*: – Updated description of bitfield FCCU_SET_CLEAR *Chapter 66: Self-Test Control Unit (STCU2)* *Table 1310: STCU_RUNSW field descriptions*: – Moved this table to above section. *Table 1312: STCU_CFG field descriptions*: – Added note “STCU2 is being clocked from PBRIDGE Clock. Please refer Clocking chapter for PBRIDGE Clock Frequency value” for “LB_DELAY” field. *Table 1314: STCU_WDG field descriptions*: – Replaced table footnote by table note for “STCU2 is being clocked from PBRIDGE Clock. Please refer Clocking chapter for PBRIDGE Clock Frequency value.” *Chapter 68: Password and Device Security Module (PASS)* *Table 1358: LCSTAT register field descriptions*: – Updated description of LIFE bitfield with value 110 (ST production)

IMPORTANT NOTICE – PLEASE READ CAREFULLY

STMicroelectronics NV and its subsidiaries (“ST”) reserve the right to make changes, corrections, enhancements, modifications, and improvements to ST products and/or to this document at any time without notice. Purchasers should obtain the latest relevant information on ST products before placing orders. ST products are sold pursuant to ST’s terms and conditions of sale in place at the time of order acknowledgement.

Purchasers are solely responsible for the choice, selection, and use of ST products and ST assumes no liability for application assistance or the design of Purchasers’ products.

No license, express or implied, to any intellectual property right is granted by ST herein.

Resale of ST products with provisions different from the information set forth herein shall void any warranty granted by ST for such product.

ST and the ST logo are trademarks of ST. All other product or service names are the property of their respective owners.

Information in this document supersedes and replaces information previously supplied in any prior versions of this document.

© 2018 STMicroelectronics – All rights reserved