RM0349
Reference manual

SPC570Sx 32-bit Power Architecture® microcontroller for automotive ASILD applications

Overview

This document defines the functionality of the SPC570Sx microcontroller for use by software and hardware developers. The SPC570Sx is built on Power Architecture® technology and integrates technologies that are important for today's and tomorrow's automotive applications. SPC570Sx targets applications such as ABS and airbag, both of which require a high safety integrity level.

The information in this book is subject to change without notice, as described in the disclaimer. 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 SPC570Sx 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 75

1.1 Register conventions 75

1.2 Acronyms and abbreviations 75

1.3 Reference documents 76

2 Introduction 77

2.1 SPC570Sx microcontroller 77

2.1.1 Target applications 77

2.1.2 Overall architecture 77

2.1.3 Core features 77

2.1.4 Memory hierarchy 77

2.2 Features 78

2.3 Feature summary 79

2.4 Packages 80

2.5 Block diagram 80

3 Embedded memories 82

3.1 Overview 82

3.2 SRAM 82

3.3 Flash memory 82

3.3.1 Flash memory controller 83

3.3.2 Flash memory array 83

3.4 Security features 85

4 Signal description 86

4.1 Pin descriptions 86

4.1.1 Pin startup and reset states 86

4.1.2 Power supply and reference voltage pins 86

4.1.3 System pins 87

4.1.4 Pin multiplexing 88

4.2 Package pads/pins 140

5 Memory map 144

5.1 Architecture . . . 144
5.2 Memory map . . . 144

6 Functional safety . . . 150
6.1 Introduction . . . 150
6.2 Assumptions and usage constraints . . . 150
6.3 Module categorization . . . 151
6.4 Safety architecture overview . . . 152
6.4.1 Lockstep . . . 153
6.4.2 End to end (E2E) protection on the data path and memory ECC . . . 155
6.4.3 I/O safety . . . 162
6.4.4 Clock . . . 162
6.4.5 Power . . . 163
6.4.6 Failure handling . . . 163
6.4.7 Failure indication . . . 174
6.4.8 Latent fault detection . . . 175
6.4.9 Common Cause Failures (CCFs) . . . 175
6.4.10 Resets . . . 175
6.4.11 LBIST/MBIST architecture partitioning . . . 176

7 Device configuration . . . 182
7.1 Introduction . . . 182
7.2 Core modules . . . 182
7.2.1 Core reset settings . . . 182
7.2.2 Special purpose register (SPR) summary . . . 184
7.3 System modules . . . 185
7.3.1 SIUL2 configuration . . . 185
7.3.2 Crossbar switch . . . 185
7.3.3 System Memory Protection Unit (SMPU) configuration . . . 188
7.3.4 Peripheral bridge configuration . . . 189
7.3.5 Interrupt controller (INTC) configuration . . . 190
7.3.6 DMA controller configuration . . . 201
7.3.7 DMA channel multiplexer (DMACHMUX) configuration . . . 202
7.3.8 Wakeup unit (WKPU) configuration . . . 204
7.3.9 Crossbar integrity checker (XBIC) . . . 204
7.4 Clocking . . . 205

7.5 Memories and memory interfaces . . . 205
7.5.1 Flash memory controller (PFLASH) configuration . . . 205
7.6 Analog modules . . . 206
7.6.1 Temperature sensor configuration . . . 206
7.6.2 SARADC configuration . . . 207
7.6.3 CTU . . . 207
7.7 Timers . . . 207
7.7.1 System Timer Module (STM) configuration . . . 207
7.7.2 Software Watchdog Timer (SWT) configuration . . . 208
7.7.3 PIT configuration . . . 208
7.7.4 eTimers . . . 210
7.8 Communication interfaces . . . 210
7.8.1 FlexCAN . . . 210
7.8.2 DSPI configuration . . . 211
7.8.3 LINFlexD . . . 213
7.9 Reset and Boot modules . . . 215
7.9.1 BAF configuration . . . 215
7.9.2 System Status and Configuration Module (SSCM) configuration . . . 215
7.9.3 Self-Test Control Unit (STCU2) configuration . . . 216
7.10 Safety modules . . . 216
7.10.1 CRC – Number of contexts . . . 216
7.10.2 MEMU configuration . . . 217
7.10.3 FCCU configuration . . . 224
7.10.4 Register protection (REG_PROT) configuration . . . 225
7.11 Safety modules . . . 232
7.11.1 Password and Device Security Module (PASS) configuration . . . 232
7.11.2 Debug and trace . . . 236

8 Reset and boot . . . 237
8.1 Introduction . . . 237
8.1.1 Architecture . . . 237
8.1.2 Functionality . . . 237
8.2 Hardware reset stage . . . 238
8.2.1 Destructive reset events . . . 238
8.2.2 Functional reset sources . . . 239
8.2.3 Functional reset escalation . . . 240

8.2.4 Destructive reset escalation . . . 240
8.3 Reset sequence . . . 241
8.4 PORST pin functionality . . . 243
8.5 ESR0 pin functionality . . . 245
8.6 PORST and ESR0 pin status during power on, destructive and functional reset . . . 246
8.7 System boot . . . 246
8.8 HW boot record search (SSCM search) . . . 247
8.9 SW boot record search (BAF search) . . . 249
8.10 Performances . . . 250

9 Device Configuration Format (DCF) Records . . . 251
9.1 Introduction . . . 251
9.2 DCF clients . . . 251
9.3 DCF records . . . 252
9.3.1 UTEST DCF records . . . 255
9.4 DCF client table . . . 255

10 Power management . . . 265
10.1 Overview . . . 265
10.2 Architecture . . . 267
10.2.1 Power management modules . . . 267
10.2.2 Power management supply description . . . 268
10.3 Functionality . . . 269
10.3.1 Regulators capabilities . . . 269
10.3.2 1.2 V supply . . . 269
10.3.3 High voltage supply . . . 270
10.4 Voltage monitors . . . 271
10.4.1 Nomenclature . . . 271
10.4.2 Voltage monitor summary . . . 271
10.4.3 LV monitors . . . 272
10.4.4 HV monitors . . . 273
10.4.5 Voltage monitor digital interface . . . 273
10.4.6 Safety requirements . . . 276
10.4.7 BIST details . . . 276

11 Security 279

11.1 Basic security 279

11.2 Advanced security 279

11.3 Detailed security information 279

12 e200z0h Core description 280

12.1 Overview 280

12.2 Microarchitecture summary 280

12.3 Block diagram 282

12.4 Features 283

12.4.1 Instruction unit features 283

12.4.2 Integer unit features 284

12.4.3 Load/Store unit features 284

12.4.4 e200z0h system bus features 284

12.4.5 Nexus 2+ features 284

12.5 Core registers and programmer's model 285

13 System Integration Unit Lite2 (SIUL2) 288

13.1 Introduction 288

13.1.1 Overview 288

13.1.2 Features 290

13.1.3 Register protection 290

13.2 Memory map and register description 290

13.2.1 Memory map 290

13.2.2 Register descriptions 293

13.3 Functional description 310

13.3.1 General 310

13.3.2 Pad control 310

13.3.3 General purpose input or output pads (GPIO) 311

13.3.4 External interrupts/DMA requests (EIRQ pins) 312

14 Crossbar switch (XBAR) 315

14.1 Introduction 315

14.1.1 Features 315

14.1.2 Memory map and register definition 315

14.2 Functional description 320

14.2.1 General operation . . . 320
14.2.2 Register coherency . . . 321
14.2.3 Arbitration . . . 321
14.3 Initialization/application information . . . 322

15 Crossbar Integrity Checker (XBIC) . . . 323
15.1 Overview . . . 323
15.2 Features . . . 323
15.3 Block diagram . . . 324
15.4 External signal description . . . 325
15.5 Memory map and register definition . . . 325
15.5.1 Memory map . . . 325
15.5.2 Register descriptions . . . 326
15.6 Functional description . . . 331

16 Peripheral bridge . . . 334
16.1 Introduction . . . 334
16.1.1 Features . . . 334
16.1.2 General operation . . . 334
16.2 Memory map/register definition . . . 334
16.2.1 Master Privilege Registers (PBRIDGE_MPRA) . . . 337
16.2.2 Peripheral Access Control Registers (PBRIDGE_PACR*x*) . . . 338
16.2.3 Off-Platform Peripheral Access Control Registers (PBRIDGE_OPACR*x*) . . . 339
16.3 Functional description . . . 343
16.3.1 Access support . . . 343

17 System Memory Protection Unit (SMPU) . . . 344
17.1 Introduction . . . 344
17.2 Block diagram . . . 344
17.3 Features . . . 346
17.4 Modes of operation . . . 346
17.5 External signal description . . . 346
17.6 Memory map and register definition . . . 347
17.6.1 SMPU Control/Error Status Register (SMPU_CESR) . . . 347
17.6.2 SMPU Error Address Register, Slave Port n (SMPU_EARn) . . . 348

17.6.3 SMPU Error Detail Register, Slave Port n (SMPU_EDRn) 349
17.6.4 SMPU Region Descriptor n (SMPU_RGD) 350
17.6.5 SMPU Region Descriptor Alternate Access Control n (SMPU_RGDAACn) 354
17.7 Functional description 355
17.7.1 Access evaluation macro 355
17.7.2 Putting it all together and AHB error terminations 359
17.8 Functional safety 360
17.9 Initialization information 360
17.10 Application information 360

18 Interrupt Controller (INTC) 363
18.1 Introduction 363
18.2 Block diagram 363
18.3 Features 365
18.4 Modes of operation 365
18.4.1 Software vector mode 365
18.5 Memory map and register definition 365
18.5.1 Memory map 365
18.5.2 Register descriptions 366
18.6 Functional description 372
18.6.1 Interrupt request sources 372
18.6.2 Priority management 373
18.6.3 Handshaking with processor 374
18.7 Initialization/application information 376
18.7.1 Initialization flow 376
18.7.2 Interrupt exception handler 376
18.7.3 ISR, RTOS, and task hierarchy 377
18.7.4 Order of execution 378
18.7.5 Priority ceiling protocol 379
18.7.6 Selecting priorities according to request rates and deadlines 382
18.7.7 Software-settable interrupt requests 383
18.7.8 Lowering priority within an ISR 383
18.7.9 Negating an interrupt request outside of its ISR 383
18.7.10 Examining LIFO contents 384
18.8 Interrupt sources 384

19 Enhanced direct memory access (eDMA) 385
19.1 Introduction 385
19.1.1 Features 385
19.2 Modes of operation 386
19.2.1 Normal mode 386
19.2.2 Debug mode 387
19.2.3 Wait mode 387
19.3 Memory map/register definition 387
19.3.1 Control Register (DMA_CR) 389
19.3.2 Error Status Register (DMA_ES) 391
19.3.3 Enable Request Register (DMA_ERQL) 393
19.3.4 Enable Error Interrupt Register (DMA_EEIL) 394
19.3.5 Set Enable Request Register (DMA_SERQ) 396
19.3.6 Clear Enable Request Register (DMA_CERQ) 397
19.3.7 Set Enable Error Interrupt Register (DMA_SEEI) 398
19.3.8 Clear Enable Error Interrupt Register (DMA_CEEI) 398
19.3.9 Clear Interrupt Request Register (DMA_CINT) 399
19.3.10 Clear Error Register (DMA_CERR) 399
19.3.11 Set START Bit Register (DMA_SSRT) 400
19.3.12 Clear DONE Status Bit Register (DMA_CDNE) 401
19.3.13 Interrupt Request Register (DMA_INTL) 401
19.3.14 Error Register (DMA_ERRL) 403
19.3.15 Hardware Request Status Register (DMA_HRSL) 405
19.3.16 Channel *n* Priority Register (DMA_DCHPRI*n*) 407
19.3.17 Channel *n* Master ID Register (DMA_DCHMID*n*) 407
19.3.18 TCD Source Address (DMA_TCD*n*_SADDR) 408
19.3.19 TCD Transfer Attributes (DMA_TCD*n*_ATTR) 409
19.3.20 TCD Signed Source Address Offset (DMA_TCD*n*_SOFF) 410
19.3.21 TCD Minor Byte Count (Minor Loop Disabled) (DMA_TCD*n*_NBYTES_MLNO) 410
19.3.22 TCD Signed Minor Loop Offset (Minor Loop Enabled and Offset Disabled) (DMA_TCD*n*_NBYTES_MLOFFNO) 411
19.3.23 TCD Signed Minor Loop Offset (Minor Loop and Offset Enabled) (DMA_TCD*n*_NBYTES_MLOFFYES) 412
19.3.24 TCD Last Source Address Adjustment (DMA_TCD*n*_SLAST) 413
19.3.25 TCD Destination Address (DMA_TCD*n*_DADDR) 414
19.3.26 TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD*n*_CITER_ELINKYES) 414

19.3.27 TCD Current Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD*n*_CITER_ELINKNO) . . . 415
19.3.28 TCD Signed Destination Address Offset (DMA_TCD*n*_DOFF) . . . 416
19.3.29 TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD*n*_DLASTSGA) . . . 417
19.3.30 TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD*n*_BITER_ELINKYES) . . . 417
19.3.31 TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD*n*_BITER_ELINKNO) . . . 418
19.3.32 TCD Control and Status (DMA_TCD*n*_CSR) . . . 419
19.4 Functional description . . . 421
19.4.1 eDMA microarchitecture . . . 422
19.4.2 eDMA basic data flow . . . 423
19.4.3 Error reporting and handling . . . 426
19.4.4 Channel preemption . . . 428
19.4.5 eDMA performance . . . 428
19.5 Initialization/application information . . . 431
19.5.1 eDMA initialization . . . 431
19.5.2 DMA programming errors . . . 433
19.5.3 DMA Arbitration mode considerations . . . 433
19.5.4 DMA transfer . . . 434
19.5.5 eDMA TCD*n* status monitoring . . . 437
19.5.6 Channel linking . . . 438
19.5.7 Dynamic programming . . . 439

20 DMA channel multiplexer (DMACHMUX) . . . 443
20.1 Introduction . . . 443
20.1.1 Overview . . . 443
20.1.2 Features . . . 444
20.1.3 Modes of operation . . . 444
20.2 External signal description . . . 444
20.2.1 Overview . . . 444
20.3 Memory map and register definition . . . 444
20.3.1 Register descriptions . . . 445
20.4 Functional description . . . 446
20.4.1 DMA channels with periodic triggering capability . . . 446
20.4.2 DMA channels with no triggering capability . . . 448
20.4.3 “Always Enabled” DMA sources . . . 448

20.5 Initialization/Application information 449
20.5.1 Reset 449
20.5.2 Enabling and configuring sources 449

21 Wakeup unit (WKPU) 454
21.1 Introduction 454
21.1.1 Overview 454
21.1.2 Features 454
21.2 Memory Map and Register Description 455
21.2.1 Memory Map 455
21.2.2 Register Descriptions 455
21.3 Functional Description 460
21.3.1 General 460
21.3.2 Non-Maskable Interrupts 460

22 Temperature sensor (TSENS) 461
22.1 Introduction 461
22.2 Features 461
22.3 Signals 461
22.4 TSENS outputs 461
22.5 Obtaining device temperature using TSENS 462
22.5.1 TSENS calibration constants 462
22.5.2 Equations for converting TSENS voltage to device temperature 462
22.6 Equations for converting CTAT and PTAT voltages into constant reference code 463

23 Clocking 464
23.1 Introduction 464
23.2 Clock generation 464
23.2.1 MC_CGM registers 467
23.3 System clock frequency limitations 467
23.3.1 JTAG frequencies 468
23.4 Default clock configuration 468
23.5 Clock sources 469
23.5.1 PLL 469
23.5.2 External oscillator (XOSC) 471

23.5.3 16 MHz internal RC oscillator (IRCOSC) 472
23.6 Peripheral clocks 472
23.6.1 FlexCAN clocking 474
23.6.2 System clock output pin 474
23.7 Clock monitoring 474
23.7.1 CMU configuration 474
23.7.2 PLL0 monitor 476
23.7.3 External oscillator (XOSC) monitor 476
23.7.4 Internal RC oscillator (IRCOSC) monitor 476
23.7.5 System clock monitors 477
23.8 Loss of system clock behavior 477
23.8.1 Loss of PLL/XOSC clock 477
23.8.2 Loss of IRCOSC clock 477

24 Dual PLL Digital Interface (PLLDIG) 478
24.1 Introduction 478
24.2 Block diagram 478
24.3 Features 478
24.4 Modes of operation 478
24.4.1 Normal mode with reference, PLL0 or both PLLs enabled 479
24.5 Memory map and register definition 479
24.5.1 Memory map 479
24.5.2 Register descriptions 480
24.5.3 Register classification for safety requirements 488
24.6 Functional description 488
24.6.1 Input clock frequency 489
24.6.2 Clock configuration 489
24.6.3 Loss of lock 490
24.6.4 Frequency modulation 490
24.7 Initialization information 492

25 Clock Monitor Unit (CMU) 494
25.1 Introduction 494
25.1.1 Main features 494
25.2 Block diagram 494
25.3 Signals 495

25.4 Memory map and register definition 495
25.4.1 Register descriptions 496
25.5 Functional description 500
25.5.1 Frequency meter 500
25.5.2 CLKMN0_RMT supervisor 500
25.5.3 CLKMN1 supervisor 500

26 Clock Generation Module (MC_CGM) 502
26.1 Introduction 502
26.1.1 Overview 502
26.1.2 Features 504
26.2 Memory map and register definition 504
26.2.1 Register descriptions 505
26.3 Functional description 532
26.3.1 System clock generation 532
26.3.2 Auxiliary clock generation 538

27 OSC Digital Interface (XOSC) 543
27.1 Introduction 543
27.2 Functional description 543
27.2.1 Oscillator power-down control and status 543
27.2.2 Oscillator startup delay 544
27.2.3 Oscillator clock available interrupt 544
27.2.4 Oscillator bypass mode 544
27.3 Memory map and register definition 544
27.3.1 Register descriptions 544

28 IRCOSC digital interface 547
28.1 Introduction 547
28.2 Functional description 547
28.3 Memory map and register definition 547
28.3.1 Register descriptions 548

29 Platform RAM controller (PRAM) 550
29.1 Introduction 550
29.2 SRAM controller memory map and register definitions 551

29.2.1 Platform RAM configuration register 1 (PRCR1) . . . 551
29.3 Functional description . . . 553
29.3.1 Read/Write introduction . . . 553
29.3.2 Writes . . . 553
29.4 Initialization/application information . . . 554
29.5 Safety considerations . . . 554
29.5.1 Hsiao ECC algorithm . . . 554
29.5.2 Transaction monitor . . . 557

30 Flash memory controller (PFLASH Controller) . . . 559
30.1 Introduction . . . 559
30.2 Features . . . 559
30.3 Block diagram . . . 559
30.4 Memory map and register definitions . . . 560
30.4.1 Overview . . . 560
30.5 Functional description . . . 572
30.5.1 Basic interface protocol . . . 572
30.5.2 Access protections . . . 573
30.5.3 Read cycles - buffer miss . . . 573
30.5.4 Read cycles – buffer hit . . . 573
30.5.5 Error termination . . . 573
30.5.6 Flash error response operation . . . 574
30.5.7 Censorship . . . 574
30.5.8 Line read buffers and prefetch operation . . . 574
30.5.9 Instruction/Data prefetch triggering . . . 575
30.5.10 Per-Master prefetch triggering . . . 575
30.5.11 Buffer allocation . . . 575
30.5.12 Safety considerations . . . 576
30.5.13 Array integrity considerations . . . 577

31 Embedded Flash Memory . . . 579
31.1 Introduction . . . 579
31.1.1 Overview . . . 579
31.1.2 Features . . . 581
31.1.3 Modes of operation . . . 581
31.2 Flash memory map and description . . . 582

31.2.1 Flash array memory map . . . 583
31.3 Register memory maps and descriptions . . . 585
31.3.1 Register memory maps . . . 586
31.3.2 User register descriptions . . . 587
31.4 Functional description . . . 616
31.4.1 Reset . . . 616
31.4.2 Power-down mode (Disable mode) . . . 617
31.4.3 Low-power mode (Sleep mode) . . . 617
31.4.4 Read mode . . . 618
31.4.5 Modify mode . . . 620
31.4.6 Protection strategy . . . 631

32 Error Injection Module (EIM) . . . 633
32.1 Introduction . . . 633
32.1.1 Overview . . . 633
32.1.2 Features . . . 633
32.2 Memory map and register definition . . . 633
32.2.1 Memory map . . . 633
32.2.2 Register descriptions . . . 633
32.3 Functional description . . . 635
32.4 Application Information . . . 637

33 Analog-to-digital converter (ADC) configuration . . . 639
33.1 Overview . . . 639
33.1.1 ADC subsystem block diagram . . . 639
33.1.2 Analog input pin multiplexing . . . 640
33.2 Configuration . . . 642
33.2.1 External signal description . . . 642
33.2.2 Integration diagram . . . 642
33.2.3 Clock sources . . . 642
33.2.4 Triggering . . . 643
33.2.5 SAR absolute voltage reference . . . 644
33.2.6 Diagnostics . . . 644
33.2.7 Channel assignment . . . 646
33.2.8 Register definitions . . . 647

34 Successive Approximation Register Analog-to-Digital Converter (SARADC) Digital Interface 659

34.1 Introduction 659
34.2 Overview 659
34.3 Feature description 661
34.3.1 Main features 661
34.4 Functional description 661
34.4.1 Normal channel conversion 661
34.4.2 Injected channel conversion 663
34.4.3 Abort conversion 664
34.4.4 Analog conversion timings and reference selection 665
34.4.5 CrossTriggering Unit (CTU) 666
34.4.6 Test channel connection with internal analog channel 669
34.4.7 Programmable analog watchdog 669
34.4.8 DMA functionality 671
34.4.9 Interrupts 671
34.4.10 Power down mode 672
34.4.11 Stop Mode 672
34.5 Memory map and registers 672
34.5.1 Register descriptions 675
34.6 Start of conversion pulse delay 706
34.7 Initialization Information 708

35 Cross Triggering Unit (CTU) 710

35.1 Introduction 710
35.2 CTU overview 710
35.3 Functional description 711
35.3.1 Trigger events features 711
35.3.2 Trigger generator subunit (TGS) 712
35.3.3 TGS in triggered mode 712
35.3.4 TGS in sequential mode 713
35.3.5 TGS counter 714
35.4 Scheduler subunit (SU) 715
35.4.1 ADC commands list 717
35.4.2 ADC commands list format 717
35.4.3 ADC results 718

35.5 Reload mechanism 718
35.6 Power safety mode 720
35.7 Interrupts and DMA requests 720
35.7.1 DMA support 720
35.7.2 CTU faults and errors 721
35.7.3 CTU interrupt/DMA requests 722
35.8 Memory map 723
35.8.1 Trigger Generator Sub-unit Input Selection Register (TGSISR) 727
35.8.2 Trigger Generator Sub-unit Control Register (TGSCR) 729
35.8.3 Trigger n Compare Register (TnCR) 730
35.8.4 TGS Counter Compare Register (TGSCCR) 730
35.8.5 TGS Counter Reload Register (TGSCRR) 731
35.8.6 Commands list control register 1 (CLCR1) 731
35.8.7 Commands list control register 2 (CLCR2) 732
35.8.8 Trigger handler control register 1 (THCR1) 732
35.8.9 Trigger handler control register 2 (THCR2) 734
35.8.10 Commands list registers (CLR*n*) *736*
35.8.11 FIFO DMA control register (FDCR) 737
35.8.12 FIFO control register (FCR) 738
35.8.13 FIFO threshold register (FTH) 740
35.8.14 FIFO status register (FST) 740
35.8.15 FIFO Right aligned data *n* (FR*n*) 742
35.8.16 FIFO signed Left aligned data *n* (FL*n*) 743
35.8.17 Cross triggering unit error flag register (CTUEFR) 743
35.8.18 Cross triggering unit interrupt flag register (CTUIFR) 744
35.8.19 Cross triggering unit interrupt/DMA register (CTUIR) 745
35.8.20 Control ON time register (COTR) 746
35.8.21 Cross triggering unit control register (CTUCR) 747
35.8.22 Cross triggering unit digital filter (CTUDF) 748

36 Enhanced Motor Control Timer (eTimer) 749

36.1 Introduction 749
36.1.1 Overview 749
36.1.2 Features 750
36.1.3 Module block diagram 750
36.1.4 Channel block diagram 751

36.2 External signal descriptions . . . 752
36.2.1 TIO[*n*:0] - Timer Input/Outputs . . . 752
36.3 Functional Description . . . 753
36.3.1 General . . . 753
36.3.2 Counting Modes . . . 753
36.3.3 Other Features . . . 761
36.4 Memory Map and Registers . . . 762
36.4.1 Overview . . . 762
36.4.2 Module Memory Map . . . 762
36.4.3 Register Descriptions . . . 763
36.4.4 Timer Channel Registers . . . 763
36.4.5 Watchdog Timer Registers . . . 779
36.4.6 Fault Registers . . . 780
36.4.7 Configuration Registers . . . 782
36.5 Resets . . . 786
36.6 Clocks . . . 787
36.7 Interrupts . . . 787
36.8 DMA . . . 787

37 System Timer Module (STM) . . . 789
37.1 Introduction . . . 789
37.1.1 Overview . . . 789
37.1.2 Features . . . 789
37.1.3 Modes of operation . . . 789
37.2 External signal description . . . 789
37.3 Memory map and register definition . . . 789
37.3.1 Memory map . . . 789
37.3.2 Register descriptions . . . 790
37.4 Functional description . . . 793

38 Software Watchdog Timer (SWT) . . . 794
38.1 Introduction . . . 794
38.1.1 Overview . . . 794
38.1.2 Features . . . 794
38.1.3 Modes of operation . . . 794
38.2 External signal description . . . 794

38.3 Memory map and register definition 794
38.3.1 Register descriptions 795
38.4 Functional description 800
38.4.1 Introduction 800
38.4.2 Configuration locking 801
38.4.3 Unlock sequence 801
38.4.4 Servicing operations 801
38.4.5 Time-out 802
38.4.6 Initialization 802

39 Periodic Interrupt Timer (PIT) 803
39.1 Introduction 803
39.1.1 Block diagram 803
39.1.2 Features 804
39.2 Signal description 804
39.2.1 Memory map/register description 804
39.3 Functional description 807
39.3.1 General 807
39.3.2 Interrupts 809
39.3.3 Chained timers 809
39.4 Initialization and application information 809

40 FlexCAN 811
40.1 Introduction 811
40.1.1 Overview 811
40.1.2 FlexCAN module features 813
40.1.3 Modes of operation 813
40.2 External signal description 814
40.2.1 Overview 814
40.2.2 Signal Descriptions 815
40.3 Memory map and registers description 815
40.3.1 FlexCAN memory mapping 815
40.3.2 Message buffer structure 817
40.3.3 Rx FIFO structure 821
40.3.4 Registers description 822
40.4 Functional description 840

40.4.1 Overview . . . 840
40.4.2 Transmit process . . . 841
40.4.3 Arbitration process . . . 841
40.4.4 Receive process . . . 842
40.4.5 Matching process . . . 843
40.4.6 Data coherence . . . 844
40.4.7 Rx FIFO . . . 847
40.4.8 CAN protocol related features . . . 848
40.4.9 Modes of operation details . . . 853
40.4.10 Interrupts . . . 854
40.4.11 Bus interface . . . 855
40.5 Initialization/application information . . . 855
40.5.1 FlexCAN initialization sequence . . . 855

41 Deserial Serial Peripheral Interface (DSPI) . . . 857
41.1 Introduction . . . 857
41.1.1 Block diagram . . . 857
41.1.2 Features . . . 858
41.1.3 SPI configuration . . . 859
41.1.4 Modes of operation . . . 860
41.2 DSPI signal description . . . 861
41.2.1 PCS0/SS — Peripheral Chip Select 0 / Slave Select . . . 861
41.2.2 PCS[1:3] — Peripheral Chip Selects 1 – 3 . . . 862
41.2.3 PCS4/MTRIG — Peripheral Chip Select 4 / Master Trigger . . . 862
41.2.4 PCS5/PCSS — Peripheral Chip Select 5 / Peripheral Chip Select Strobe . . . 862
41.2.5 PCS[6:7] — Peripheral Chip Selects 6 – 7 . . . 862
41.2.6 SIN — Serial Input . . . 862
41.2.7 SOUT — Serial Output . . . 862
41.2.8 SCK — Serial Clock . . . 862
41.2.9 Memory map/register definition . . . 862
41.3 Register classification for safety . . . 888
41.4 Functional description . . . 888
41.4.1 Start and Stop of DSPI transfers . . . 889
41.4.2 Serial Peripheral Interface (SPI) configuration . . . 890
41.4.3 DSPI baud rate and clock delay generation . . . 893
41.4.4 Transfer formats . . . 896

41.4.5 Continuous Serial Communications Clock 905
41.4.6 Slave mode operation constraints 906
41.4.7 Parity generation and check 907
41.4.8 Interrupts/DMA requests 907
41.4.9 Power saving features 910
41.5 Initialization/application information 911
41.5.1 Managing DSPI queues 911
41.5.2 Switching master and slave mode 911
41.5.3 Initializing DSPI in Master/Slave Modes 912
41.5.4 Baud rate settings 912
41.5.5 Delay settings 913
41.5.6 Calculation of FIFO pointer addresses 914

42 LINFlexD 917
42.1 Introduction 917
42.2 Main features 919
42.2.1 LIN mode features 919
42.2.2 UART mode features 920
42.2.3 Standard features 920
42.3 Functional description 920
42.3.1 LIN protocol 920
42.3.2 LINFlexD features 922
42.3.3 Timer 938
42.3.4 UART mode 939
42.3.5 DMA interface 944
42.4 Memory map and register description 962
42.4.1 Memory map 962
42.4.2 Register description 964
42.5 Programming considerations 997
42.5.1 Master node 997
42.5.2 Slave node 999
42.5.3 Extended frames 1003
42.5.4 Timeout 1003
42.5.5 UART mode 1004
42.5.6 Interrupts 1004

43 Reset Generation Module (MC_RGM) 1006

43.1 Introduction . . . 1006
43.1.1 Overview . . . 1006
43.1.2 Features . . . 1007
43.1.3 Reset sources . . . 1008
43.2 External signal description . . . 1009
43.3 Memory map and register definition . . . 1009
43.3.1 Memory map . . . 1009
43.3.2 Register descriptions . . . 1011
43.4 Functional description . . . 1042
43.4.1 Reset state machine . . . 1042
43.4.2 'Destructive' resets . . . 1046
43.4.3 'Functional' resets . . . 1046
43.4.4 Alternate event generation . . . 1047
43.4.5 'Functional' reset escalation . . . 1047
43.4.6 'Destructive' reset escalation . . . 1048
43.4.7 Individual peripheral resets . . . 1048

44 Boot Assist Flash (BAF) . . . 1049
44.1 Introduction . . . 1049
44.2 Memory map . . . 1049
44.2.1 BAF image header . . . 1049
44.3 Functional description . . . 1050
44.3.1 Boot modes . . . 1050
44.3.2 Device initialization . . . 1050
44.3.3 Flow of control . . . 1051
44.3.4 Initialization of BAF DCF clients . . . 1052
44.3.5 Optionally perform a serial boot . . . 1057
44.3.6 Serial boot configuration . . . 1062
44.4 Resources accessed by BAF code execution . . . 1065

45 System Status and Configuration Module (SSCM) . . . 1066
45.1 Introduction . . . 1066
45.1.1 Overview . . . 1066
45.1.2 Features . . . 1066
45.1.3 Modes of operation . . . 1066
45.2 Memory map and register definition . . . 1066

45.2.1 Register descriptions 1067
45.3 Functional description 1071
45.3.1 DCF mechanism 1071
45.3.2 ECC error monitoring 1073
45.3.3 Boot mode functionality 1074
45.3.4 User code boot sector search 1074
45.3.5 Life Cycle 1076
45.4 Initialization and application information 1077
45.4.1 Reset 1077
45.5 Additional safety measures 1077
45.5.1 Spurious reset protection 1077

46 Power management controller digital interface (PMC_Dig) 1078
46.1 Introduction 1078
46.2 Main features 1079
46.2.1 Standard features 1080
46.3 IPS bus interface 1080
46.4 Memory map/register definition 1080
46.5 Registers description 1083
46.5.1 Event Pending Register (EPR_LV0) 1083
46.5.2 Reset Event Enable Register (REE_LV0) 1083
46.5.3 Reset Event Select Register (RES_LV0) 1084
46.5.4 Interrupt Enable (IE_LV0) 1085
46.5.5 FCCU Event Enable Register (FEE_LV0) 1085
46.5.6 Event Pending Register (EPR_LV1) 1086
46.5.7 Reset Event Enable Register (REE_LV1) 1087
46.5.8 Reset Event Select Register (RES_LV1) 1087
46.5.9 Interrupt Enable (IE_LV1) 1088
46.5.10 FCCU Event Enable Register (FEE_LV1) 1089
46.5.11 Event Pending Register (EPR_HV0) 1089
46.5.12 Reset Event Enable Register (REE_HV0) 1090
46.5.13 Reset Event Select Register (RES_HV0) 1091
46.5.14 Interrupt Enable (IE_HV0) 1092
46.5.15 FCCU Event Enable Register (FEE_HV0) 1093
46.5.16 Event Pending Register (EPR_HV1) 1094
46.5.17 Reset Event Enable Register (REE_HV1) 1095

46.5.18 Reset Event Select Register (RES_HV1) 1096
46.5.19 Interrupt Enable Register (IE_HV1) 1097
46.5.20 FCCU Event Enable Register (FEE_HV1) 1098
46.5.21 Supply Gauge Status Register (*GR_S*) 1099
46.5.22 Pending Gauge Status Register (GR_P) 1103
46.5.23 Interrupt Enable Pending Register (IE_G) 1105
46.5.24 Main Regulator Control Register (MREG_CTRL) 1106
46.5.25 Event Pending Register (EPR_TD) 1106
46.5.26 Reset Event Enable Register (REE_TD) 1108
46.5.27 Reset Event Select Register (RES_TD) 1109
46.5.28 Temperature Sensor Configuration Register (CTL_TD) 1110
46.5.29 Temperature Sensor FCCU Event Enable Register (FEE_TD) 1112
46.5.30 ESR0 Configuration Register (ESR0_CFG) 1112
46.5.31 User BIST Flags Register (BIST_FLAGS) 1113
46.5.32 User BIST Control Register (BIST_CTRL) 1114
46.5.33 User BIST Time1 and Time0 Register (BIST_TIME10) 1116
46.5.34 User BIST Time3 and Time2 Register (BIST_TIME32) 1116
46.5.35 User BIST Time6 and Time5 Register (BIST_TIME65) 1117
46.5.36 User BIST VD Under Test Monitor Register (BIST_DEBUG) 1117
46.6 Temperature Sensor Interface Logic 1118
46.7 Power On Reset (MC_RGM Phase Gates) 1118
46.8 Flash interface (DCF client usage) 1119

47 Power Control Unit (MC_PCU) 1121
47.1 Introduction 1121
47.1.1 Overview 1121
47.1.2 Features 1122
47.2 External signal description 1122
47.3 Memory map and register definition 1122
47.3.1 Memory map 1122
47.3.2 Register descriptions 1122

48 Mode Entry Module (MC_ME) 1124
48.1 Introduction 1124
48.1.1 Overview 1124
48.1.2 Features 1126

48.1.3 Modes of operation 1126
48.2 External signal description 1127
48.3 Memory map and register definition 1127
48.3.1 Memory map 1128
48.3.2 Register description 1130
48.4 Functional description 1167
48.4.1 Mode Transition Request 1167
48.4.2 Modes details 1168
48.4.3 Mode Transition Process 1172
48.4.4 Protection of mode configuration registers 1180
48.4.5 Mode Transition Interrupts 1180
48.4.6 Peripheral Clock Gating 1182
48.4.7 Application example 1182

49 e200z0h Core Debug Support 1184
49.1 Overview 1184
49.2 Software debug facilities 1184
49.2.1 Power Architecture Book E compatibility 1185
49.2.2 Additional debug facilities 1185
49.2.3 Hardware debug facilities 1185
49.3 Software debug events and exceptions 1186
49.3.1 Instruction address compare event 1187
49.3.2 Data address compare event 1188
49.3.3 Linked instruction address and data address compare event 1188
49.3.4 Trap debug event 1189
49.3.5 Branch taken debug event 1189
49.3.6 Instruction complete debug event 1189
49.3.7 Interrupt taken debug event 1189
49.3.8 Critical interrupt taken debug event 1190
49.3.9 Return debug event 1190
49.3.10 Critical return debug event 1190
49.3.11 External debug event 1190
49.3.12 Unconditional debug event 1190
49.4 Debug registers 1191
49.4.1 Debug address and value registers 1191
49.4.2 Debug control and status registers 1191

49.5 External debug support 1199
49.5.1 OnCE introduction 1200
49.5.2 JTAG/OnCE pins 1203
49.5.3 OnCE internal interface signals 1203
49.5.4 OnCE interface signals 1204
49.5.5 e200z0h OnCE controller and serial interface 1205
49.5.6 Access to debug resources 1212
49.5.7 Methods of entering Debug mode 1214
49.5.8 CPU Status and Control Scan Chain Register (CPUSCR) 1215
49.5.9 Reserved registers (Reserved) 1220
49.6 Watchpoint support 1220
49.7 Basic steps for enabling, using, and exiting External Debug mode . . . 1221

50 Debug and Trace 1224
50.1 Introduction 1224
50.1.1 Features 1224
50.1.2 Architecture 1224
50.1.3 Overview 1225
50.2 Register description 1226
50.2.1 DCI control register (DCI_CR) 1226
50.2.2 DCI pin multiplexing control register (DCI_PINCR) 1227
50.2.3 Nexus Crossbar Multimaster Trace Clients (NXMC) 1229
50.2.4 Sequence Processing Unit (SPU) 1230
50.3 External signal description 1242
50.3.1 Interfaces 1242
50.3.2 Nexus trace enabling during reset 1243
50.4 Functionality 1244
50.4.1 Debug Control Interface (DCI) 1244
50.5 Operating modes 1250
50.5.1 Nexus reset control 1250

51 JTAG Controller (JTAGC) 1252
51.1 Introduction 1252
51.1.1 Block Diagram 1252
51.1.2 Features 1253
51.1.3 Modes of operation 1253

51.2 External signal description . . . 1254
51.2.1 TCK—Test clock input . . . 1254
51.2.2 TDI—Test data input . . . 1254
51.2.3 TDO—Test data output . . . 1254
51.2.4 TMS—Test mode select . . . 1254
51.3 Register description . . . 1254
51.3.1 Instruction register . . . 1255
51.3.2 Bypass register . . . 1255
51.3.3 Device identification register . . . 1255
51.3.4 JTAG_PASSWORD register . . . 1256
51.3.5 Boundary scan register . . . 1257
51.4 Functional description . . . 1257
51.4.1 JTAGC reset configuration . . . 1257
51.4.2 IEEE 1149.1-2001 (JTAG) Test Access Port . . . 1257
51.4.3 TAP controller state machine . . . 1258
51.4.4 JTAGC block instructions . . . 1260
51.4.5 Boundary scan . . . 1263
51.5 Initialization/Application information . . . 1263

52 JTAG Data Communication (JDC) . . . 1265
52.1 Introduction . . . 1265
52.2 Overview . . . 1265
52.3 Register definition . . . 1266
52.3.1 Register descriptions . . . 1266
52.3.2 Non-Memory mapped register definition . . . 1268
52.4 Functional description . . . 1269
52.4.1 IPS register access . . . 1270
52.4.2 JTAG register access . . . 1270

53 JTAG Master (JTAGM) . . . 1271
53.1 Introduction . . . 1271
53.1.1 Overview . . . 1271
53.1.2 Feature description . . . 1272
53.2 Functional description . . . 1272
53.2.1 Module architecture . . . 1272
53.2.2 JTAGM Ready signal . . . 1273

53.2.3 EVTO and EVTI signals . . . 1273
53.2.4 JTAGM configuration and status monitoring . . . 1274
53.2.5 JTAGM to DCI serial interface . . . 1274
53.2.6 Software interface . . . 1274
53.3 Modes of operation . . . 1274
53.4 Memory map and register definition . . . 1274
53.4.1 Register descriptions . . . 1275

54 Nexus Port Controller (NPC) . . . 1282
54.1 Information specific to this device . . . 1282
54.1.1 Parameter values . . . 1282
54.1.2 Unavailable features . . . 1282
54.2 Introduction . . . 1282
54.2.1 Overview . . . 1283
54.2.2 Features . . . 1283
54.2.3 Modes of operation . . . 1284
54.3 External signal description . . . 1285
54.3.1 Overview . . . 1285
54.3.2 Detailed signal descriptions . . . 1285
54.4 Register definition . . . 1286
54.4.1 Register descriptions . . . 1287
54.5 Functional description . . . 1290
54.5.1 NPC reset configuration . . . 1290
54.5.2 Auxiliary output port . . . 1290
54.5.3 IEEE 1149.1-2001 (JTAG) TAP . . . 1293
54.5.4 Nexus JTAG port sharing . . . 1297
54.5.5 MCKO and ipg_sync_mcko . . . 1297
54.5.6 EVTO sharing . . . 1297
54.5.7 Nexus reset control . . . 1297
54.6 Initialization/Application information . . . 1298
54.6.1 Accessing NPC tool-mapped registers . . . 1298

55 Sequence Processing Unit (SPU) . . . 1299
55.1 Introduction . . . 1299
55.1.1 SPU block diagram . . . 1299
55.1.2 Overview . . . 1300

55.1.3 Features . . . 1302
55.2 SPU actions . . . 1303
55.3 Operation mode . . . 1304
55.3.1 SPU enable/disable . . . 1304
55.4 SPU module interface . . . 1304
55.5 Register definition . . . 1304
55.5.1 Registers description . . . 1311
55.6 Functional description . . . 1349
55.6.1 Input Mux unit . . . 1349
55.6.2 Processor exception vector encoding . . . 1351
55.6.3 State logic unit (SLU) . . . 1352
55.6.4 Sequence formation . . . 1353
55.6.5 Performance counters/timers unit . . . 1356
55.6.6 Action unit . . . 1357
55.6.7 Optional trace group . . . 1357

56 e200z0h Nexus 2+ Module . . . 1358
56.1 Introduction . . . 1358
56.1.1 General Description . . . 1358
56.1.2 Terms and Definitions . . . 1358
56.1.3 Feature List . . . 1359
56.1.4 Functional Block Diagram . . . 1361
56.2 Enabling Nexus 2+ Operation . . . 1361
56.3 TCODEs supported . . . 1362
56.4 Nexus 2+ Programmer’s Model . . . 1366
56.4.1 Client Select Control Register (CSC) . . . 1367
56.4.2 Port Configuration Register - reference only (PCR) . . . 1367
56.4.3 Nexus Development Control Register 1 (DC1) . . . 1368
56.4.4 Nexus Development Control Registers 2-3 (DC2-3) . . . 1370
56.4.5 Nexus Development Control Register 4 (DC4) . . . 1372
56.4.6 Development Status Register (DS) . . . 1373
56.4.7 Watchpoint Trigger Registers (WT, PTSTC, PTETC) . . . 1374
56.4.8 Nexus Watchpoint Mask Register (WMSK) . . . 1377
56.4.9 Nexus Overrun Control Register (OVCR) . . . 1379
56.4.10 Read/Write Access Control/Status (RWCS) . . . 1379
56.4.11 Read/Write Access Data (RWD) . . . 1381

56.4.12 Read/Write Access Address (RWA) . . . 1382
56.5 Nexus 2+ Register Access via JTAG/OnCE . . . 1383
56.6 Nexus Message Fields . . . 1383
56.6.1 TCODE Field . . . 1384
56.6.2 Source ID Field (SRC) . . . 1384
56.6.3 Relative Address Field (U-ADDR) . . . 1384
56.6.4 Full Address Field (F-ADDR) . . . 1385
56.6.5 Address Space Indication Field (MAP) . . . 1385
56.7 Nexus Message Queues . . . 1385
56.7.1 Message Queue Overrun . . . 1385
56.7.2 CPU Stall . . . 1386
56.7.3 Message Suppression . . . 1386
56.7.4 Nexus Message Priority . . . 1386
56.7.5 Data Acquisition Message Priority Loss Response . . . 1387
56.7.6 Ownership Trace Message Priority Loss Response . . . 1387
56.7.7 Program Trace Message Priority Loss Response . . . 1387
56.8 Debug Status Messages . . . 1387
56.9 Error Messages . . . 1388
56.10 Ownership Trace . . . 1388
56.10.1 Overview . . . 1388
56.10.2 Ownership Trace Messaging (OTM) . . . 1388
56.11 Program Trace . . . 1389
56.11.1 Branch Trace Messaging Types . . . 1389
56.11.2 BTM Message Formats . . . 1391
56.11.3 Program Trace Message Fields . . . 1392
56.11.4 Resource Full Messages . . . 1393
56.11.5 Program Correlation Messages . . . 1394
56.11.6 Program Trace Overflow Error Messages . . . 1395
56.11.7 Program Trace Synchronization Messages . . . 1395
56.11.8 Enabling Program Trace . . . 1397
56.11.9 Program Trace Timing Diagrams (2 MDO / 1 MSEO configuration) . 1397
56.12 Data Acquisition Messaging . . . 1398
56.12.1 Data Acquisition ID Tag Field . . . 1399
56.12.2 Data Acquisition Data Field . . . 1399
56.12.3 Data Acquisition Trace event . . . 1399
56.13 Watchpoint Trace Messaging . . . 1399

56.13.1 Watchpoint Timing Diagram (2 MDO / 1 MSEO configuration) 1401
56.14 Nexus 2+ Read/Write Access to Memory-Mapped Resources 1401
56.14.1 Single Write Access 1402
56.14.2 Block Write Access 1402
56.14.3 Single Read Access 1403
56.14.4 Block Read Access 1403
56.14.5 Error Handling 1404
56.14.6 Read/Write Access Error Message 1404
56.15 Nexus 2+ Pin Interface 1405
56.15.1 Pins Implemented 1405
56.15.2 Pin Protocol 1407
56.16 Rules for Output Messages 1409
56.17 Auxiliary Port Arbitration 1410
56.18 Examples 1410
56.19 Electrical Characteristics 1412
56.20 IEEE 1149.1 (JTAG) RD/WR Sequences 1412
56.20.1 JTAG Sequence for Accessing Internal Nexus Registers 1412
56.20.2 JTAG Sequence for Read Access of Memory-Mapped Resources . . 1413
56.20.3 JTAG Sequence for Write Access of Memory-Mapped Resources . . 1413

57 Nexus Crossbar Multi-Master Client (NXMC) 1415
57.1 Introduction 1415
57.1.1 Features 1415
57.2 External signal description 1415
57.3 Supported messages and TCODEs 1416
57.4 Register description 1419
57.4.1 Register definitions 1420
57.5 Programming considerations (RESET) 1427
57.5.1 IEEE 1149.1 (JTAG) Test Access Port 1427
57.5.2 Enabling NXMC operation 1427
57.5.3 Enabling NXMC TAP controller 1427
57.5.4 NXMC register access via JTAG 1428
57.6 Functional description 1428
57.6.1 Data trace 1428
57.6.2 Watchpoint support 1433

58 Cyclic Redundancy Check (CRC) Unit 1435
58.1 Introduction 1435
58.2 Main features 1435
58.2.1 Standard features 1435
58.3 Block diagram 1435
58.4 Signal description 1436
58.4.1 Peripheral bus interface 1436
58.5 Memory map and registers 1436
58.5.1 Register description 1436
58.6 Functional description 1440
58.7 Use cases 1442
58.7.1 Programming example 1442
58.7.2 Register programming 1444

59 Memory Error Management Unit (MEMU) 1445
59.1 Introduction 1445
59.2 Features 1446
59.3 Block diagram 1446
59.4 Design overview 1447
59.5 External signal description 1450
59.6 Memory map and register definition 1450
59.6.1 Memory map 1450
59.6.2 Register descriptions 1452
59.7 Functional description 1466
59.7.1 Initializing MEMU 1466
59.7.2 Reading the reporting table 1467
59.7.3 Handling overflows (Multiple error reporting) 1468

60 Fault Collection and Control Unit (FCCU) 1470
60.1 Introduction 1470
60.2 Features 1470
60.2.1 Standard features 1471
60.3 Block diagram 1471
60.4 Signal description 1472
60.4.1 Pinout 1472

60.4.2 IPS bus interface 1473
60.5 Functional description 1473
60.5.1 Definitions 1473
60.5.2 FSM description 1474
60.5.3 Reset interface 1475
60.5.4 Fault priority scheme and nesting 1476
60.5.5 Fault recovery 1477
60.5.6 NMI/WKPU interface 1479
60.5.7 FAULT interface 1480
60.5.8 STCU interface 1481
60.5.9 EOUT interface 1482
60.5.10 Error signal flow for RF case 1488
60.6 Register description 1490
60.6.1 FCCU Control Register (FCCU_CTRL) 1493
60.6.2 FCCU CTRL Key Register (FCCU_CTRLK) 1495
60.6.3 FCCU Configuration Register (FCCU_CFG) 1495
60.6.4 FCCU RF Configuration Register n (FCCU_RF_CFGn) 1499
60.6.5 FCCU RFS Configuration Register n (FCCU_RFS_CFGn) 1500
60.6.6 FCCU RF Status register n (FCCU_RF_Sn) 1501
60.6.7 FCCU RF Key register (FCCU_RFK) 1503
60.6.8 FCCU RF Enable Register n (FCCU_RF_En) 1503
60.6.9 FCCU RF Timeout Enable Register n (FCCU_RF_TOEn) 1504
60.6.10 FCCU RF Timeout Register (FCCU_RF_TO) 1505
60.6.11 FCCU CFG Timeout Register (FCCU_CFG_TO) 1506
60.6.12 FCCU IO Control Register (FCCU_EINOUT) 1507
60.6.13 FCCU Status Register (FCCU_STAT) 1509
60.6.14 FCCU NA Freeze Status Register (N2AF_STATUS) 1510
60.6.15 FCCU AF Freeze Status Register (A2FF_STATUS) 1511
60.6.16 FCCU NF Freeze Status Register (N2FF_STATUS) 1512
60.6.17 FCCU FA Freeze Status Register (F2A_STATUS) 1513
60.6.18 FCCU RF Fake Register (FCCU_RFF) 1514
60.6.19 FCCU IRQ Status Register (FCCU_IRQ_STAT) 1515
60.6.20 FCCU IRQ Enable Register (FCCU_IRQ_EN) 1516
60.6.21 FCCU XTMR Register (FCCU_XTMR) 1517
60.6.22 FCCU MCS Register (FCCU_MCS) 1518
60.6.23 FCCU Transient Register (FCCU_TRANS_LOCK) 1520
60.6.24 FCCU Permanent Lock Register (FCCU_PERMNT_LOCK) 1521

60.6.25 FCCU Delta T Register (FCCU_DELTA_T) 1521
60.6.26 FCCU IRQ Alarm Enable register n (FCCU_IRQ_ALARM_ENn) . . . 1522
60.6.27 FCCU NMI Enable registers n (FCCU_NMI_ENn) 1523
60.6.28 FCCU EOUT Signaling Enable registers n (FCCU_EOUT_SIG_ENn) 1524
60.7 FCCU Output Supervision Unit 1525
60.8 Use cases and limitations 1527
60.8.1 Misconfigurations 1527
60.8.2 Recommendations to configure FCCU 1528

61 Self-Test Control Unit (STCU2) 1529
61.1 Introduction 1529
61.2 Main features 1530
61.3 Block diagram 1531
61.3.1 IPS bus interface 1532
61.4 Functional description 1533
61.4.1 FSM description 1533
61.4.2 Reset management 1533
61.4.3 LBIST and MBIST scheduling 1534
61.4.4 ABORT management 1534
61.4.5 PLL interface 1534
61.4.6 FCCU interface 1535
61.4.7 Watchdogs 1535
61.4.8 CRC 1536
61.4.9 SSCM interface 1536
61.5 Register description 1536
61.5.1 STCU2 Run Register (STCU_RUN) 1538
61.5.2 STCU2 SK Code Register (STCU_SKC) 1538
61.5.3 STCU2 Configuration Register (STCU_CFG) 1540
61.5.4 STCU2 PLL Configuration Register (STCU_PLL_CFG) 1542
61.5.5 STCU2 Watchdog Register Granularity (STCU_WDG) 1543
61.5.6 STCU2 CRC Expected Status Register (STCU_CRCE) 1544
61.5.7 STCU2 CRC Read Status Register (STCU_CRCR) 1545
61.5.8 STCU2 Error Register (STCU_ERR_STAT) 1545
61.5.9 STCU2 Error FM Register (STCU_ERR_FM) 1547
61.5.10 STCU2 Off-line LBIST Status Register (STCU_LBS) 1548
61.5.11 STCU2 Off-line LBIST End Flag Register (STCU_LBE) 1549

61.5.12 STCU2 LBIST Unrecoverable FM Register (STCU_LBUFM) 1550
61.5.13 STCU2 Off-line MBIST Status Low Register (STCU_MBSL) 1550
61.5.14 STCU2 Off-line MBIST Status Medium Register (STCU_MBSM) . . . 1551
61.5.15 STCU2 Off-line MBIST Status High Register (STCU_MBSH) 1552
61.5.16 STCU2 Off-line MBIST End Flag Low Register (STCU_MBEL) 1552
61.5.17 STCU2 Off-line MBIST End Flag Medium Register (STCU_MBEM) . 1553
61.5.18 STCU2 Off-line MBIST End Flag High Register (STCU_MBEH) 1554
61.5.19 STCU2 MBIST Unrecoverable FM Low Register (STCU_MBUFML) 1554
61.5.20 STCU2 MBIST Unrecoverable FM Medium Register (STCU_MBUFMM) 1555
61.5.21 STCU2 MBIST Unrecoverable FM High Register (STCU_MBUFMH) 1556
61.5.22 STCU2 LBIST Control Register n (STCU_LB_CTRLn) 1557
61.5.23 STCU2 LBIST PC Stop Register n (STCU_LB_PCSn) 1559
61.5.24 STCU2 LBIST PRPG Low Register n (STCU_LB_PRPGLn) 1560
61.5.25 STCU2 LBIST PRPG High Register n (STCU_LB_PRPGHn) 1561
61.5.26 STCU2 Off-line LBIST MISR Expected Low Register n (STCU_LB_MISRELn) 1561
61.5.27 STCU2 Off-line LBIST MISR Expected High Register n (STCU_LB_MISREHn) 1562
61.5.28 STCU2 Off-line LBIST MISR Read Low Register n (STCU_LB_MISRRLn) 1563
61.5.29 STCU2 Off-line LBIST MISR Read High Register n (STCU_LB_MISRRHn) 1563
61.5.30 STCU2 MBIST Control Register n (STCU_MB_CTRLn) 1564
61.6 Register map 1565
61.7 Use cases and limitations 1568
61.7.1 Off-line Self-Test sequence 1568
61.7.2 On-line Self-Test Sequence 1569
61.7.3 Bypass USER Mode 1571
61.7.4 Design implementation information 1571

62 Register Protection (REG_PROT) 1572
62.1 Introduction 1572
62.1.1 Overview 1572
62.1.2 Features 1572
62.1.3 Modes of operation 1573
62.2 External signal description 1573
62.3 Memory map and register definition 1573

62.3.1 Memory map . . . 1574
62.3.2 Register descriptions . . . 1575
62.4 Functional description . . . 1577
62.4.1 General . . . 1578
62.4.2 Change lock settings . . . 1578
62.4.3 Access errors . . . 1581
62.5 Initialization/application information . . . 1582
62.5.1 Reset . . . 1582

63 Password and Device Security Module (PASS) . . . 1583
63.1 Introduction . . . 1583
63.1.1 Overview . . . 1583
63.1.2 Features . . . 1583
63.1.3 Modes of operation . . . 1583
63.2 Memory map and register definition . . . 1584
63.2.1 Register descriptions . . . 1585
63.3 DCF clients . . . 1594
63.3.1 Censorship . . . 1595
63.4 Functional description . . . 1595
63.4.1 Censorship . . . 1595
63.4.2 Debug Interface Access . . . 1595
63.4.3 Flash Memory Read Protection . . . 1596
63.4.4 UTEST protection . . . 1597
63.4.5 Retest Password . . . 1597
63.5 Initialization/application information . . . 1598
63.5.1 Reset . . . 1598
63.5.2 Setting lock bits in a password group . . . 1598
63.5.3 Censoring and uncensoring the device . . . 1599

Appendix A Acronyms and abbreviations . . . 1600

Revision history . . . 1604

List of tables

Table 1. Register conventions . . . 75
Table 2. SPC570Sx device feature summary (Family Superset Configuration) . . . 79
Table 3. Pin startup and reset states . . . 86
Table 4. Power supply and reference pins . . . 86
Table 5. System pins . . . 87
Table 6. Pin multiplexing description . . . 88
Table 7. Peripheral multiplexing description . . . 123
Table 8. Cross-mapping between pads and package pins . . . 141
Table 9. Memory map (XBAR slave ports) . . . 144
Table 10. Flash memory map . . . 144
Table 11. Peripheral bridge 1 memory map . . . 145
Table 12. Peripheral bridge 0 memory map . . . 146
Table 13. UTest memory map . . . 148
Table 14. PMHF module categories . . . 151
Table 15. Module classification . . . 152
Table 16. ECC Engine Summary . . . 156
Table 17. FCCU failure inputs . . . 165
Table 18. MBIST partitioning . . . 176
Table 19. Self test configurations . . . 177
Table 20. MBIST configuration . . . 177
Table 21. LBIST configuration . . . 178
Table 22. Full self-test configuration . . . 179
Table 23. Bypass configuration . . . 181
Table 24. SPC570Sx core modules . . . 182
Table 25. Reset settings for e200z0h resources . . . 182
Table 26. Special purpose registers . . . 184
Table 27. Reference links to related information . . . 186
Table 28. Crossbar switch master assignments . . . 186
Table 29. Master ID number assignment . . . 187
Table 30. Crossbar switch slave assignments . . . 187
Table 31. Flash memory port assignments . . . 187
Table 32. Reference links to related information . . . 187
Table 33. Reference links to related information . . . 188
Table 34. SMPU_0 logical bus master assignments . . . 189
Table 35. SMPU port allocation . . . 189
Table 36. Reference links to related information . . . 190
Table 37. INTC implemented registers . . . 191
Table 38. Interrupt sources . . . 192
Table 39. Reference links to related information . . . 202
Table 40. DMAMUX peripheral DMA request to input source mapping . . . 202
Table 41. PIT periodic trigger to DMA channel mapping . . . 204
Table 42. Reference links to related information . . . 205
Table 43. Flash controller bus master assignments . . . 205
Table 44. Reference links to related information . . . 206
Table 45. Reference links to related information . . . 206
Table 46. Timers . . . 207
Table 47. System Timer Module (STM) instances . . . 208
Table 48. Software Watchdog Timer (SWT) instances . . . 208

Table 49. Register reset values . . . 208
Table 50. Reference links to related information . . . 209
Table 51. PIT information . . . 209
Table 52. eTimer instantiation . . . 210
Table 53. eTimer registers memory map . . . 210
Table 54. Reference links to related information . . . 211
Table 55. DSPI instantiation . . . 212
Table 56. DSPI registers memory map . . . 212
Table 57. LINFlexD configurations . . . 213
Table 58. LINFlexD register comparison map . . . 213
Table 59. LinFlexD and FlexCAN pin configuration . . . 215
Table 60. SSCM_MEMCONFIG reset values . . . 215
Table 61. SSCM startup information addresses . . . 216
Table 62. CRC contexts . . . 216
Table 63. MEMU error sources . . . 217
Table 64. Number of entries in each reporting table . . . 218
Table 65. Corresponding error sources for system RAM OFLW0 register . . . 218
Table 66. Corresponding error sources for peripheral RAM OFLW0 register . . . 219
Table 67. Corresponding error sources for flash memory OFLW0 register . . . 220
Table 68. MEMU registers . . . 221
Table 69. FCCU output signals . . . 224
Table 70. Reset values of selected FCCU registers . . . 224
Table 71. Reference links to related information . . . 226
Table 72. Protected SIUL2 registers . . . 226
Table 73. Protected CMU registers . . . 227
Table 74. Protected PMC_DIG registers . . . 228
Table 75. Protected PLL_DIG registers . . . 228
Table 76. Protected XOSC registers . . . 229
Table 77. Protected MEMU registers . . . 229
Table 78. Protected MC_ME registers . . . 229
Table 79. Protected MC_CGM registers . . . 231
Table 80. Protected MC_RGM registers . . . 231
Table 81. Protected SSCM registers . . . 232
Table 82. PASS DCF records . . . 235
Table 83. Reset pins . . . 237
Table 84. Destructive reset events . . . 238
Table 85. Functional reset events . . . 239
Table 86. Reset sequence phases . . . 242
Table 87. Module status during the reset phase . . . 243
Table 88. PORST electrical characteristics . . . 245
Table 89. PORST filter characteristics . . . 245
Table 90. PORST pull-down characteristics . . . 245
Table 91. SSCM boot record structure . . . 247
Table 92. SSCM boot record search locations . . . 248
Table 93. BAF boot record structure . . . 250
Table 94. BAF boot record search locations . . . 250
Table 95. Boot timings . . . 250
Table 96. DCF record field description . . . 252
Table 97. Series of DCF records in UTEST flash memory . . . 254
Table 98. DCF client list . . . 256
Table 99. UTEST Miscellaneous DCF Client field descriptions . . . 262
Table 100. PMC_REE bits DCF client field descriptions . . . 264

Table 101. SYSTEM modes . . . 265
Table 102. USER modes . . . 266
Table 103. Device power management controller external signals . . . 268
Table 104. 1.2 V voltage monitors . . . 271
Table 105. 3.3 V voltage monitors . . . 272
Table 106. 5 V voltage monitors . . . 272
Table 107. Voltage monitor features . . . 274
Table 108. POR and voltage monitors description . . . 275
Table 109. Timing descriptions . . . 277
Table 110. Calculating the BIST_TIME values . . . 277
Table 111. Monitors to check . . . 278
Table 112. SIUL2 memory map . . . 291
Table 113. SIUL2_MIDR1 field descriptions . . . 294
Table 114. SIUL2_MIDR2 field description . . . 295
Table 115. SIUL2_DISR0 field descriptions . . . 297
Table 116. SIUL2_DIRER0 field descriptions . . . 297
Table 117. SIUL2_DIRSR0 field descriptions . . . 298
Table 118. SIUL2_IREER0 field descriptions . . . 298
Table 119. SIUL2_IFEER0 field descriptions . . . 299
Table 120. SIUL2_IFER0 field descriptions . . . 300
Table 121. SIUL2_IFMCR0–SIUL2_IFMCR31 field descriptions . . . 300
Table 122. SIUL2_IFCPR field descriptions . . . 301
Table 123. SIUL2_MSCR0–SIUL2_MSCR511 field description . . . 303
Table 124. I/O MSCR Reset State Exceptions . . . 305
Table 125. SIUL2_MSCRn Reset value . . . 305
Table 126. SIUL2_MSCR512–SIUL2_MSCR1023 field descriptions . . . 306
Table 127. SIUL2_MSCRn – Reset value . . . 306
Table 128. SIUL2_GPDO0–SIUL2_GPDO79 field descriptions . . . 307
Table 129. SIUL2_GPDI0–SIUL2_GPDI79 field descriptions . . . 308
Table 130. SIUL2_PGPDOn field descriptions . . . 308
Table 131. SIUL2_PGPDIn field descriptions . . . 309
Table 132. SIUL2_MPGPDOn field descriptions . . . 310
Table 133. Interrupt source mapping to SIUL2 interrupt request output . . . 314
Table 134. XBAR memory map . . . 316
Table 135. XBAR_PRS*n* field descriptions . . . 317
Table 136. XBAR_CRS*n* field descriptions . . . 319
Table 137. XBAR_MGPCRn field descriptions . . . 320
Table 138. XBIC memory map . . . 326
Table 139. XBIC_MCR field descriptions . . . 327
Table 140. XBIC_EIR field descriptions . . . 328
Table 141. XBIC_ESR field descriptions . . . 329
Table 142. XBIC_EAR Register field descriptions . . . 331
Table 143. Hexadecimal attribute single-bit error syndromes . . . 331
Table 144. XBIC H Matrix Definition . . . 333
Table 145. Peripheral bridge memory map . . . 334
Table 146. MPROTn field descriptions . . . 337
Table 147. PACR memory map . . . 338
Table 148. PACR*n* field descriptions . . . 339
Table 149. OPACR memory map . . . 339
Table 150. OPACR*n* field descriptions . . . 342
Table 151. SMPU memory map . . . 347
Table 152. SMPU_CESR field descriptions . . . 348

Table 153. SMPU_EARn field descriptions . . . 349
Table 154. SMPU_EDR*n* field descriptions . . . 350
Table 155. SMPU_RGDn_WORD0 field descriptions . . . 351
Table 156. SMPU_RGDn_WORD1 field descriptions . . . 351
Table 157. SMPU_RGDn_WORD2_FMT0 field descriptions . . . 352
Table 158. SMPU_RGDn_WORD3 field descriptions . . . 354
Table 159. SMPU_RGDAACn field descriptions . . . 355
Table 160. Byte Strobe Assertion for Transfers . . . 357
Table 161. Protection violation definition . . . 358
Table 162. Overlapping region descriptor example . . . 362
Table 163. INTC memory map . . . 366
Table 164. INTC_BCR field descriptions . . . 367
Table 165. INTC_CPR0 field descriptions . . . 368
Table 166. INTC_IACKR0 field descriptions . . . 369
Table 167. INTC_EOIR0 field descriptions . . . 369
Table 168. INTC_SSCIRn field descriptions . . . 370
Table 169. INTC_PSRn field descriptions . . . 371
Table 170. INTC_PSRn field descriptions . . . 371
Table 171. Order of ISR execution example . . . 378
Table 172. Raised priority preserved events . . . 382
Table 173. DMA memory map . . . 387
Table 174. DMA_CR field descriptions . . . 390
Table 175. DMA_ES field descriptions . . . 391
Table 176. DMA_ERQL field descriptions . . . 393
Table 177. DMA_EEIL field descriptions . . . 395
Table 178. DMA_SERQ field descriptions . . . 397
Table 179. DMA_CERQ field descriptions . . . 397
Table 180. DMA_SEEI field descriptions . . . 398
Table 181. DMA_CEEI field descriptions . . . 399
Table 182. DMA_CINT field descriptions . . . 399
Table 183. DMA_CERR field descriptions . . . 400
Table 184. DMA_SSRT field descriptions . . . 400
Table 185. DMA_CDNE field descriptions . . . 401
Table 186. DMA_INTL field descriptions . . . 402
Table 187. DMA_ERRL field descriptions . . . 404
Table 188. DMA_HRSL field descriptions . . . 406
Table 189. DMA_DCHPRI*n* field descriptions . . . 407
Table 190. DMA_DCHMID*n* field descriptions . . . 408
Table 191. DMA_TCD*n*_SADDR field descriptions . . . 409
Table 192. DMA_TCD*n*_ATTR field descriptions . . . 409
Table 193. DMA_TCD*n*_SOFF field descriptions . . . 410
Table 194. DMA_TCD*n*_NBYTES_MLNO field descriptions . . . 411
Table 195. DMA_TCD*n*_NBYTES_MLOFFNO field descriptions . . . 412
Table 196. DMA_TCD*n*_NBYTES_MLOFFYES field descriptions . . . 413
Table 197. DMA_TCD*n*_SLAST field descriptions . . . 414
Table 198. DMA_TCD*n*_DADDR field descriptions . . . 414
Table 199. DMA_TCD*n*_CITER_ELINKYES field descriptions . . . 415
Table 200. DMA_TCD*n*_CITER_ELINKNO field descriptions . . . 416
Table 201. DMA_TCD*n*_DOFF field descriptions . . . 416
Table 202. DMA_TCD*n*_DLASTSGA field descriptions . . . 417
Table 203. DMA_TCD*n*_BITER_ELINKYES field descriptions . . . 418
Table 204. DMA_TCD*n*_BITER_ELINKNO field descriptions . . . 419

Table 205. DMA_TCD*n*_CSR field descriptions . . . 420
Table 206. eDMA peak transfer rates (Mbyte/s) . . . 428
Table 207. eDMA peak request rate (MReq/s) . . . 430
Table 208. TCD control and status fields . . . 431
Table 209. Modulo feature example . . . 437
Table 210. TCD status sequence – software activated channel . . . 437
Table 211. TCD status sequence – hardware activated channel . . . 438
Table 212. Channel linking parameters . . . 439
Table 213. Coherency model for a dynamic channel link request . . . 440
Table 214. Coherency model for method 1 . . . 441
Table 215. Coherency model for method 2 . . . 441
Table 216. DMACHMUX memory map . . . 444
Table 217. CHCFGn field descriptions . . . 445
Table 218. Channel and trigger enabling . . . 445
Table 219. WKPU Memory Map . . . 455
Table 220. NSR Field Descriptions . . . 456
Table 221. NCR Field Descriptions . . . 458
Table 222. TSENS calibration constants . . . 462
Table 223. MC_CGM relationship to clocks . . . 467
Table 224. Maximum system level clock frequencies . . . 467
Table 225. JTAG frequencies . . . 468
Table 226. Dual PLL Digital Interface register reset values . . . 470
Table 227. XOSC register reset values . . . 472
Table 228. Clock input sources . . . 476
Table 229. CMU registers availability . . . 476
Table 230. Dual PLL Digital Interface memory map . . . 479
Table 231. PLL0CR field descriptions . . . 480
Table 232. PLL0SR field descriptions . . . 481
Table 233. PLL0DV field descriptions . . . 482
Table 234. PLL1CR field descriptions . . . 484
Table 235. PLL1SR field descriptions . . . 485
Table 236. PLL1DV field descriptions . . . 486
Table 237. PLL1FM field descriptions . . . 487
Table 238. PLL1FD field descriptions . . . 488
Table 239. Signal description . . . 495
Table 240. CMU memory map . . . 495
Table 241. CMU_CSR field descriptions . . . 496
Table 242. CMU_FDR field descriptions . . . 497
Table 243. CMU_HFREFR field descriptions . . . 497
Table 244. CMU_LFREFR field descriptions . . . 498
Table 245. CMU_ISR field descriptions . . . 499
Table 246. CMU_MDR field descriptions . . . 500
Table 247. MC_CGM register description . . . 504
Table 248. CGM_PCS_SDUR field descriptions . . . 506
Table 249. CGM_PCS_DIVC1 field descriptions . . . 507
Table 250. CGM_PCS_DIVS1 field descriptions . . . 507
Table 251. CGM_PCS_DIVE1 field descriptions . . . 508
Table 252. CGM_PCS_DIVC2 field descriptions . . . 509
Table 253. CGM_PCS_DIVS2 field descriptions . . . 509
Table 254. CGM_PCS_DIVE2 field descriptions . . . 510
Table 255. CGM_PCS_DIVC4 field descriptions . . . 511
Table 256. CGM_PCS_DIVS4 field descriptions . . . 511

Table 257. CGM_PCS_DIVE4 field descriptions . . . 512
Table 258. CGM_SC_DIV_RC field descriptions . . . 513
Table 259. CGM_DIV_UPD_TYPE field descriptions . . . 514
Table 260. CGM_DIV_UPD_TRIG field descriptions . . . 515
Table 261. CGM_DIV_UPD_STAT field descriptions . . . 515
Table 262. CGM_SC_SS field descriptions . . . 516
Table 263. CGM_SC_DC0 field descriptions . . . 517
Table 264. CGM_SC_DC1 field descriptions . . . 518
Table 265. CGM_SC_DC2 field descriptions . . . 519
Table 266. CGM_AC0_SC field descriptions . . . 520
Table 267. CGM_AC0_SS field descriptions . . . 521
Table 268. CGM_AC0_DC0 field descriptions . . . 522
Table 269. CGM_AC0_DC1 field descriptions . . . 523
Table 270. CGM_AC0_DC2 field descriptions . . . 523
Table 271. CGM_AC0_DC3 field descriptions . . . 524
Table 272. CGM_AC0_DC4 field descriptions . . . 525
Table 273. CGM_AC0_DC5 field descriptions . . . 525
Table 274. CGM_AC1_SC field descriptions . . . 526
Table 275. CGM_AC1_SS field descriptions . . . 527
Table 276. CGM_AC1_DC0 field descriptions . . . 528
Table 277. CGM_AC2_SC field descriptions . . . 529
Table 278. CGM_AC2_SS field descriptions . . . 530
Table 279. CGM_AC3_SC field descriptions . . . 531
Table 280. CGM_AC3_SS field descriptions . . . 532
Table 281. MC_CGM CGM_PCS_DIVCn[RATE] values . . . 535
Table 282. MC_CGM Example System Clock Division Values Compatible with Divider Synchronization . . . 537
Table 283. XOSC configurations . . . 543
Table 284. XOSC memory map . . . 544
Table 285. XOSC_CTL register field descriptions . . . 545
Table 286. IRCOSC memory map . . . 547
Table 287. IRCOSC_CTL field descriptions . . . 548
Table 288. IRCOSC_CTL[USER_TRIM] frequency trimming calculation . . . 548
Table 289. IRCOSC_NT field descriptions . . . 549
Table 290. PRAM controller memory map . . . 551
Table 291. PRCR1 field descriptions . . . 552
Table 292. PRAM controller basic H-matrix definition . . . 555
Table 293. Flash memory controller memory map . . . 561
Table 294. PFCR1 field descriptions . . . 562
Table 295. PFCR2 field descriptions . . . 565
Table 296. PFCR3 field descriptions . . . 568
Table 297. PFAPR field descriptions . . . 570
Table 298. Flash memory partition 0 memory map . . . 583
Table 299. Flash Memory Partition 1 memory map . . . 583
Table 300. TestFlash block memory map . . . 584
Table 301. Flash User Registers memory map . . . 586
Table 302. MCR field descriptions . . . 588
Table 303. MCR Bits Set/Clear Priority Levels . . . 594
Table 304. MCRE field descriptions . . . 595
Table 305. LOCK0 field descriptions . . . 598
Table 306. LOCK1 field descriptions . . . 599
Table 307. LOCK2 field descriptions . . . 600

Table 308. SEL0 field descriptions . . . 601
Table 309. SEL1 field descriptions . . . 602
Table 310. SEL2 field descriptions . . . 603
Table 311. ADR update mode list . . . 603
Table 312. ADR content: priority list . . . 604
Table 313. ADR field descriptions . . . 605
Table 314. Flash memory partition 0 addresses mapping . . . 605
Table 315. Flash Memory Partition 1 addresses mapping . . . 606
Table 316. UT0 field descriptions . . . 607
Table 317. UM0 field descriptions . . . 610
Table 318. UM1 field descriptions . . . 610
Table 319. UM2 field descriptions . . . 611
Table 320. UM3 field descriptions . . . 612
Table 321. UM8 field descriptions . . . 612
Table 322. UM9 field descriptions . . . 613
Table 323. OPP0 field descriptions . . . 614
Table 324. OPP1 field descriptions . . . 615
Table 325. OPP2 field descriptions . . . 615
Table 326. TDM field descriptions . . . 616
Table 327. Bits Manipulation: doublewords with the same ECC value . . . 620
Table 328. Flash Modify operations . . . 621
Table 329. Test Mode Disable block select . . . 632
Table 330. EIM memory map . . . 633
Table 331. EIM_CTRL field descriptions . . . 634
Table 332. EIM_DATA field descriptions . . . 635
Table 333. SPC570Sx analog input pin multiplexing . . . 641
Table 334. SARADC external signal description . . . 642
Table 335. SARADC analog input channel assignment . . . 646
Table 336. SARADC_B analog test channel assignment . . . 646
Table 337. SARADC register descriptions . . . 647
Table 338. SARADC test channel register descriptions (SARADC_B only) . . . 648
Table 339. SARADC_B register definitions . . . 648
Table 340. SARADC_B test channel register definitions . . . 652
Table 341. SARADC_0 register definitions . . . 655
Table 342. CTU modes . . . 667
Table 343. Values of WDGxH and WDGxL fields . . . 670
Table 344. SARADC digital interface register map . . . 673
Table 345. MCR field descriptions . . . 675
Table 346. MSR field descriptions . . . 678
Table 347. ISR field descriptions . . . 680
Table 348. ICIPR0–ICIPR2 field descriptions . . . 681
Table 349. ICIPR0–2 Registers to channel association . . . 681
Table 350. IMR field descriptions . . . 682
Table 351. ICIMR0–ICIMR2 field descriptions . . . 683
Table 352. Interrupt Mask Registers to Channel Association . . . 683
Table 353. WTISR field descriptions . . . 684
Table 354. WTIMR field descriptions . . . 685
Table 355. DMAE register field descriptions . . . 686
Table 356. ICDSR0–ICDSR2 field descriptions . . . 687
Table 357. Internal Channel DMA Select Registers to Channel Association . . . 687
Table 358. WTHRHLR0–WTHRHLR3 field descriptions . . . 687
Table 359. CTR0–CTRL3 field descriptions . . . 688

Table 360. ICNCMR0–ICNCMR2 field descriptions . . . 689
Table 361. Internal Channel Normal Conversion Mask Registers to Channel Association . . . 689
Table 362. ICJCMR0–ICJCMR2 field descriptions . . . 690
Table 363. Internal Channel Normal Conversion Mask Registers to Channel Association . . . 690
Table 364. PDEDR field descriptions . . . 691
Table 365. ICDR0–ICDR95 field descriptions . . . 692
Table 366. ICWSELR0–ICWSELR11 field descriptions . . . 694
Table 367. Internal Channel Watchdog Enable Registers to Channel Association . . . 694
Table 368. ICWENR0–ICWENR2 field descriptions . . . 695
Table 369. Internal Channel Watchdog Enable Registers to Channel Association . . . 695
Table 370. ICAWORR0–ICAWORR2 field descriptions . . . 696
Table 371. Internal Channel Analog Watchdog Registers to Channel Association . . . 696
Table 372. TCIPR field descriptions . . . 697
Table 373. TCIPR Register to Channel Association . . . 697
Table 374. TCIMR field descriptions . . . 697
Table 375. Test Channel Interrupt Mask Register to Channel Association . . . 697
Table 376. TCDSR field descriptions . . . 698
Table 377. Test Channel DMA Select Registers to Channel Association . . . 698
Table 378. TCNCMR field descriptions . . . 699
Table 379. Test Channel Normal Conversion Mask Register to Channel Association . . . 699
Table 380. TCJCMR field descriptions . . . 700
Table 381. Test Channel Normal Conversion Mask Register to Channel Association . . . 700
Table 382. TCWSELR0–TCWSELR3 field descriptions . . . 700
Table 383. Test Channel Watchdog Select Registers to Channel Association . . . 701
Table 384. TCWENR field descriptions . . . 702
Table 385. Test Channel Watchdog Enable Registers to Channel Association . . . 702
Table 386. TCAWORR field descriptions . . . 702
Table 387. Analog Watchdog Register to Channel Association . . . 703
Table 388. Test Channel Connection with Analog Pin Registers 0–7 (TCCAPR0–TCCAPR7) field descriptions . . . 704
Table 389. Test Channel Short Registers to Channel Association . . . 704
Table 390. TCDR96–TCDR127 field descriptions . . . 705
Table 391. Start of conversion pulse delay table . . . 707
Table 392. CTU interrupts . . . 722
Table 393. CTU memory map . . . 723
Table 394. TGS registers . . . 725
Table 395. SU registers . . . 726
Table 396. CTU registers . . . 726
Table 397. FIFO registers . . . 726
Table 398. TGSISR field descriptions . . . 727
Table 399. TGSCR field descriptions . . . 729
Table 400. TnCR field descriptions . . . 730
Table 401. TGSCCR field format . . . 730
Table 402. TGSCRR field descriptions . . . 731
Table 403. CLCR1 field descriptions . . . 731
Table 404. CLCR2 field descriptions . . . 732
Table 405. THCR1 field descriptions . . . 733
Table 406. THCR2 field descriptions . . . 734
Table 407. CLR*n* (CMS = 0) field descriptions . . . 736
Table 408. CLRn (CMS = 1) field descriptions . . . 737
Table 409. FDCR field descriptions . . . 738
Table 410. FCR field descriptions . . . 738

Table 411. FTH field descriptions . . . 740
Table 412. FST field descriptions . . . 741
Table 413. FR*n* field descriptions . . . 742
Table 414. FL*n* field descriptions . . . 743
Table 415. CTUEFR field descriptions . . . 743
Table 416. CTUIFR field descriptions . . . 744
Table 417. CTUIR field descriptions . . . 745
Table 418. COTR field descriptions . . . 746
Table 419. CTUCR field descriptions . . . 747
Table 420. CTUDF field descriptions . . . 748
Table 421. eTimer Memory Map . . . 762
Table 422. Compare Register 1n (COMP1n) field descriptions . . . 764
Table 423. Compare Register 2n (COMP2n) field descriptions . . . 764
Table 424. Capture Register 1n (CAPT1n) field descriptions . . . 765
Table 425. Capture Register 2n (CAPT2n) field descriptions . . . 765
Table 426. Load Register n (LOADn) field descriptions . . . 765
Table 427. Hold Register n (HOLDn) field descriptions . . . 766
Table 428. Counter Register n (CNTRn) field descriptions . . . 766
Table 429. Control Register 1n (CTRL1n) field descriptions . . . 767
Table 430. Control Register 2n (CTRL2n) field descriptions . . . 769
Table 431. Control Register 3n (CTRL3n) field descriptions . . . 771
Table 432. Status Register n (STSn) field descriptions . . . 773
Table 433. Interrupt and DMA Enable Register n (INTDMAn) field descriptions . . . 774
Table 434. CMPLD1n field descriptions . . . 775
Table 435. CMPLD2n field descriptions . . . 776
Table 436. Compare and Capture Control Register n (CCCTRLn) field descriptions . . . 776
Table 437. Input Filter Register n (FILTn) field descriptions . . . 778
Table 438. WDTOL field descriptions . . . 779
Table 439. WDTOH field descriptions . . . 780
Table 440. FCTRL field descriptions . . . 781
Table 441. FSTS field descriptions . . . 781
Table 442. FFILT field descriptions . . . 782
Table 443. Channel Enable Register (ENBL) field descriptions . . . 782
Table 444. DMA Request 0 Select Register (DREQ0) field descriptions . . . 784
Table 445. DMA Request 1 Select Register (DREQ1) field descriptions . . . 785
Table 446. Interrupt Summary . . . 787
Table 447. DMA Summary . . . 788
Table 448. STM memory map . . . 790
Table 449. STM_CR field descriptions . . . 791
Table 450. STM_CNT field descriptions . . . 791
Table 451. STM_CCRn field descriptions . . . 792
Table 452. STM_CIRn field descriptions . . . 792
Table 453. STM_CMPn field descriptions . . . 793
Table 454. SWT memory map . . . 795
Table 455. SWT_CR field descriptions . . . 796
Table 456. SWT_IR field descriptions . . . 797
Table 457. SWT_TO field descriptions . . . 797
Table 458. SWT_WN field descriptions . . . 798
Table 459. SWT_SR field descriptions . . . 798
Table 460. SWT_CO field descriptions . . . 799
Table 461. SWT_SK field descriptions . . . 799
Table 462. PIT memory map . . . 804

Table 463. MCR field descriptions . . . 805
Table 464. LDVALn field descriptions . . . 806
Table 465. CVALn field descriptions . . . 806
Table 466. TCTRLn field descriptions . . . 807
Table 467. TFLGn field descriptions . . . 807
Table 468. FlexCAN signals . . . 814
Table 469. FlexCAN module memory map . . . 815
Table 470. FlexCAN register reset status . . . 816
Table 471. Message Buffer MB0 memory mapping . . . 817
Table 472. Message Buffer structure field description . . . 817
Table 473. Message buffer code for Rx buffers . . . 819
Table 474. Message Buffer code for Tx buffers . . . 820
Table 475. MB0–MB31 addresses . . . 820
Table 476. Rx FIFO Structure field description . . . 822
Table 477. MCR field descriptions . . . 823
Table 478. IDAM coding . . . 826
Table 479. CTRL field descriptions . . . 827
Table 480. TIMER field descriptions . . . 830
Table 481. RXGMASK field description . . . 831
Table 482. RX14MASK field description . . . 832
Table 483. RX15MASK field description . . . 833
Table 484. Error Counter Register (ECR) field descriptions . . . 834
Table 485. Error and Status Register (ESR) field description . . . 835
Table 486. Fault confinement state . . . 837
Table 487. IMASK1 field descriptions . . . 838
Table 488. IFLAG1 field descriptions . . . 839
Table 489. RXIMRn field descriptions . . . 840
Table 490. Recommended FEN and BCC settings . . . 848
Table 491. Time segment syntax . . . 851
Table 492. CAN standard compliant bit time segment settings . . . 851
Table 493. Minimum ratio between peripheral clock frequency and CAN bit rate . . . 852
Table 494. DSPI signal description . . . 861
Table 495. DSPI memory map . . . 863
Table 496. DSPI*x*_MCR field descriptions . . . 864
Table 497. DSPIx_TCR field descriptions . . . 869
Table 498. DSPI*x*_CTAR*n* field descriptions . . . 870
Table 499. DSPI SCK duty cycle . . . 873
Table 500. DSPI*x*_CTAR0_SLAVE field descriptions . . . 874
Table 501. DSPI*x*_CTAR1_SLAVE field descriptions . . . 875
Table 502. DSPI*x*_SR field descriptions . . . 876
Table 503. DSPI*x*_RSER field descriptions . . . 879
Table 504. DSPI*x*_PUSHR field descriptions . . . 882
Table 505. DSPI*x*_PUSHR_SLAVE field descriptions . . . 884
Table 506. DSPI*x*_POPR field descriptions . . . 885
Table 507. DSPI*x*_TXFRn field descriptions . . . 886
Table 508. DSPI*x*_RXFRn field descriptions . . . 886
Table 509. DSPI*x*_CTARE*n* field descriptions . . . 887
Table 510. DSPI*x*_SREX field descriptions . . . 888
Table 511. Baud rate computation example . . . 894
Table 512. PCS to SCK delay computation example . . . 894
Table 513. After SCK delay computation example . . . 894
Table 514. Delay after transfer computation example . . . 895

Table 515. Peripheral Chip Select Strobe Assert Computation Example . . . 896
Table 516. Peripheral Chip Select Strobe Negate Computation Example . . . 896
Table 517. Interrupt and DMA request conditions . . . 908
Table 518. Baud rate values (bps) . . . 913
Table 519. Delay values . . . 914
Table 520. 10-bit-shift-register sample register . . . 933
Table 521. BDRL access in UART mode . . . 941
Table 522. BDRM access in UART mode, . . . 942
Table 523. LINFlexD control/status fields description . . . 944
Table 524. LINFlexD DMA control fields description . . . 946
Table 525. TCD control fields description . . . 946
Table 526. Master node: Tx mode - Register setting . . . 948
Table 527. TCD setting: Master node -Tx mode . . . 948
Table 528. TCD setting: Master node - Rx mode . . . 950
Table 529. Slave node: Tx mode - Register setting . . . 952
Table 530. TCD setting: Slave node - Tx mode . . . 952
Table 531. Slave node: Rx mode - Register setting . . . 954
Table 532. TCD setting: Slave node - Rx mode . . . 955
Table 533. TCD setting: UART - Tx mode . . . 957
Table 534. TCD setting: UART - Rx mode . . . 959
Table 535. LINFlexD memory map . . . 962
Table 536. LINCR1 field descriptions . . . 964
Table 537. LINIER field descriptions . . . 967
Table 538. LINSR field descriptions . . . 969
Table 539. LINESR field descriptions . . . 972
Table 540. UARTCR field descriptions . . . 974
Table 541. UARTSR field descriptions . . . 978
Table 542. LINTCSR field descriptions . . . 980
Table 543. LINOCR field descriptions . . . 981
Table 544. LINTOCR field descriptions . . . 982
Table 545. LINFBRR field descriptions . . . 983
Table 546. LINIBRR field descriptions . . . 984
Table 547. LINCFR field descriptions . . . 985
Table 548. LINCR2 field descriptions . . . 986
Table 549. BIDR field descriptions . . . 987
Table 550. BDRL field descriptions . . . 988
Table 551. BDRM field descriptions . . . 989
Table 552. IFER field descriptions . . . 989
Table 553. IFMI field descriptions . . . 990
Table 554. IFMR field descriptions . . . 990
Table 555. IFCR2n field descriptions . . . 991
Table 556. IFCR2n+1 field descriptions . . . 992
Table 557. GCR field descriptions . . . 993
Table 558. UARTPTO field descriptions . . . 995
Table 559. UARTCTO field descriptions . . . 996
Table 560. DMATXE field descriptions . . . 996
Table 561. DMARXE field descriptions . . . 997
Table 562. Interrupts . . . 1004
Table 563. MC_RGM memory map . . . 1009
Table 564. RGM_DES register field descriptions . . . 1012
Table 565. RGM_DERD register field descriptions . . . 1013
Table 566. RGM_DEAR register field descriptions . . . 1015

Table 567. RGM_DBRE register field descriptions . . . 1015
Table 568. Functional Event Status Register (RGM_FES) field descriptions . . . 1017
Table 569. RGM_FERD register field descriptions . . . 1019
Table 570. RGM_FEAR register field descriptions . . . 1021
Table 571. Functional Event Status Register (RGM_FBRE) field descriptions . . . 1022
Table 572. RGM_FESS register field descriptions . . . 1023
Table 573. RGM_FRET register field descriptions . . . 1025
Table 574. RGM_DRET register field descriptions . . . 1026
Table 575. RGM_PRST*0 register* field descriptions . . . 1027
Table 576. RGM_PRST*1* field descriptions . . . 1028
Table 577. RGM_PRST*2 register* field descriptions . . . 1029
Table 578. RGM_PRST*3 register* field descriptions . . . 1030
Table 579. RGM_PRST*4 register* field descriptions . . . 1032
Table 580. RGM_PRST*5 register* field descriptions . . . 1033
Table 581. RGM_PRST*6* register field descriptions . . . 1034
Table 582. RGM_PRST*7 register* field descriptions . . . 1035
Table 583. RGM_PSTAT0 register field descriptions . . . 1036
Table 584. RGM_PSTAT*1 register* field descriptions . . . 1036
Table 585. RGM_PSTAT*2 register* field descriptions . . . 1037
Table 586. RGM_PSTAT*3 register* field descriptions . . . 1038
Table 587. RGM_PSTAT*4 register* field descriptions . . . 1039
Table 588. RGM_PSTAT*5 register* field descriptions . . . 1040
Table 589. RGM_PSTAT*6*) register field descriptions . . . 1041
Table 590. RGM_PSTAT*7* register field descriptions . . . 1042
Table 591. MC_RGM reset implications . . . 1042
Table 592. MC_RGM Alternate Event Selection . . . 1047
Table 593. BAF memory organization . . . 1049
Table 594. BAF image header . . . 1049
Table 595. BAF image version . . . 1050
Table 596. BAF DCF client list . . . 1053
Table 597. BAF DCF ADDR client description . . . 1054
Table 598. DATA32 DCF client configuration . . . 1055
Table 599. DATA16 DCF client configuration . . . 1055
Table 600. DATA8 DCF client configuration . . . 1056
Table 601. Serial boot callback function return value definition . . . 1057
Table 602. LinFlexD_0 and FlexCAN_1 baud rates . . . 1063
Table 603. LINFlexD serial boot mode download protocol . . . 1063
Table 604. FlexCAN serial boot mode download protocol . . . 1064
Table 605. SSCM memory map . . . 1067
Table 606. STATUS field descriptions . . . 1068
Table 607. MEMCONFIG field descriptions . . . 1069
Table 608. ERROR field descriptions . . . 1070
Table 609. PSA field descriptions . . . 1070
Table 610. LCSTAT field descriptions . . . 1071
Table 611. Series of DCF records in UTEST . . . 1073
Table 612. Life Cycle slots . . . 1076
Table 613. Life Cycle Slots in Memory . . . 1077
Table 614. Resulting Life Cycle . . . 1077
Table 615. Power Management Controller Memory Map . . . 1080
Table 616. 32-bit registers LV0, LV1, HV0, HV1 arrangement table . . . 1082
Table 617. VD number versus VD name table . . . 1082
Table 618. EPR_LV0 field descriptions . . . 1083

Table 619. REE_LV0 field descriptions . . . 1084
Table 620. RES_LV0 field descriptions . . . 1085
Table 621. IE_LV0 field descriptions . . . 1085
Table 622. FEE_LV0 field descriptions . . . 1086
Table 623. EPR_LV1 field descriptions . . . 1087
Table 624. REE_LV1 field descriptions . . . 1087
Table 625. RES_LV1 field descriptions . . . 1088
Table 626. IE_LV1 field descriptions . . . 1089
Table 627. FEE_LV1 field descriptions . . . 1089
Table 628. EPR_HV0 field descriptions . . . 1090
Table 629. REE_HV0 field descriptions . . . 1091
Table 630. RES_HV0 field descriptions . . . 1092
Table 631. IE_HV0 field descriptions . . . 1093
Table 632. FEE_HV0 field descriptions . . . 1094
Table 633. EPR_HV1 field descriptions . . . 1095
Table 634. REE_HV1 field descriptions . . . 1096
Table 635. RES_HV1 field descriptions . . . 1097
Table 636. IE_HV1 field descriptions . . . 1098
Table 637. FEE_HV1 field descriptions . . . 1099
Table 638. *GR_S* field descriptions . . . 1100
Table 639. GR_P field descriptions . . . 1104
Table 640. IE_G field descriptions . . . 1105
Table 641. MREG_CTRL field descriptions . . . 1106
Table 642. EPR_TD field descriptions . . . 1107
Table 643. REE_TD field descriptions . . . 1109
Table 644. RES_TD field descriptions . . . 1110
Table 645. CTL_TD field descriptions . . . 1111
Table 646. Temperature Sensor FEE_TD field descriptions . . . 1112
Table 647. ESR0_CFG field descriptions . . . 1113
Table 648. BIST_FLAGS field description . . . 1114
Table 649. BIST_CTRL field descriptions . . . 1115
Table 650. BIST_TIME10 field descriptions . . . 1116
Table 651. BIST_TIME32 field descriptions . . . 1117
Table 652. BIST_TIME65 field descriptions . . . 1117
Table 653. BIST_DEBUG field description . . . 1118
Table 654. MC_PCU register description . . . 1122
Table 655. MC_PCU memory map . . . 1122
Table 656. Power Domain Status Register (PCU_PSTAT) field descriptions . . . 1123
Table 657. MC_ME mode descriptions . . . 1126
Table 658. MC_ME register description . . . 1128
Table 659. ME_GS field descriptions . . . 1131
Table 660. ME_MCTL field descriptions . . . 1134
Table 661. ME_ME field descriptions . . . 1135
Table 662. ME_IS field descriptions . . . 1136
Table 663. ME_IM field descriptions . . . 1137
Table 664. ME_IMTS field descriptions . . . 1138
Table 665. ME_DMTS field descriptions . . . 1140
Table 666. ME_RESET_MC field descriptions . . . 1144
Table 667. ME_TEST_MC field descriptions . . . 1146
Table 668. ME_SAFE_MC field descriptions . . . 1148
Table 669. ME_DRUN_MC field descriptions . . . 1150
Table 670. ME_RUNn_MC field descriptions . . . 1152

Table 671. ME_HALT0_MC field descriptions . . . 1154
Table 672. ME_STOP0_MC field descriptions . . . 1156
Table 673. ME_PS*n* field descriptions . . . 1161
Table 674. ME_RUN_PCn field descriptions . . . 1161
Table 675. ME_LP_PCn field descriptions . . . 1162
Table 676. ME_PCTLn field descriptions . . . 1163
Table 677. ME_CS field descriptions . . . 1165
Table 678. ME_CCTL0 field descriptions . . . 1165
Table 679. ME_CADDR0 field descriptions . . . 1167
Table 680. MC_ME resource control overview . . . 1172
Table 681. MC_ME system clock selection overview . . . 1176
Table 682. DBCR0 field descriptions . . . 1192
Table 683. DBCR1 field descriptions . . . 1194
Table 684. DBCR2 field descriptions . . . 1196
Table 685. DBSR field descriptions . . . 1198
Table 686. JTAG/OnCE primary interface signals . . . 1203
Table 687. OnCE Status register field descriptions . . . 1207
Table 688. OnCE Command register field descriptions . . . 1208
Table 689. e200z0h OnCE register addressing . . . 1209
Table 690. OnCE Control register field descriptions . . . 1210
Table 691. OnCE register access requirements . . . 1213
Table 692. CTL field descriptions . . . 1217
Table 693. Watchpoint output signal assignments . . . 1221
Table 694. DCI_CR field descriptions . . . 1227
Table 695. DCI_PINCR field descriptions . . . 1228
Table 696. Nexus trace source identifiers . . . 1229
Table 697. L1SEL0 register field descriptions . . . 1230
Table 698. L1SEL1 register field descriptions . . . 1232
Table 699. L1SEL2 register field descriptions . . . 1233
Table 700. L1SEL3 register field descriptions . . . 1235
Table 701. L1SEL4 register field descriptions . . . 1236
Table 702. L1SEL5 register field descriptions . . . 1238
Table 703. L1SEL6 register field descriptions . . . 1239
Table 704. L1SEL7 register field descriptions . . . 1241
Table 705. JTAG signals . . . 1242
Table 706. Nexus signals . . . 1243
Table 707. DCI JTAG source selection . . . 1250
Table 708. DCI operating modes . . . 1250
Table 709. Nexus reset truth table . . . 1251
Table 710. JTAG signal properties . . . 1254
Table 711. Device identification register field descriptions . . . 1256
Table 712. JTAG_PASSWORD register field descriptions . . . 1257
Table 713. 6-bit JTAG instructions . . . 1260
Table 714. JDC memory map . . . 1266
Table 715. JDC_MCR register field descriptions . . . 1266
Table 716. JDC_MSR register field descriptions . . . 1267
Table 717. JDC_JOUT_IPS register field descriptions . . . 1268
Table 718. JDC_JIN_IPS register field descriptions . . . 1268
Table 719. JOUT JTAG register field descriptions . . . 1269
Table 720. JIN JTAG register field descriptions . . . 1269
Table 721. JTAG instructions . . . 1270
Table 722. Memory map . . . 1274

Table 723. JTAGM_MCR register field descriptions . . . 1276
Table 724. JTAGM_SR register field descriptions . . . 1278
Table 725. JTAGM_DOR0 register field descriptions . . . 1279
Table 726. JTAGM_DOR1 register field descriptions . . . 1279
Table 727. JTAGM_DOR2 register field descriptions . . . 1280
Table 728. JTAGM_DOR3 register field descriptions . . . 1280
Table 729. JTAGM_DIR0 register field descriptions . . . 1280
Table 730. JTAGM_DIR1 register field descriptions . . . 1281
Table 731. Device parameter values . . . 1282
Table 732. NPC signal properties . . . 1285
Table 733. NPC registers . . . 1286
Table 734. DID field descriptions . . . 1288
Table 735. PCR field descriptions . . . 1289
Table 736. PCR[MCKO_DIV] values . . . 1290
Table 737. NPC reset configuration options . . . 1290
Table 738. NPC output messages . . . 1292
Table 739. Implemented instructions . . . 1293
Table 740. Loading NEXUS-ENABLE instruction . . . 1295
Table 741. Write to a 32-bit Nexus Client Register . . . 1296
Table 742. SPU register summary . . . 1305
Table 743. ITLD2 register field descriptions . . . 1316
Table 744. ITLD3 register field descriptions . . . 1318
Table 745. C0PEVP register field descriptions . . . 1321
Table 746. C0PIS register field descriptions . . . 1321
Table 747. L2nSEL0 register field descriptions . . . 1323
Table 748. L2nSEL1 register field descriptions . . . 1324
Table 749. L2nSEL2 register field descriptions . . . 1326
Table 750. L2nSEL3 register field descriptions . . . 1327
Table 751. IOICn register field descriptions . . . 1328
Table 752. SCTRL register field descriptions . . . 1330
Table 753. SnGC register field descriptions . . . 1332
Table 754. SS register field descriptions . . . 1334
Table 755. SE register field descriptions . . . 1336
Table 756. SPU action decoder . . . 1337
Table 757. SnTA0 register field descriptions . . . 1339
Table 758. SnTA1 register field descriptions . . . 1339
Table 759. SnFA0 register field descriptions . . . 1340
Table 760. SnFA1 register field descriptions . . . 1341
Table 761. Trace group configuration register field descriptions . . . 1342
Table 762. INTS register field descriptions . . . 1342
Table 763. CCTRLn register field descriptions . . . 1343
Table 764. CCMPn register field descriptions . . . 1344
Table 765. CCOMS register field descriptions . . . 1345
Table 766. COS register field descriptions . . . 1347
Table 767. CCAPS register field descriptions . . . 1348
Table 768. Processor exception processing conditions . . . 1352
Table 769. Terms and Definitions . . . 1358
Table 770. Supported TCODEs . . . 1362
Table 771. Error Code (ECODE) Encoding (TCODE = 8) . . . 1365
Table 772. Error Type (ETYPE) Encoding (TCODE = 8) . . . 1365
Table 773. Resource Code (RCODE) values (TCODE = 27) . . . 1365
Table 774. Event Code (EVCODE) Encoding (TCODE = 33) . . . 1366

Table 775. Nexus 2+ Register Map . . . 1366
Table 776. CSC Register field descriptions . . . 1367
Table 777. PCR field descriptions . . . 1368
Table 778. DC1 Register field descritions . . . 1369
Table 779. DC2 Register field descriptions . . . 1371
Table 780. DC3 Register field descriptions . . . 1372
Table 781. DC4 Register Field description . . . 1373
Table 782. DS Register field descriptions . . . 1374
Table 783. WT Register Field descriptions . . . 1375
Table 784. PTSTC Register field descriptions . . . 1376
Table 785. PTETC Register field descriptions . . . 1377
Table 786. WMSK Register field descriptions . . . 1378
Table 787. OVCR field descriptions . . . 1379
Table 788. RWCS register field descriptions . . . 1380
Table 789. Read/Write Access Status Bit Encoding . . . 1381
Table 790. RWD data placement for Transfers . . . 1382
Table 791. RWD byte lane mapping . . . 1382
Table 792. JTAG DR field descriptions . . . 1383
Table 793. Message Type Priority and Message Dropped Responses . . . 1386
Table 794. Indirect Branch Message Sources . . . 1390
Table 795. Direct Branch Message Sources . . . 1390
Table 796. Branch/Predicate History Events . . . 1393
Table 797. RCODE Encoding . . . 1394
Table 798. Program Trace Exception Summary . . . 1396
Table 799. Watchpoint Source Encoding . . . 1400
Table 800. JTAG Pins for Nexus 2+ . . . 1405
Table 801. Nexus 2+ Auxiliary Pins . . . 1406
Table 802. Nexus Port Arbitration Signals . . . 1407
Table 803. MSEO Pin(s) Protocol . . . 1407
Table 804. MDO Request Encodings . . . 1410
Table 805. Indirect Branch Message Example (2 MDO / 1 MSEO) . . . 1411
Table 806. Indirect Branch Message Example (8 MDO / 2 MSEO) . . . 1411
Table 807. Direct Branch Message Example (2 MDO / 1 MSEO) . . . 1412
Table 808. Direct Branch Message Example (8 MDO / 2 MSEO) . . . 1412
Table 809. Accessing Internal Nexus 2+ Registers via JTAG/OnCE . . . 1412
Table 810. Accessing Memory-Mapped Resources (Reads) . . . 1413
Table 811. Accessing Memory-Mapped Resources (Writes) . . . 1413
Table 812. JTAG pins for NXMC . . . 1416
Table 813. NXMC auxiliary pins . . . 1416
Table 814. Supported public TCODEs . . . 1417
Table 815. Data trace size (DSZ) encodings (TCODE = 5,6,13,14) . . . 1418
Table 816. Error Type (ETYPE) encodings (applicable to all error messages) . . . 1418
Table 817. Error Code (ECODE) encodings (applicable based on error source) . . . 1419
Table 818. NXMC registers . . . 1419
Table 819. DC1 register field description . . . 1420
Table 820. DC2 register field description . . . 1421
Table 821. WT register field description . . . 1422
Table 822. DTC field description . . . 1423
Table 823. Data trace address range options . . . 1424
Table 824. BWC1 field description . . . 1425
Table 825. BWC2 field description . . . 1426
Table 826. JTAG DR field description for Nexus register access . . . 1428

Table 827. Data trace exception summary . . . 1431
Table 828. System Bus cycle special cases . . . 1433
Table 829. Watchpoint source description . . . 1434
Table 830. CRC unit memory map . . . 1436
Table 831. CRC_CFGn field descriptions . . . 1437
Table 832. CRC_INPn field descriptions . . . 1438
Table 833. CRC_CSTATn field descriptions . . . 1439
Table 834. CRC_OUTPn field descriptions . . . 1440
Table 835. MEMU memory map . . . 1451
Table 836. CTRL field descriptions . . . 1452
Table 837. ERR_FLAG field descriptions . . . 1453
Table 838. DEBUG field descriptions . . . 1456
Table 839. SYS_RAM_CERR_STSn field descriptions . . . 1458
Table 840. SYS_RAM_CERR_ADDRn field descriptions . . . 1458
Table 841. SYS_RAM_UNCERR_STS field descriptions . . . 1459
Table 842. SYS_RAM_UNCERR_ADDR field descriptions . . . 1459
Table 843. SYS_RAM_OFLWn field descriptions . . . 1460
Table 844. PERIPH_RAM_CERR_STSn field descriptions . . . 1461
Table 845. PERIPH_RAM_CERR_ADDRn field descriptions . . . 1461
Table 846. PERIPH_RAM_UNCERR_STS field descriptions . . . 1462
Table 847. PERIPH_RAM_UNCERR_ADDR field descriptions . . . 1462
Table 848. PERIPH_RAM_OFLWn field descriptions . . . 1463
Table 849. FLASH_CERR_STSn register field descriptions . . . 1464
Table 850. FLASH_CERR_ADDRn register field descriptions . . . 1464
Table 851. UNCERR_STS field descriptions . . . 1465
Table 852. FLASH_UNCERR_ADDR field descriptions . . . 1465
Table 853. FLASH_OFLWn field descriptions . . . 1466
Table 854. FCCU submodules . . . 1472
Table 855. Input reset . . . 1476
Table 856. Reset reactions . . . 1476
Table 857. Reset sources . . . 1476
Table 858. MC actions . . . 1476
Table 859. FCCU error pin behavior on STATE transitions . . . 1483
Table 860. Dual rail encoding . . . 1484
Table 861. Time switching encoding . . . 1485
Table 862. Bi-stable encoding . . . 1486
Table 863. FCCU memory map . . . 1490
Table 864. FCCU_CTRL field descriptions . . . 1494
Table 865. FCCU_CTRLK field descriptions . . . 1495
Table 866. FCCU_CFG field descriptions . . . 1496
Table 867. FCCU_RF_CFGn field descriptions . . . 1499
Table 868. FCCU RF configuration register channels . . . 1499
Table 869. FCCU_RFS_CFGn field descriptions . . . 1500
Table 870. FCCU RFS configuration register channels . . . 1501
Table 871. FCCU_RF_Sn field descriptions . . . 1502
Table 872. FCCU RF status register channels . . . 1503
Table 873. FCCU_RFK field descriptions . . . 1503
Table 874. FCCU_RF_En field descriptions . . . 1504
Table 875. FCCU RF_En enable register channels . . . 1504
Table 876. FCCU_RF_TOEn field descriptions . . . 1505
Table 877. FCCU RF timeout enable register channels . . . 1505
Table 878. FCCU_RF_TO field descriptions . . . 1506

Table 879. FCCU_CFG_TO field descriptions . . . 1507
Table 880. Bi-stable encoding . . . 1507
Table 881. FCCU_EINOUT field descriptions . . . 1508
Table 882. FCCU_STAT field descriptions . . . 1509
Table 883. N2AF_STATUS field descriptions . . . 1511
Table 884. A2FF_STATUS field descriptions . . . 1512
Table 885. N2FF_STATUS field descriptions . . . 1513
Table 886. F2AF_STATUS field descriptions . . . 1514
Table 887. FCCU_RFF field descriptions . . . 1514
Table 888. FCCU_IRQ_STAT field descriptions . . . 1515
Table 889. FCCU_IRQ_EN field descriptions . . . 1517
Table 890. FCCU_XTMR field descriptions . . . 1518
Table 891. Timer state/value . . . 1518
Table 892. FCCU_MCS field descriptions . . . 1519
Table 893. FCCU_TRANS_LOCK field descriptions . . . 1521
Table 894. FCCU_PERMNT_LOCK field descriptions . . . 1521
Table 895. FCCU_DELTA_T field descriptions . . . 1522
Table 896. FCCU_IRQ_ALARM_ENn field descriptions . . . 1523
Table 897. FCCU IRQ alarm enable register channels . . . 1523
Table 898. FCCU_NMI_ENn field descriptions . . . 1524
Table 899. FCCU NMI enable register channels . . . 1524
Table 900. FCCU_EOUT_SIG_ENn field descriptions . . . 1525
Table 901. FCCU EOUT signaling enable register channels . . . 1525
Table 902. STCU2 register set . . . 1536
Table 903. STCU_RUN field descriptions . . . 1538
Table 904. STCU_SKC field descriptions . . . 1539
Table 905. STCU_CFG field descriptions . . . 1541
Table 906. STCU_PLL_CFG field descriptions . . . 1543
Table 907. STCU_WDG field descriptions . . . 1544
Table 908. STCU_CRCE field descriptions . . . 1545
Table 909. STCU_CRCR field descriptions . . . 1545
Table 910. STCU_ERR_STAT field descriptions . . . 1546
Table 911. STCU_ERR_FM field descriptions . . . 1548
Table 912. STCU_LBS field descriptions . . . 1549
Table 913. STCU_LBE field descriptions . . . 1549
Table 914. STCU_LBUFM field descriptions . . . 1550
Table 915. STCU_MBSL field descriptions . . . 1551
Table 916. STCU_MBSM field descriptions . . . 1552
Table 917. STCU_MBSH field descriptions . . . 1552
Table 918. STCU_MBEL field descriptions . . . 1553
Table 919. STCU_MBEM field descriptions . . . 1554
Table 920. STCU_MBEH field descriptions . . . 1554
Table 921. STCU_MBUFML field descriptions . . . 1555
Table 922. STCU_MBUFMM field descriptions . . . 1556
Table 923. STCU_MBUFMH field descriptions . . . 1557
Table 924. STCU_LB_CTRLn field descriptions . . . 1558
Table 925. STCU_LB_PCSn field descriptions . . . 1560
Table 926. STCU_LB_PRPGLn field descriptions . . . 1561
Table 927. STCU_LB_PRPGHn field descriptions . . . 1561
Table 928. STCU_LB_MISRELn field descriptions . . . 1562
Table 929. STCU_LB_MISREHn field descriptions . . . 1563
Table 930. STCU_LB_MISRRLn field descriptions . . . 1563

Table 931. STCU_LB_MISRRHn field descriptions . . . 1564
Table 932. STCU_MB_CTRLn field descriptions . . . 1565
Table 933. STCU2 register map . . . 1566
Table 934. REG_PROT Memory Map . . . 1574
Table 935. Soft Lock Bits vs. Protected Address . . . 1575
Table 936. SLBRn field descriptions . . . 1576
Table 937. GCR field descriptions . . . 1577
Table 938. PASS memory map . . . 1584
Table 939. LCSTAT register field descriptions . . . 1586
Table 940. CHSEL register field descriptions . . . 1587
Table 941. CSTAT register field descriptions . . . 1587
Table 942. CINn register field descriptions . . . 1588
Table 943. LOCK0_PGn register field descriptions . . . 1589
Table 944. LOCK1_PGn register field descriptions . . . 1590
Table 945. LOCK2_PGn register field descriptions . . . 1591
Table 946. LOCK3_PGn register field descriptions . . . 1593
Table 947. Censorship register field descriptions . . . 1595
Table 948. Debug Interface Enable Truth Table . . . 1596
Table 949. Flash Memory Read Protection Truth Table . . . 1597
Table 950. CS/ADDR Values for Lock Registers . . . 1598
Table 951. Setting some bits in the MIDLOCK field . . . 1599
Table 952. Acronyms and abbreviations . . . 1600
Table 953. Document revision history . . . 1604

List of figures

Figure 1. Key to register fields . . . 75
Figure 2. Block diagram . . . 81
Figure 3. SPC570Sx flash memory block diagram . . . 83
Figure 4. Flash memory segmentation and read-while-write partitioning . . . 85
Figure 5. Replication architecture . . . 154
Figure 6. CPU to Flash write path protection . . . 157
Figure 7. CPU to Flash read path protection . . . 158
Figure 8. CPU to RAM write path protection . . . 159
Figure 9. CPU to RAM read path protection . . . 160
Figure 10. CPU to peripheral write path protection . . . 161
Figure 11. CPU to peripheral read path protection . . . 162
Figure 12. LBIST partitioning . . . 176
Figure 13. Crossbar switch integration . . . 186
Figure 14. SMPU_0 configuration . . . 188
Figure 15. Peripheral bridge configuration . . . 190
Figure 16. Interrupt scheme . . . 191
Figure 17. Interrupt timing . . . 201
Figure 18. DMA controller configuration . . . 201
Figure 19. DMA channel multiplexing . . . 202
Figure 20. PIT configuration . . . 209
Figure 21. DSPI configuration . . . 211
Figure 22. DSPI Module Configuration Register (DSPI*x*_MCR) . . . 212
Figure 23. MEMU system connections . . . 217
Figure 24. PASS_LOCK0_PGn Register . . . 233
Figure 25. PASS_LOCK1_PGn Register . . . 233
Figure 26. PASS_LOCK2_PGn Register . . . 234
Figure 27. PASS_LOCK3_PGn Register . . . 234
Figure 28. Reset sequence . . . 241
Figure 29. PORS Reset pin vs. Phase 3 Destructive . . . 244
Figure 30. PORS Reset pin vs. Phase 3 Functional . . . 244
Figure 31. PORST reset pin state during the reset sequence . . . 246
Figure 32. System boot sequence . . . 247
Figure 33. Valid boot header search . . . 249
Figure 34. DCF record structure . . . 252
Figure 35. DCF start record . . . 253
Figure 36. DCF stop record . . . 253
Figure 37. Appending DCF records . . . 254
Figure 38. UTEST Miscellaneous DCF Client . . . 262
Figure 39. PMC_REE bits DCF Client . . . 263
Figure 40. Mode relationships . . . 267
Figure 41. Power supply scheme . . . 268
Figure 42. Power segment and voltage monitor architecture . . . 269
Figure 43. LV supply range . . . 270
Figure 44. HV supply range . . . 270
Figure 45. LV supply monitors vs. voltage undershoot events . . . 273
Figure 46. HV supply monitors vs. voltage undershoot events . . . 273
Figure 47. Monitor checking timing diagram . . . 276
Figure 48. e200z0h block diagram . . . 282

Figure 49. e200z0h SUPERVISOR mode program model SPRs . . . 286
Figure 50. e200z0h USER mode program model . . . 287
Figure 51. SIUL2 block diagram . . . 289
Figure 52. SIUL2 MCU ID Register #1 (SIUL2_MIDR1) . . . 293
Figure 53. SIUL2 MCU ID Register #2 (SIUL2_MIDR2) . . . 295
Figure 54. SIUL2 DMA/Interrupt Status Flag Register (SIUL2_DISR0) . . . 296
Figure 55. SIUL2 DMA/Interrupt Request Enable Register 0 (SIUL2_DIRER0) . . . 297
Figure 56. SIUL2 DMA/Interrupt Request Select Register 0 (SIUL2_DIRSR0) . . . 298
Figure 57. SIUL2 Interrupt Rising-Edge Event Enable Register 0 (SIUL2_IREER0) . . . 298
Figure 58. SIUL2 Interrupt Falling-Edge Event Enable Register 0 (SIUL2_IFEER0) . . . 299
Figure 59. SIUL2 Interrupt Filter Enable Register 0 (SIUL2_IFER0) . . . 299
Figure 60. SIUL2 Interrupt Filter Maximum Counter Register (SIUL2_IFMCR0–SIUL2_IFMCR31) . 300
Figure 61. SIUL2 Interrupt Filter Clock Prescaler Register (SIUL2_IFCPR) . . . 301
Figure 62. MSCR I/O pin and IP block port connectivity . . . 302
Figure 63. I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR0–SIUL2_MSCR511) 303
Figure 64. Multiplexed Signal Configuration Registers for Multiplexed Input Selection (SIUL2_MSCR512–SIUL2_MSCR1023) . . . 306
Figure 65. SIUL2 GPIO Pad Data Out Register (SIUL2_GPDO0–SIUL2_GPDO79) . . . 307
Figure 66. SIUL2 GPIO Pad Data Input Registers (SIUL2_GPDI0–SIUL2_GPDI79) . . . 308
Figure 67. SIUL2 Parallel GPIO Pad Data Out Register (SIUL2_PGPDOn) . . . 308
Figure 68. SIUL2 Parallel GPIO Pad Data In Register (SIUL2_PGPDIn) . . . 309
Figure 69. SIUL2 Masked Parallel GPIO Pad Data Out Register (SIUL2_MPGPDOn) . . . 309
Figure 70. Data Port example arrangement showing configuration for different port width accesses 311
Figure 71. External interrupt pad diagram . . . 312
Figure 72. Interrupt to vector mapping at MCU level . . . 314
Figure 73. Priority Registers Slave (XBAR_PRS*n*) . . . 316
Figure 74. XBAR Control Register (XBAR_CRS*n*) . . . 318
Figure 75. Master General Purpose Control Registers (XBAR_MGPCRn) . . . 320
Figure 76. XBIC system block diagram . . . 324
Figure 77. XBIC block diagram . . . 325
Figure 78. XBIC Module Control Register (XBIC_MCR) . . . 326
Figure 79. XBIC Error Injection Register (XBIC_EIR) . . . 328
Figure 80. XBIC Error Status Register (XBIC_ESR) . . . 329
Figure 81. Error Address Register (XBIC_EAR) . . . 330
Figure 82. Master Privilege Register A (MPRA) . . . 337
Figure 83. MPROT*n* fields . . . 337
Figure 84. Peripheral Access Control Registers (PBRIDGE_PACRx) . . . 338
Figure 85. PACR*n* fields . . . 339
Figure 86. Off-Platform Peripheral Access Control Registers (PBRIDGE_OPACRx) . . . 342
Figure 87. OPACR*n* fields . . . 342
Figure 88. SMPU block diagram . . . 345
Figure 89. SMPU Control/Error Status Register (SMPU_CESR) . . . 348
Figure 90. SMPU Error Address Register, Slave Port n (SMPU_EARn) . . . 349
Figure 91. SMPU Error Detail Register, Slave Port n (SMPU_EDRn) . . . 349
Figure 92. SMPU Region Descriptor n, Word 0 Register (SMPU_RGDn_WORD0) . . . 351
Figure 93. SMPU Region Descriptor n – Word 1 Register (SMPU_RGDn_WORD1) . . . 351
Figure 94. SMPU Region Descriptor n – Word 2 Register (SMPU_RGDn_WORD2_FMT0) . . . 352
Figure 95. SMPU Region Descriptor n – Word 3 Register (SMPU_RGDn_WORD3) . . . 353
Figure 96. SMPU RGD Alternate Access Control n (SMPU_RGDAACn) . . . 354
Figure 97. SMPU access evaluation macro . . . 356
Figure 98. Access evaluation macro critical timing path . . . 359
Figure 99. INTC block diagram . . . 364

Figure 100. INTC Block Configuration Register (INTC_BCR) . . . 367
Figure 101. INTC Current Priority Register for Processor 0 (INTC_CPR0) . . . 368
Figure 102. INTC Interrupt Acknowledge Register for Processor 0 (INTC_IACKR0) . . . 368
Figure 103. INTC End of Interrupt Register for Processor 0 (INTC_EOIR0) . . . 369
Figure 104. INTC Software Set/Clear Interrupt Register n (INTC_SSCIRn) . . . 370
Figure 105. INTC Priority Select Register n (INTC_PSRn) . . . 371
Figure 106. INTC Priority Select Register n (INTC_PSRn) . . . 371
Figure 107. Timing diagram of software vector mode handshaking . . . 375
Figure 108. OSEK Code Sequence . . . 380
Figure 109. Interrupt request block diagram . . . 380
Figure 110. Timing diagram of raised priority preserved . . . 381
Figure 111. eDMA block diagram . . . 385
Figure 112. Control Register (DMA_CR) . . . 389
Figure 113. Error Status Register (DMA_ES) . . . 391
Figure 114. Enable Request Register (DMA_ERQL) . . . 393
Figure 115. Enable Error Interrupt Register (DMA_EEIL) . . . 395
Figure 116. Set Enable Request Register (DMA_SERQ) . . . 397
Figure 117. Clear Enable Request Register (DMA_CERQ) . . . 397
Figure 118. Set Enable Error Interrupt Register (DMA_SEEI) . . . 398
Figure 119. Clear Enable Error Interrupt Register (DMA_CEEI) . . . 398
Figure 120. Clear Interrupt Request Register (DMA_CINT) . . . 399
Figure 121. Clear Error Register (DMA_CERR) . . . 400
Figure 122. Set START Bit Register (DMA_SSRT) . . . 400
Figure 123. Clear DONE Status Bit Register (DMA_CDNE) . . . 401
Figure 124. Interrupt Request Register (DMA_INTL) . . . 402
Figure 125. Error Register (DMA_ERRL) . . . 404
Figure 126. Hardware Request Status Register (DMA_HRSL) . . . 405
Figure 127. Channel *n* Priority Register (DMA_DCHPRI*n*) . . . 407
Figure 128. Channel *n* Master ID Register (DMA_DCHMID*n*) . . . 408
Figure 129. TCD Source Address (DMA_TCD*n*_SADDR) . . . 408
Figure 130. TCD Transfer Attributes (DMA_TCD*n*_ATTR) . . . 409
Figure 131. TCD Signed Source Address Offset (DMA_TCD*n*_SOFF) . . . 410
Figure 132. TCD Minor Byte Count (Minor Loop Disabled) (DMA_TCD*n*_NBYTES_MLNO) . . . 410
Figure 133. TCD Signed Minor Loop Offset (Minor Loop Enabled and Offset Disabled) (DMA_TCD*n*_NBYTES_MLOFFNO) . . . 411
Figure 134. TCD Signed Minor Loop Offset (Minor Loop and Offset Enabled) (DMA_TCD*n*_NBYTES_MLOFFYES) . . . 412
Figure 135. TCD Last Source Address Adjustment (DMA_TCD*n*_SLAST) . . . 413
Figure 136. TCD Destination Address (DMA_TCD*n*_DADDR) . . . 414
Figure 137. TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD*n*_CITER_ELINKYES) . . . 414
Figure 138. TCD Current Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD*n*_CITER_ELINKNO) . . . 415
Figure 139. TCD Signed Destination Address Offset (DMA_TCD*n*_DOFF) . . . 416
Figure 140. TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD*n*_DLASTSGA) . . . 417
Figure 141. TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD*n*_BITER_ELINKYES) . . . 417
Figure 142. TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD*n*_BITER_ELINKNO) . . . 418
Figure 143. TCD Control and Status (DMA_TCD*n*_CSR) . . . 419
Figure 144. DMA operation, part 1 . . . 424

Figure 145. DMA operation, part 2. . . . 425
Figure 146. DMA operation, part 3. . . . 426
Figure 147. Example of multiple loop iterations 432
Figure 148. Memory array terms 433
Figure 149. DMACHMUX block diagram 443
Figure 150. Channel Configuration Registers (CHCFGn) 445
Figure 151. DMACHMUX triggered channels 447
Figure 152. DMACHMUX channel triggering: normal operation 447
Figure 153. DMACHMUX channel triggering: ignored trigger 448
Figure 154. Wakeup unit Block Diagram 454
Figure 155. NMI Status Flag Register (NSR). . . . 455
Figure 156. NMI Configuration Register (NCR) 457
Figure 157. TSENS block diagram 461
Figure 158. Clock generation architecture 466
Figure 159. SPC570Sx dual PLL digital interface block diagram 469
Figure 160. Clock distribution 473
Figure 161. CLOCKOUT pin muxing 474
Figure 162. CMU_0 block diagram 475
Figure 163. CMU[1:3] block diagram 475
Figure 164. Loss of clock logic. . . . 477
Figure 165. PLL0 Control Register (PLL0CR) 480
Figure 166. PLL0 Status Register (PLL0SR) 481
Figure 167. PLL0 Divider Register (PLL0DV) 482
Figure 168. PLL1 Control Register (PLL1CR) 483
Figure 169. PLL1 Status Register (PLL1SR) 484
Figure 170. PLL1 Divider Register (PLL1DV) 485
Figure 171. PLL1 Frequency Modulation Register (PLL1FM) 487
Figure 172. PLL1 Fractional Divider Register (PLL1FD) 488
Figure 173. Triangular frequency modulation 491
Figure 174. CMU Control Status Register (CMU_CSR) 496
Figure 175. CMU Frequency Display Register (CMU_FDR) 497
Figure 176. CMU High Frequency Reference Register CLKMN1 (CMU_HFREFR) 497
Figure 177. CMU Low Frequency Reference Register CLKMN1 (CMU_LFREFR). . . . 498
Figure 178. CMU Interrupt Status Register (CMU_ISR) 498
Figure 179. CMU Measurement Duration Register (CMU_MDR) 499
Figure 180. MC_CGM block diagram 503
Figure 181. PCS Switch Duration Register (CGM_PCS_SDUR). . . . 506
Figure 182. PCS Divider Change Register 1 (CGM_PCS_DIVC1) 506
Figure 183. PCS Divider Start Register 1 (CGM_PCS_DIVS1). . . . 507
Figure 184. PCS Divider End Register 1 (CGM_PCS_DIVE1) 508
Figure 185. PCS Divider Change Register 2 (CGM_PCS_DIVC2) 508
Figure 186. PCS Divider Start Register 2 (CGM_PCS_DIVS2). . . . 509
Figure 187. PCS Divider End Register 2 (CGM_PCS_DIVE2) 510
Figure 188. PCS Divider Change Register 4 (CGM_PCS_DIVC4) 510
Figure 189. PCS Divider Start Register 4 (CGM_PCS_DIVS4). . . . 511
Figure 190. PCS Divider End Register 4 (CGM_PCS_DIVE4) 512
Figure 191. CGM_SC_DIV_RC 513
Figure 192. CGM_DIV_UPD_TYPE 513
Figure 193. CGM_DIV_UPD_TRIG 514
Figure 194. CGM_DIV_UPD_STAT. . . . 515
Figure 195. System Clock Select Status Register (CGM_SC_SS) 516
Figure 196. System Clock Divider 0 Configuration Register (CGM_SC_DC0) 517

Figure 197. System Clock Divider 1 Configuration Register (CGM_SC_DC1) . . . 518
Figure 198. System Clock Divider 2 Configuration Register (CGM_SC_DC2) . . . 519
Figure 199. Auxiliary Clock 0 Select Control Register (CGM_AC0_SC) . . . 520
Figure 200. Auxiliary Clock 0 Select Status Register (CGM_AC0_SS) . . . 521
Figure 201. Auxiliary Clock 0 Divider 0 Configuration Register (CGM_AC0_DC0) . . . 522
Figure 202. Auxiliary Clock 0 Divider 1 Configuration Register (CGM_AC0_DC1) . . . 522
Figure 203. Auxiliary Clock 0 Divider 2 Configuration Register (CGM_AC0_DC2) . . . 523
Figure 204. Auxiliary Clock 0 Divider 3 Configuration Register (CGM_AC0_DC3) . . . 524
Figure 205. Auxiliary Clock 0 Divider 4 Configuration Register (CGM_AC0_DC4) . . . 524
Figure 206. Auxiliary Clock 0 Divider 5 Configuration Register (CGM_AC0_DC5) . . . 525
Figure 207. Auxiliary Clock 1 Select Control Register (CGM_AC1_SC) . . . 526
Figure 208. Auxiliary Clock 1 Select Status Register (CGM_AC1_SS) . . . 527
Figure 209. Auxiliary Clock 1 Divider 0 Configuration Register (CGM_AC1_DC0) . . . 528
Figure 210. Auxiliary Clock 2Select Control Register (CGM_AC2_SC) . . . 528
Figure 211. Auxiliary Clock 2 Select Status Register (CGM_AC2_SS) . . . 529
Figure 212. Auxiliary Clock 3 Select Control Register (CGM_AC3_SC) . . . 530
Figure 213. Auxiliary Clock 3Select Status Register (CGM_AC3_SS) . . . 531
Figure 214. MC_CGM system clock generation overview . . . 533
Figure 215. MC_CGM System Clock Ramp-Down Timing (k = 6 example) . . . 536
Figure 216. MC_CGM System Clock Ramp-Up Timing (k = 6 example) . . . 537
Figure 217. MC_CGM auxiliary clock 0 generation overview . . . 539
Figure 218. MC_CGM auxiliary clock 1 generation overview . . . 540
Figure 219. MC_CGM auxiliary clock 2 generation overview . . . 540
Figure 220. MC_CGM auxiliary clock 3 generation overview . . . 541
Figure 221. XOSC_CTL register . . . 545
Figure 222. IRCOSC control register (IRCOSC_CTL) . . . 548
Figure 223. IRCOSC Native Trimming register (IRCOSC_NT) . . . 549
Figure 224. Simplified platform block diagram . . . 550
Figure 225. Platform RAM configuration register 1 (PRCR1) . . . 552
Figure 226. E2EECC consideration for partial write - 16-bit write request . . . 556
Figure 227. E2EECC consideration for partial write - memory read from RAM . . . 556
Figure 228. E2EECC consideration for partial write - Separation of check-bits of targeted bit-lines . . . 557
Figure 229. E2EECC consideration for partial write - calculation of final check-bits . . . 557
Figure 230. RAM controller transaction monitor block diagram . . . 558
Figure 231. Platform-centric block diagram with flash memory controller . . . 560
Figure 232. Platform Flash Configuration Register 1 (PFCR1) . . . 561
Figure 233. Platform Flash Configuration Register 2 (PFCR2) . . . 564
Figure 234. Platform Flash Configuration Register 3 (PFCR3) . . . 567
Figure 235. Platform Flash Access Protection Register (PFAPR) . . . 569
Figure 236. Flash memory controller 4-entry, 4-way mini-cache organization . . . 575
Figure 237. Flash memory module structure . . . 580
Figure 238. Module Configuration Register (MCR) . . . 587
Figure 239. Extended Module Configuration Register (MCRE) . . . 594
Figure 240. Lock 0 register (LOCK0) . . . 597
Figure 241. Lock 1 register (LOCK1) . . . 599
Figure 242. Lock 2 Register (LOCK2) . . . 600
Figure 243. Select 0 register (SEL0) . . . 601
Figure 244. Select 1 register (SEL1) . . . 602
Figure 245. Select 2 register (SEL2) . . . 603
Figure 246. Address register (ADR) . . . 604
Figure 247. User Test 0 register (UT0) . . . 606
Figure 248. User Multiple Input Signature 0 register (UM0) . . . 610

Figure 249. User Multiple Input Signature 1 register (UM1). . . . 610
Figure 250. User Multiple Input Signature 2 register (UM2). . . . 611
Figure 251. User Multiple Input Signature 3 register (UM3). . . . 611
Figure 252. User Multiple Input Signature 8 register (UM8). . . . 612
Figure 253. User Multiple Input Signature 9 register (UM9). . . . 613
Figure 254. Over-program protection 0 register (OPP0) 613
Figure 255. Over-program protection 1 register (OPP1) 614
Figure 256. Over-program protection 2 register (OPP2) 615
Figure 257. Test Mode Disable Password Check register (TMD) 616
Figure 258. Doubleword program of data 0x55AA_55AA at address 0x00_AAA8 and data 0xAA55_AA55 at address 0x00_AAAC 623
Figure 259. Doubleword Program Suspend. . . . 624
Figure 260. Doubleword Program Resume 624
Figure 261. Erase of blocks B0F1 and B0F2 626
Figure 262. Block Erase Suspend 626
Figure 263. Block Erase Resume 627
Figure 264. Array Integrity Check of blocks B0F1 and B0F2 629
Figure 265. Margin Read Check versus 1’s. . . . 631
Figure 266. EIM Control Register (EIM_CTRL) 634
Figure 267. EIM Data Register (EIM_DATA) 635
Figure 268. EIM functional description. . . . 636
Figure 269. EIM replication with lockstep operation 637
Figure 270. ADC subsystem block diagram. . . . 640
Figure 271. SARADC clock architecture 643
Figure 272. SARADC input trigger diagram. . . . 643
Figure 273. Self Test implementation 645
Figure 274. SARADC block diagram 660
Figure 275. Normal conversion flow 662
Figure 276. Injected sample/conversion sequence 663
Figure 277. Analog conversion sequence 665
Figure 278. SARADC crosstrigger unit 667
Figure 279. CTU interface timings 668
Figure 280. Analog watchdog guarded area 670
Figure 281. Main Configuration Register (MCR) 675
Figure 282. Main Status Register (MSR) 678
Figure 283. Interrupt Status Register (ISR) 679
Figure 284. Internal Channel Interrupt Pending Registers (ICIPR0–ICIPR2) 681
Figure 285. Interrupt Mask Register (IMR) 682
Figure 286. Internal Channel Interrupt Mask Registers (ICIMR0–ICIMR2) 683
Figure 287. Watchdog Threshold Interrupt Status Register (WTISR) 684
Figure 288. Watchdog Threshold Interrupt Mask Register (WTIMR). . . . 685
Figure 289. DMA Enable Register (DMAE) 686
Figure 290. Internal Channel DMA Select Register (ICDSR0–ICDSR2) 686
Figure 291. Watchdog Threshold Registers 0–3 (WTHRHLR0–WTHRHLR3) 687
Figure 292. Conversion Timing Registers 0–3 (CTR0–CTRL3). . . . 688
Figure 293. Internal Channel Normal Conversion Mask Registers (ICNCMR0–ICNCMR2) 689
Figure 294. Internal Channel Injected Conversion Mask Registers (ICJCMR0–ICJCMR2) 690
Figure 295. Power Down Exit Delay Register (PDEDR) 691
Figure 296. Internal Channel Data Registers (ICDR0–ICDR95) 691
Figure 297. Internal Channel Watchdog Select Registers (ICWSELR0–ICWSELR11). . . . 693
Figure 298. Internal Channel Watchdog Enable Registers (ICWENR0–ICWENR2) 694
Figure 299. Internal Channel Analog Watchdog Out of Range Registers (ICAWORR0–ICAWORR2) 695

Figure 300. Test Channel Interrupt Pending Register (TCIPR) . . . 696
Figure 301. Test Channel Interrupt Mask Register (TCIMR) . . . 697
Figure 302. Test Channel DMA Select Register (TCDSR) . . . 698
Figure 303. Test Channel Normal Conversion Mask Register (TCNCMR) . . . 699
Figure 304. Test Channel Injected Conversion Mask Register (TCJCMR) . . . 699
Figure 305. Test Channels Watchdog Select Registers 0–3 (TCWSELR0–TCWSELR3) . . . 700
Figure 306. Test Channel Watchdog Enable Register (TCWENR) . . . 701
Figure 307. Test Channel Analog Watchdog Out of Range Register (TCAWORR) . . . 702
Figure 308. Test Channel Connection with Analog Pin Registers 0–7 (TCCAPR0–TCCAPR7) . . . 703
Figure 309. Test Channel Data Registers (TCDR96–TCDR127) . . . 705
Figure 310. Cross triggering unit diagram . . . 711
Figure 311. TGS in triggered mode . . . 712
Figure 312. Example timing for TGS in triggered mode . . . 713
Figure 313. TGS in sequential mode . . . 714
Figure 314. Example timing for TGS in sequential mode . . . 714
Figure 315. TGS counter cases . . . 715
Figure 316. Scheduler subunit . . . 717
Figure 317. Reload error scenario . . . 720
Figure 318. Trigger Generator Sub-unit Input Selection Register (TGSISR) . . . 727
Figure 319. Trigger Generator Sub-unit Control Register (TGSCR) . . . 729
Figure 320. Trigger n Compare Register (TnCR) . . . 730
Figure 321. TGS Counter Compare Register (TGSCCR) . . . 730
Figure 322. TGS Counter Reload Register (TGSCRR) . . . 731
Figure 323. Commands list control register 1 (CLCR1) . . . 731
Figure 324. Commands list control register 2 (CLCR2) . . . 732
Figure 325. Trigger handler control register 1 (THCR1) . . . 732
Figure 326. Trigger handler control register 2 (THCR2) . . . 734
Figure 327. Commands list register *n* (CLR*n*) (CMS = 0) . . . 736
Figure 328. Commands list register *n* (CLRn) (CMS = 1) . . . 737
Figure 329. FIFO DMA control register (FDCR) . . . 737
Figure 330. FIFO control register (FCR) . . . 738
Figure 331. FIFO threshold register (FTH) . . . 740
Figure 332. FIFO status register (FST) . . . 740
Figure 333. FIFO Right aligned data *n* (FR*n*) . . . 742
Figure 334. FIFO signed Left aligned data *n* (FL*n*) . . . 743
Figure 335. Cross triggering unit error flag register (CTUEFR) . . . 743
Figure 336. Cross triggering unit interrupt flag register (CTUIFR) . . . 744
Figure 337. Cross triggering unit interrupt/DMA register (CTUIR) . . . 745
Figure 338. Control ON time register (COTR) . . . 746
Figure 339. Cross triggering unit control register (CTUCR) . . . 747
Figure 340. Cross triggering unit digital filter (CTUDF) . . . 748
Figure 341. eTimer block diagram . . . 751
Figure 342. eTimer channel block diagram . . . 752
Figure 343. Quadrature Incremental Position Encoder . . . 754
Figure 344. Triggered Count Mode (Length=1) . . . 755
Figure 345. One-Shot Mode (Length=1) . . . 755
Figure 346. Pulse Output Mode . . . 756
Figure 347. Variable PWM Waveform . . . 757
Figure 348. Variable Frequency PWM Mode Timing . . . 758
Figure 349. Compare Register and OFLAG Timing . . . 759
Figure 350. OFLAG timing during non-continuous counting . . . 760
Figure 351. OFLAG timing during continuous counting . . . 761

Figure 352. Compare Register 1n (COMP1n) . . . 764
Figure 353. Compare Register 2n (COMP2n) . . . 764
Figure 354. Capture Register 1n (CAPT1n) . . . 764
Figure 355. Capture Register 2n (CAPT2n) . . . 765
Figure 356. Load Register n (LOADn) . . . 765
Figure 357. Hold Register n (HOLDn) . . . 766
Figure 358. Counter Register n (CNTRn) . . . 766
Figure 359. Control Register 1n (CTRL1n) . . . 766
Figure 360. Control Register 2n (CTRL2n) . . . 769
Figure 361. Control Register 3n (CTRL3n) . . . 771
Figure 362. Status Register n (STSn) . . . 772
Figure 363. Interrupt and DMA Enable Register n (INTDMAn) . . . 774
Figure 364. Comparator Load 1n (CMPLD1n) . . . 775
Figure 365. Comparator Load 2n (CMPLD2n) . . . 775
Figure 366. Compare and Capture Control Register n (CCCTRLn) . . . 776
Figure 367. Input Filter Register n (FILTn) . . . 778
Figure 368. Watchdog Time-out Low Word Register (WDTOL) . . . 779
Figure 369. Watchdog Time-out High Word Register (WDTOH) . . . 780
Figure 370. Fault Control Register (FCTRL) . . . 780
Figure 371. Fault Status Register (FSTS) . . . 781
Figure 372. Fault Filter Register (FFILT) . . . 781
Figure 373. Channel Enable Register (ENBL) . . . 782
Figure 374. DMA Request 0 Select Register (DREQ0) . . . 783
Figure 375. DMA Request 1 Select Register (DREQ1) . . . 785
Figure 376. STM Control Register (STM_CR) . . . 790
Figure 377. STM Count Register (STM_CNT) . . . 791
Figure 378. STM Channel Control Register (STM_CCRn) . . . 792
Figure 379. STM Channel Interrupt Register (STM_CIRn) . . . 792
Figure 380. STM Channel Compare Register (STM_CMPn) . . . 793
Figure 381. SWT Control Register (SWT_CR) . . . 795
Figure 382. SWT Interrupt Register (SWT_IR) . . . 797
Figure 383. SWT Time-Out Register (SWT_TO) . . . 797
Figure 384. SWT Window Register (SWT_WN) . . . 798
Figure 385. SWT Service Register (SWT_SR) . . . 798
Figure 386. SWT Counter Output Register (SWT_CO) . . . 799
Figure 387. SWT Service Key Register (SWT_SK) . . . 799
Figure 388. Block diagram of the PIT . . . 803
Figure 389. PIT Module Control Register (MCR) . . . 805
Figure 390. PIT Timer Load Value Register n (LDVALn) . . . 805
Figure 391. PIT Current Timer Value Register n (CVALn) . . . 806
Figure 392. Timer Control Register (TCTRLn) . . . 806
Figure 393. Timer Flag Register n (TFLGn) . . . 807
Figure 394. Stopping and starting a timer . . . 808
Figure 395. Modifying running timer period . . . 808
Figure 396. Dynamically setting a new load value . . . 808
Figure 397. FlexCAN block diagram . . . 811
Figure 398. Message buffer structure . . . 817
Figure 399. Rx FIFO structure . . . 821
Figure 400. ID Table 0–7 . . . 822
Figure 401. Module Configuration Register (MCR) . . . 823
Figure 402. Control Register (CTRL) . . . 827
Figure 403. Free Running Timer (TIMER) . . . 830

Figure 404. Rx Global Mask register (RXGMASK) . . . 831
Figure 405. Rx Buffer 14 Mask register (RX14MASK) . . . 832
Figure 406. Rx Buffer 15 Mask register (RX15MASK) . . . 832
Figure 407. Error Counter Register (ECR) . . . 834
Figure 408. Error and Status Register (ESR) . . . 835
Figure 409. Interrupt Masks 1 Register (IMASK1) . . . 837
Figure 410. Interrupt Flags 1 Register (IFLAG1) . . . 838
Figure 411. Rx Individual Mask Registers (RXIMRn) . . . 840
Figure 412. CAN engine clocking scheme . . . 849
Figure 413. Segments within the bit time . . . 851
Figure 414. Arbitration, match, and move time windows . . . 852
Figure 415. DSPI block diagram (1) . . . 857
Figure 416. DSPI with queues and DMA . . . 860
Figure 417. DSPI Module Configuration Register (DSPI*x*_MCR) . . . 864
Figure 418. DSPI Transfer Count Register (DSPIx_TCR) . . . 868
Figure 419. DSPI Clock and Transfer Attributes Register (In Master Mode) (DSPI*x*_CTAR*n*) . . . 869
Figure 420. DSPI Clock and Transfer Attributes Register (In Slave Mode) (DSPI*x*_CTAR0_SLAVE) 874
Figure 421. DSPI Clock and Transfer Attributes Register (In Slave Mode) (DSPI*x*_CTAR1_SLAVE) 875
Figure 422. DSPI Status Register (DSPI*x*_SR) . . . 876
Figure 423. DSPI DMA/Interrupt Request Select and Enable Register (DSPI*x*_RSER) . . . 879
Figure 424. DSPI PUSH FIFO Register In Master Mode (DSPI*x*_PUSHR) . . . 881
Figure 425. DSPI PUSH FIFO Register In Slave Mode (DSPI*x*_PUSHR_SLAVE) . . . 884
Figure 426. DSPI POP FIFO Register (DSPI*x*_POPR) . . . 885
Figure 427. DSPI Transmit FIFO Registers (DSPI*x*_TXFRn) . . . 885
Figure 428. DSPI Receive FIFO Registers (DSPI*x*_RXFRn) . . . 886
Figure 429. DSPI Clock and Transfer Attributes Register Extended (DSPI*x*_CTARE*n*) . . . 887
Figure 430. DSPI Status Register Extended (DSPI*x*_SREX) . . . 887
Figure 431. SPI serial protocol overview . . . 889
Figure 432. Communications clock prescalers and scalers . . . 893
Figure 433. Peripheral Chip Select Strobe Timing . . . 895
Figure 434. DSPI transfer timing diagram (MTFE = 0, CPHA = 0, FMSZ = 8) . . . 897
Figure 435. DSPI transfer timing diagram (MTFE = 0, CPHA = 1, FMSZ = 8) . . . 898
Figure 436. DSPI Modified Transfer Format (MTFE = 1, CPHA = 0, $f_{sck} = f_p/4$) . . . 900
Figure 437. DSPI Modified Transfer Format (MTFE = 1, CPHA = 0, $f_{sck} = f_p/2$) . . . 900
Figure 438. DSPI Modified Transfer Format (MTFE = 1, CPHA = 0, $f_{sck} = f_p/3$) . . . 901
Figure 439. DSPI Modified Transfer Format (MTFE = 1, CPHA = 1, $f_{sck} = f_p/2$) . . . 901
Figure 440. DSPI Modified Transfer Format (MTFE = 1, CPHA = 1, $f_{sck} = f_p/3$) . . . 902
Figure 441. DSPI Modified Transfer Format (MTFE = 1, CPHA = 1, $f_{sck} = f_p/4$) . . . 902
Figure 442. Example of non-continuous format (CPHA = 1, CONT = 0) . . . 903
Figure 443. Example of continuous transfer (CPHA = 1, CONT = 1) . . . 903
Figure 444. Example of Fast Continuous Selection Format . . . 905
Figure 445. Continuous SCK timing diagram (CONT = 0) . . . 906
Figure 446. Continuous SCK Timing Diagram (CONT = 1) . . . 906
Figure 447. TX FIFO pointers and counter . . . 915
Figure 448. Block diagram . . . 918
Figure 449. Frames . . . 921
Figure 450. Structure of LIN frame . . . 921
Figure 451. Structure of byte field . . . 921
Figure 452. Break field . . . 922
Figure 453. Sync byte field . . . 922
Figure 454. Identifier field . . . 922
Figure 455. Operating modes . . . 923

Figure 456. Loop Back test mode . . . 924
Figure 457. Incomplete response (for example, missing checksum) . . . 928
Figure 458. No response . . . 929
Figure 459. Identifier List mode . . . 932
Figure 460. Identifier Mask mode . . . 932
Figure 461. Start detection and sampling for over sampling rate = 4 (Case 1) . . . 935
Figure 462. Start detection and sampling for over sampling rate = 4 (Case 2a) . . . 935
Figure 463. Start detection and sampling for over sampling rate = 4 (Case 2b) . . . 936
Figure 464. Start detection and sampling for over sampling rate = 4 (Case 3) . . . 936
Figure 465. Start detection and sampling for over sampling rate = 4 (Case 4) . . . 936
Figure 466. Start detection and sampling for over sampling rate = 4 (Case 5) . . . 937
Figure 467. Wakeup sequence . . . 938
Figure 468. UART mode 8-bit data frame . . . 939
Figure 469. UART mode 9-bit data frame . . . 939
Figure 470. UART mode 16-bit data frame . . . 940
Figure 471. UART mode 17-bit data frame . . . 940
Figure 472. UART mode 13-bit data frame . . . 940
Figure 473. Structure of 8-byte buffer . . . 941
Figure 474. Master node: TX memory map . . . 947
Figure 475. Master node: DMA Tx FSM (concept scheme) . . . 949
Figure 476. Master node: RX memory map . . . 950
Figure 477. Master node — DMA Rx FSM (concept scheme) . . . 951
Figure 478. Slave node — TX memory map . . . 952
Figure 479. Slave node: DMA Tx FSM (concept scheme) . . . 953
Figure 480. Slave node: RX memory map . . . 954
Figure 481. Slave node — DMA Rx FSM (concept scheme) . . . 956
Figure 482. UART: TX memory map . . . 956
Figure 483. UART: DMA Tx FSM (concept scheme) . . . 958
Figure 484. UART: RX memory map . . . 958
Figure 485. UART — DMA Rx FSM (concept scheme) . . . 960
Figure 486. DMA interface (major loop — single iteration) . . . 961
Figure 487. DMA interface (major loop — N iteration) . . . 961
Figure 488. LIN Control Register 1 (LINCR1) . . . 964
Figure 489. LIN Interrupt enable register (LINIER) . . . 966
Figure 490. LIN Status Register (LINSR) . . . 969
Figure 491. LIN error status register (LINESR) . . . 972
Figure 492. UART Mode Control Register (UARTCR) . . . 973
Figure 493. UART Mode Status Register (UARTSR) . . . 977
Figure 494. LIN Time-Out Control Status Register (LINTCSR) . . . 980
Figure 495. LIN Output Compare Register (LINOCR) . . . 981
Figure 496. LIN Time-Out Control Register (LINTOCR) . . . 982
Figure 497. LIN Fractional Baud Rate Register (LINFBRR) . . . 983
Figure 498. LIN Integer Baud Rate Register (LINIBRR) . . . 984
Figure 499. LIN Checksum Field Register (LINCFR) . . . 984
Figure 500. LIN Control Register 2 (LINCR2) . . . 985
Figure 501. Buffer Identifier Register (BIDR) . . . 987
Figure 502. Buffer Data Register Least Significant (BDRL) . . . 988
Figure 503. Buffer Data Register Most Significant (BDRM) . . . 988
Figure 504. Identifier Filter Enable Register (IFER) . . . 989
Figure 505. Identifier Filter Match Index (IFMI) . . . 990
Figure 506. Identifier Filter Mode Register (IFMR) . . . 990
Figure 507. Identifier Filter Control Register (IFCR2n) . . . 991

Figure 508. Identifier Filter Control Register (IFCR2n+1) . . . 992
Figure 509. Global Control Register (GCR) . . . 993
Figure 510. UART Preset Timeout Register (UARTPTO) . . . 994
Figure 511. UART Current Timeout Register (UARTCTO) . . . 995
Figure 512. DMA Tx Enable Register (DMATXE) . . . 996
Figure 513. DMA Rx Enable Register (DMARXE) . . . 997
Figure 514. Master node: transmitter configuration . . . 997
Figure 515. Master node: receiver configuration . . . 998
Figure 516. Master node: transmitter, bit error configuration . . . 998
Figure 517. Master node: transmitter, checksum error configuration . . . 998
Figure 518. Master node: receiver, checksum error configuration . . . 998
Figure 519. Slave node: transmitter (no identifier filters) configuration . . . 999
Figure 520. Slave node: receiver (no identifier filters) configuration . . . 999
Figure 521. Slave node: receiver (no identifier filters) configuration . . . 999
Figure 522. Slave node: No filters, transmitter, bit error configuration . . . 999
Figure 523. Slave node: No filters, receiver, checksum error configuration . . . 1000
Figure 524. Slave node: one TX filter configuration . . . 1000
Figure 525. Slave node: one RX filter configuration . . . 1000
Figure 526. RX only, TX only, RX & TX filters configuration . . . 1000
Figure 527. TX filter, BF is set configuration: ID has matched . . . 1001
Figure 528. TX filter, BF is set configuration: ID not matched . . . 1001
Figure 529. RX filter, BF is set configuration: ID matched . . . 1001
Figure 530. RX filter, BF is set configuration: ID not matched (ID is Rx) . . . 1002
Figure 531. RX filter, BF is set configuration: (ID is Tx) . . . 1002
Figure 532. TX filter, RX filter, BF set configuration: (ID matched) . . . 1002
Figure 533. TX filter, RX filter, BF set configuration: (ID matched + 1) . . . 1002
Figure 534. TX filter, RX filter, BF set configuration: (ID not matched, RX/TX Interrupt) . . . 1003
Figure 535. Extended frames (TX Interrupt) . . . 1003
Figure 536. Extended frames (RX Interrupt) . . . 1003
Figure 537. Response timeout . . . 1003
Figure 538. Header timeout . . . 1004
Figure 539. UART mode configuration . . . 1004
Figure 540. Interrupt diagram . . . 1005
Figure 541. MC_RGM block diagram . . . 1007
Figure 542. Destructive Event Status Register (RGM_DES) . . . 1011
Figure 543. Destructive Event Reset Disable Register (RGM_DERD) . . . 1013
Figure 544. Destructive Event Alternate Request Register (RGM_DEAR) . . . 1014
Figure 545. Destructive Bidirectional Reset Enable Register (RGM_DBRE) . . . 1015
Figure 546. Functional Event Status Register (RGM_FES) . . . 1017
Figure 547. Functional Event Reset Disable Register (RGM_FERD) . . . 1019
Figure 548. Functional Event Alternate Request Register (RGM_FEAR) . . . 1020
Figure 549. Functional Bidirectional Reset Enable Register (RGM_FBRE) . . . 1021
Figure 550. Functional Event Short Sequence Register (RGM_FESS) . . . 1023
Figure 551. Functional Reset Escalation Threshold Register (RGM_FRET) . . . 1024
Figure 552. Destructive Reset Escalation Threshold Register (RGM_DRET) . . . 1025
Figure 553. Peripheral Reset Register 0 (RGM_PRST0) . . . 1027
Figure 554. Peripheral Reset Register 1 (RGM_PRST1) . . . 1028
Figure 555. Peripheral Reset Register 2 (RGM_PRST2) . . . 1029
Figure 556. Peripheral Reset Register 3 (RGM_PRST3) . . . 1030
Figure 557. Peripheral Reset Register 4 (RGM_PRST4) . . . 1031
Figure 558. Peripheral Reset Register 5 (RGM_PRST5) . . . 1032
Figure 559. Peripheral Reset Register 6 (RGM_PRST6) . . . 1033

Figure 560. Peripheral Reset Register 7 (RGM_PRST7) . . . 1034
Figure 561. Peripheral Reset Status Register 0 (RGM_PSTAT0) . . . 1035
Figure 562. Peripheral Reset Status Register 1 (RGM_PSTAT1) . . . 1036
Figure 563. Peripheral Reset Status Register 2 (RGM_PSTAT2) . . . 1037
Figure 564. Peripheral Reset Status Register 3 (RGM_PSTAT3) . . . 1038
Figure 565. Peripheral Reset Status Register 4 (RGM_PSTAT4) . . . 1039
Figure 566. Peripheral Reset Status Register 5 (RGM_PSTAT5) . . . 1040
Figure 567. Peripheral Reset Status Register 6 (RGM_PSTAT6) . . . 1041
Figure 568. Peripheral Reset Status Register 7 (RGM_PSTAT7) . . . 1042
Figure 569. MC_RGM State Machine . . . 1044
Figure 570. Flow of control . . . 1052
Figure 571. BAF DCF ADDR client configuration . . . 1054
Figure 572. DATA32 DCF client configuration . . . 1055
Figure 573. DATA16 DCF Client Configuration . . . 1055
Figure 574. DATA8 DCF Client Configuration . . . 1056
Figure 575. Flow of control during serial boot . . . 1058
Figure 576. Start address . . . 1061
Figure 577. VLE bit, NO_ECHO bit and code size . . . 1061
Figure 578. LINFlexD bit timing in UART mode . . . 1062
Figure 579. FlexCAN Bit Timing . . . 1063
Figure 580. SSCM overview . . . 1066
Figure 581. System Status Register (STATUS) . . . 1068
Figure 582. System memory and ID register (MEMCONFIG) . . . 1069
Figure 583. Error Configuration Register (ERROR) . . . 1069
Figure 584. Processor Start Address Register (PSA) . . . 1070
Figure 585. Life Cycle Status Register (LCSTAT) . . . 1071
Figure 586. DCF start record . . . 1072
Figure 587. Format of a DCF data record . . . 1072
Figure 588. DCF stop record . . . 1072
Figure 589. Appending DCF records . . . 1073
Figure 590. Flash memory partitioning and RCHW search . . . 1075
Figure 591. Reset Configuration Half-Word . . . 1075
Figure 592. Digital PMU block diagram . . . 1079
Figure 593. Event Pending Register (*EPR_LV0*) . . . 1083
Figure 594. Reset Event Enable Register (*REE_LV0*) . . . 1084
Figure 595. Reset Event Select Register (*RES_LV*0) . . . 1084
Figure 596. Interrupt Enable Register (*IE_LV0*) . . . 1085
Figure 597. FCCU Event Enable Register (*FEE_LV0*) . . . 1086
Figure 598. Event Pending Register (*EPR_LV1*) . . . 1086
Figure 599. Reset Event Enable Register (*REE_LV1*) . . . 1087
Figure 600. Reset Event Select Register (*RES_LV1*) . . . 1088
Figure 601. Interrupt Enable Register (*IE_LV1*) . . . 1088
Figure 602. FCCU Event Enable Register (*FEE_LV1*) . . . 1089
Figure 603. Event Pending Register (*EPR_HV0*) . . . 1090
Figure 604. Reset Event Enable Register (*REE_HV0*) . . . 1091
Figure 605. Reset Event Select Register (*RES_HV*0) . . . 1092
Figure 606. Interrupt Enable Register (*IE_HV0*) . . . 1093
Figure 607. FCCU Event Enable Register (*FEE_HV0*) . . . 1094
Figure 608. Event Pending Register (*EPR_HV1*) . . . 1095
Figure 609. Reset Event Enable Register (*REE_HV1*) . . . 1096
Figure 610. Reset Event Select Register (*RES_HV1*) . . . 1097
Figure 611. Interrupt Enable Register (*IE_HV1*) . . . 1098

Figure 612. FCCU Event Enable Register (*FEE_HV1*) . . . 1099
Figure 613. Supply Gauge Status Register (*GR_S*) . . . 1100
Figure 614. Pending Gauge Status Register (*GR_P*) . . . 1103
Figure 615. Interrupt Enable Global Register (*IE_G*) . . . 1105
Figure 616. Main Regulator Control Register (MREG_CTRL) . . . 1106
Figure 617. Event Pending Register (*EPR_TD*) . . . 1107
Figure 618. Reset Event Enable Register (*REE_TD*) . . . 1109
Figure 619. Reset Event Select Register (*RES_TD*) . . . 1110
Figure 620. Temperature Sensor Configuration Register (CTL_TD) . . . 1111
Figure 621. Temperature Sensor FCCU Event Enable Register (*FEE_TD*) . . . 1112
Figure 622. ESR0 ConFiGuration Register (ESR0_CFG) . . . 1113
Figure 623. User BIST Flags Register (BIST_FLAGS) . . . 1113
Figure 624. User BIST Control Register (BIST_CTRL) . . . 1115
Figure 625. User BIST Time1 and Time0 Register (BIST_TIME10) . . . 1116
Figure 626. User BIST Time3 and Time2 Register (BIST_TIME32) . . . 1116
Figure 627. User BIST Time6 and Time5 Register (BIST_TIME65) . . . 1117
Figure 628. User BIST VD Under Test Monitor Register (BIST_DEBUG) . . . 1118
Figure 629. MC_PCU block diagram . . . 1121
Figure 630. Power Domain Status Register (PCU_PSTAT) . . . 1123
Figure 631. MC_ME block diagram . . . 1125
Figure 632. Global Status Register (ME_GS) . . . 1131
Figure 633. Mode Control Register (ME_MCTL) . . . 1133
Figure 634. Mode Enable Register (ME_ME) . . . 1134
Figure 635. Interrupt Status Register (ME_IS) . . . 1136
Figure 636. Interrupt Mask Register (ME_IM) . . . 1137
Figure 637. Invalid Mode Transition Status Register (ME_IMTS) . . . 1138
Figure 638. Debug Mode Transition Status Register (ME_DMTS) . . . 1139
Figure 639. RESET Mode Configuration Register (ME_RESET_MC) . . . 1143
Figure 640. TEST Mode Configuration Register (ME_TEST_MC) . . . 1145
Figure 641. SAFE Mode Configuration Register (ME_SAFE_MC) . . . 1147
Figure 642. DRUN Mode Configuration Register (ME_DRUN_MC) . . . 1149
Figure 643. RUNn Mode Configuration Register (ME_RUNn_MC) . . . 1151
Figure 644. HALT0 Mode Configuration Register (ME_HALT0_MC) . . . 1153
Figure 645. STOP0 Mode Configuration Register (ME_STOP0_MC) . . . 1155
Figure 646. Peripheral Status Register 0 (ME_PS0) . . . 1157
Figure 647. Peripheral Status Register 1 (ME_PS1) . . . 1158
Figure 648. Peripheral Status Register 2 (ME_PS2) . . . 1158
Figure 649. Peripheral Status Register 3 (ME_PS3) . . . 1159
Figure 650. Peripheral Status Register 4 (ME_PS4) . . . 1159
Figure 651. Peripheral Status Register 6 (ME_PS6) . . . 1160
Figure 652. Peripheral Status Register 7 (ME_PS7) . . . 1160
Figure 653. Run Peripheral Configuration Registers n (ME_RUN_PCn) . . . 1161
Figure 654. Low-Power Peripheral Configuration Register n (ME_LP_PCn) . . . 1162
Figure 655. Peripheral Control Register n (ME_PCTLn) . . . 1163
Figure 656. Core Status Register (ME_CS) . . . 1164
Figure 657. Core Control Register (ME_CCTL0) . . . 1165
Figure 658. Core Address Register 0 (ME_CADDR0) . . . 1166
Figure 659. MC_ME mode diagram . . . 1168
Figure 660. MC_ME transition diagram . . . 1179
Figure 661. MC_ME application example flow diagram . . . 1183
Figure 662. e200z0h debug resources . . . 1186
Figure 663. DBCR0 register . . . 1192

Figure 664. DBCR1 register . . . 1194
Figure 665. DBCR2 register . . . 1196
Figure 666. DBSR register . . . 1197
Figure 667. OnCE TAP controller and registers . . . 1201
Figure 668. OnCE controller implementation . . . 1202
Figure 669. e200z0h OnCE controller and serial interface . . . 1206
Figure 670. OnCE Status register . . . 1206
Figure 671. OnCE Command register . . . 1208
Figure 672. OnCE Control register . . . 1210
Figure 673. CPU Scan Chain Register (CPUSCR) . . . 1216
Figure 674. Control State Register (CTL) . . . 1217
Figure 675. Debug architecture . . . 1225
Figure 676. DCI block diagram . . . 1226
Figure 677. DCI control register (DCI_CR) . . . 1227
Figure 678. DCI pin multiplexing control register (DCI_PINCR) . . . 1228
Figure 679. DCI debug control mechanism . . . 1245
Figure 680. DCI debug signal generation . . . 1246
Figure 681. DCI EVTO management . . . 1247
Figure 682. DCI EVTO timing diagram . . . 1247
Figure 683. JTAG (IEEE 1149.1) block diagram . . . 1252
Figure 684. 6-bit instruction register . . . 1255
Figure 685. Device identification register . . . 1256
Figure 686. JTAG_PASSWORD register . . . 1257
Figure 687. Shifting data through a register . . . 1258
Figure 688. IEEE 1149.1-2001 TAP controller finite state machine . . . 1259
Figure 689. JDC block diagram . . . 1265
Figure 690. Module Configuration Register (JDC_MCR) . . . 1266
Figure 691. Module Status Register (JDC_MSR) . . . 1267
Figure 692. JDC_JOUT_IPS register . . . 1267
Figure 693. JDC_JIN_IPS register . . . 1268
Figure 694. JOUT JTAG register . . . 1268
Figure 695. JIN JTAG Register . . . 1269
Figure 696. JTAGM block diagram . . . 1272
Figure 697. JTAGM to DCI serial interface diagram . . . 1274
Figure 698. JTAGM_MCR register . . . 1275
Figure 699. JTAGM_SR register . . . 1277
Figure 700. JTAGM_DOR0 register . . . 1279
Figure 701. JTAGM_DOR1 register . . . 1279
Figure 702. JTAGM_DOR2 register . . . 1279
Figure 703. JTAGM_DOR3 register . . . 1280
Figure 704. JTAGM_DIR0 register . . . 1280
Figure 705. JTAGM_DIR1 register . . . 1281
Figure 706. Nexus port controller block diagram . . . 1283
Figure 707. 4-bit Instruction Register . . . 1287
Figure 708. Nexus Device ID Register . . . 1287
Figure 709. Port Configuration Register (PCR) . . . 1288
Figure 710. MSEO transfers (for 1-bit MSEO) . . . 1291
Figure 711. Message field sizes . . . 1292
Figure 712. Transmission sequence of messages . . . 1293
Figure 713. Shifting data into register . . . 1293
Figure 714. IEEE 1149.1-2001 TAP controller state machine . . . 1294
Figure 715. NEXUS controller state machine . . . 1295

Figure 716. IEEE 1149.1 controller command input . . . 1296
Figure 717. SPU block diagram . . . 1299
Figure 718. SPU module interface . . . 1304
Figure 719. Level1 Mux Selection 0 (L1SEL0) . . . 1312
Figure 720. Level1 Mux Selection 1 (L1SEL1) . . . 1312
Figure 721. Level1 Mux Selection 2 (L1SEL2) . . . 1313
Figure 722. Level1 Mux Selection 3 (L1SEL3) . . . 1313
Figure 723. Level1 Mux Selection 4 (L1SEL4) . . . 1314
Figure 724. Level1 Mux Selection 5 (L1SEL5) . . . 1314
Figure 725. Level1 Mux Selection 6 (L1SEL6) . . . 1315
Figure 726. Level1 Mux Selection 7 (L1SEL7) . . . 1315
Figure 727. Input triggers level detection control 2 (ITLD2) . . . 1316
Figure 728. Input triggers level detection control 3 (ITLD3) . . . 1318
Figure 729. CPU0 processor exception vector prefix (C0PEVP) . . . 1320
Figure 730. CPU0 interrupt priority selection (C0PIS) . . . 1321
Figure 731. Level2 Mux AND gates input pin routing for State0 . . . 1322
Figure 732. Level2 Mux state n selection 0 (L2nSEL0) . . . 1323
Figure 733. Level2 Mux state n selection 1 (L2nSEL1) . . . 1324
Figure 734. Level2 Mux state n selection 2 (L2nSEL2) . . . 1325
Figure 735. Level2 Mux state n selection 3 (L2nSEL3) . . . 1327
Figure 736. Input/output Inversion control state n (IOICn) . . . 1328
Figure 737. Sequence control (SCTRL) . . . 1329
Figure 738. State n goto control (SnGC) . . . 1332
Figure 739. SLU status register (SS) . . . 1333
Figure 740. SPU enable register (SE) . . . 1336
Figure 741. State n true action 0 (SnTA0) . . . 1338
Figure 742. State n true action 1 (SnTA1) . . . 1339
Figure 743. State n false action 0 (SnFA0) . . . 1340
Figure 744. State n false action 1 (SnFA1) . . . 1340
Figure 745. Trace group configuration register format . . . 1341
Figure 746. Interrupt status (INTS) . . . 1342
Figure 747. Counters control n registers (CCTRLn) . . . 1343
Figure 748. Counters compare n registers (CCMPn) . . . 1344
Figure 749. Counter compare status (CCOMS) . . . 1345
Figure 750. Counter overflow status (COS) . . . 1346
Figure 751. Counter capture status (CCAPS) . . . 1348
Figure 752. Input Mux unit block diagram . . . 1351
Figure 753. SLU block diagram . . . 1353
Figure 754. Example of a simple configurable if-then-else sequence . . . 1354
Figure 755. Complex sequence example . . . 1355
Figure 756. Counters/timers overview . . . 1356
Figure 757. Input selection for count increment . . . 1357
Figure 758. SPU counters increment assignment . . . 1357
Figure 759. Nexus 2+ Functional Block Diagram . . . 1361
Figure 760. Client Select Control register (CSC) . . . 1367
Figure 761. Port Configuration Register (PCR) . . . 1368
Figure 762. Development Control register 1 (DC1) . . . 1369
Figure 763. Development Control register 2 (DC2) . . . 1370
Figure 764. Development Control register 3 (DC3) . . . 1371
Figure 765. Development Control register 4 (DC4) . . . 1372
Figure 766. Development Status Register (DS) . . . 1373
Figure 767. Watchpoint Trigger register (WT) . . . 1375

Figure 768. Program Trace Start Trigger Control register (PTSTC) . . . 1376
Figure 769. Program Trace End Trigger Control register (PTETC) . . . 1377
Figure 770. Watchpoint Mask register (WMSK) . . . 1378
Figure 771. Nexus Overrun Control Register (OVCR) . . . 1379
Figure 772. Read/Write Access Control/Status register (RWCS) . . . 1380
Figure 773. Read/Write Access Data register (RWD) . . . 1381
Figure 774. Read/Write Access Address register (RWA) . . . 1382
Figure 775. JTAG DR format . . . 1383
Figure 776. Relative Address Generation and Re-creation . . . 1384
Figure 777. Debug Status Message Format . . . 1388
Figure 778. Error Message Format . . . 1388
Figure 779. Ownership Trace Message Format . . . 1389
Figure 780. Indirect Branch Message (History) Format . . . 1391
Figure 781. Indirect Branch Message Format . . . 1392
Figure 782. Direct Branch Message Format . . . 1392
Figure 783. Resource Full Message Format . . . 1394
Figure 784. Program Correlation Message Formats . . . 1395
Figure 785. Direct/Indirect Branch with Sync Message Format . . . 1396
Figure 786. Indirect Branch History w/ Sync. Message Format . . . 1396
Figure 787. Program Trace - Indirect Branch Message (Traditional) . . . 1397
Figure 788. Program Trace - Indirect Branch Message (History) . . . 1398
Figure 789. Program Trace - Direct Branch (Traditional) & Error Messages . . . 1398
Figure 790. Program Trace - Indirect Branch with Synchronization Message . . . 1398
Figure 791. .Data Acquisition Message Format . . . 1399
Figure 792. Watchpoint Message Format . . . 1400
Figure 793. Watchpoint Message & Watchpoint Error Message . . . 1401
Figure 794. Error Message Format . . . 1405
Figure 795. Single Pin MSEO Transfers . . . 1408
Figure 796. Dual Pin MSEO Transfers . . . 1409
Figure 797. Development control register 1 (DC1) . . . 1420
Figure 798. Development control register 2 (DC2) . . . 1421
Figure 799. Watchpoint trigger register (WT) . . . 1422
Figure 800. Data trace control (DTC) register . . . 1423
Figure 801. Data trace start address registers 1 and 2 (DTSA1 and DTSA2) . . . 1424
Figure 802. Data trace end address registers 1 and 2 (DTEA1 and DTEA2) . . . 1424
Figure 803. Breakpoint/watchpoint control register 1(BWC1) . . . 1425
Figure 804. Breakpoint/watchpoint control register 2(BWC2) . . . 1426
Figure 805. Breakpoint /watchpoint address registers 1 and 2 (BWA1 and BWA2) . . . 1427
Figure 806. JTAG DR for Nexus register access . . . 1428
Figure 807. DTM message format for DATA width = 32 bits . . . 1429
Figure 808. DTM error message format (fixed length = 48 bits) . . . 1430
Figure 809. Data write/read with synchronization message format . . . 1430
Figure 810. Relative addressing example . . . 1432
Figure 811. Watchpoint message format (fixed length = 18 bits) . . . 1434
Figure 812. Error message format (fixed length = 28 bits) . . . 1434
Figure 813. Peripheral DATA bus fields . . . 1434
Figure 814. CRC top level diagram . . . 1435
Figure 815. CRC Configuration Register n (CRC_CFGn) . . . 1437
Figure 816. CRC Input Register n (CRC_INPn) . . . 1438
Figure 817. CRC Current Status Register n (CRC_CSTATn) . . . 1439
Figure 818. CRC Output Register n (CRC_OUTPn) . . . 1439
Figure 819. CRC-CCITT engine concept scheme . . . 1440

Figure 820. CRC computation flow . . . 1441
Figure 821. DMA-CRC transmission sequence . . . 1443
Figure 822. MEMU block diagram . . . 1446
Figure 823. MEMU error buffer uniqueness check flow chart . . . 1449
Figure 824. MEMU reporting table uniqueness check flow chart . . . 1450
Figure 825. Control register (CTRL) . . . 1452
Figure 826. Error flag register (ERR_FLAG) . . . 1453
Figure 827. Debug register (DEBUG) . . . 1456
Figure 828. System RAM correctable error reporting table status register (SYS_RAM_CERR_STSn) . . . 1457
Figure 829. System RAM correctable error reporting table address register (SYS_RAM_CERR_ADDRn) . . . 1458
Figure 830. System RAM uncorrectable error reporting table status register (SYS_RAM_UNCERR_STS) . . . 1459
Figure 831. System RAM uncorrectable error reporting table address register (SYS_RAM_UNCERR_ADDR) . . . 1459
Figure 832. System RAM concurrent overflow registers (SYS_RAM_OFLWn) . . . 1460
Figure 833. Peripheral RAM correctable error reporting table status register (PERIPH_RAM_CERR_STSn) . . . 1460
Figure 834. Peripheral RAM correctable error reporting table address register (PERIPH_RAM_CERR_ADDRn) . . . 1461
Figure 835. Peripheral RAM uncorrectable error reporting table status register (PERIPH_RAM_UNCERR_STS) . . . 1462
Figure 836. Peripheral RAM uncorrectable error reporting table address register (PERIPH_RAM_UNCERR_ADDR) . . . 1462
Figure 837. Peripheral RAM concurrent overflow registers (PERIPH_RAM_OFLWn) . . . 1463
Figure 838. Flash memory correctable error reporting table status register (FLASH_CERR_STSn) 1463
Figure 839. Flash memory correctable error reporting table address register (FLASH_CERR_ADDRn) . . . 1464
Figure 840. Flash memory uncorrectable error reporting table status register (FLASH_UNCERR_STS) . . . 1465
Figure 841. Flash memory uncorrectable error reporting table address register (FLASH_UNCERR_ADDR) . . . 1465
Figure 842. Flash memory concurrent overflow registers (FLASH_OFLWn) . . . 1466
Figure 843. Generic representation of the reporting table of one MEMU reporting block . . . 1468
Figure 844. FCCU block diagram . . . 1472
Figure 845. FCCU state diagram . . . 1475
Figure 846. RF (ALARM state) recovery (a) . . . 1477
Figure 847. RF (ALARM state) recovery (b) . . . 1478
Figure 848. RF(ALARM -> FAULT state) recovery . . . 1479
Figure 849. NMI/WKUP scheme . . . 1480
Figure 850. FAULT injection . . . 1480
Figure 851. FAULT clearing . . . 1481
Figure 852. STCU-FCCU (case a) . . . 1481
Figure 853. STCU-FCCU (case b) . . . 1482
Figure 854. STCU-FCCU (case c) . . . 1482
Figure 855. Dual rail protocol (slow switching mode) . . . 1484
Figure 856. Dual rail protocol (fast switching mode) . . . 1485
Figure 857. Time switching protocol . . . 1486
Figure 858. Bi-stable protocol . . . 1487
Figure 859. Option1 , Option 2 . . . 1487
Figure 860. Option 3 . . . 1488

Figure 861. Error signal flow . . . 1489
Figure 862. FCCU Control Register (FCCU_CTRL) . . . 1493
Figure 863. FCCU CTRL Key Register (FCCU_CTRLK) . . . 1495
Figure 864. FCCU Configuration Register (FCCU_CFG) . . . 1496
Figure 865. FCCU RF Configuration Register n (FCCU_RF_CFGn) . . . 1499
Figure 866. FCCU RFS Configuration Register n (FCCU_RFS_CFGn) . . . 1500
Figure 867. FCCU RF Status register n (FCCU_RF_Sn) . . . 1502
Figure 868. FCCU RF Key register (FCCU_RFK) . . . 1503
Figure 869. FCCU RF Enable Register n (FCCU_RF_En) . . . 1504
Figure 870. FCCU RF Timeout Enable Register n (FCCU_RF_TOEn) . . . 1505
Figure 871. FCCU RF Timeout Register (FCCU_RF_TO) . . . 1506
Figure 872. FCCU CFG Timeout Register (FCCU_CFG_TO) . . . 1507
Figure 873. FCCU IO Control Register (FCCU_EINOUT) . . . 1508
Figure 874. FCCU Status Register (FCCU_STAT) . . . 1509
Figure 875. FCCU NA Freeze Status Register (N2AF_STATUS) . . . 1510
Figure 876. FCCU AF Freeze Status Register (A2FF_STATUS) . . . 1511
Figure 877. FCCU NF Freeze Status Register (N2FF_STATUS) . . . 1512
Figure 878. FCCU FA Freeze Status Register (F2A_STATUS) . . . 1513
Figure 879. FCCU RF Fake Register (FCCU_RFF) . . . 1514
Figure 880. FCCU IRQ Status Register (FCCU_IRQ_STAT) . . . 1515
Figure 881. FCCU IRQ Enable Register (FCCU_IRQ_EN) . . . 1516
Figure 882. FCCU XTMR Register (FCCU_XTMR) . . . 1517
Figure 883. FCCU MCS Register (FCCU_MCS) . . . 1518
Figure 884. FCCU Transient Register (FCCU_TRANS_LOCK) . . . 1520
Figure 885. FCCU Permanent Lock Register (FCCU_PERMNT_LOCK) . . . 1521
Figure 886. FCCU Delta T Register (FCCU_DELTA_T) . . . 1522
Figure 887. FCCU IRQ Alarm Enable register n (FCCU_IRQ_ALARM_ENn) . . . 1523
Figure 888. FCCU NMI Enable register n (FCCU_NMI_ENn) . . . 1524
Figure 889. FCCU EOUT Signaling Enable register n (FCCU_EOUT_SIG_ENn) . . . 1525
Figure 890. FOSU connections to the FCCU and MC_RGM . . . 1527
Figure 891. STCU2 block diagram . . . 1531
Figure 892. STCU2 Run Register (STCU_RUN) . . . 1538
Figure 893. STCU2 SK Code Register (STCU_SKC) . . . 1539
Figure 894. STCU2 Configuration Register (STCU_CFG) . . . 1540
Figure 895. STCU2 PLL Configuration Register (STCU_PLL_CFG) . . . 1542
Figure 896. STCU2 Watchdog Register Granularity (STCU_WDG) . . . 1543
Figure 897. STCU2 CRC Expected Status Register (STCU_CRCE) . . . 1544
Figure 898. STCU2 CRC Read Status Register (STCU_CRCR) . . . 1545
Figure 899. STCU2 Error Register (STCU_ERR_STAT) . . . 1546
Figure 900. STCU2 Error FM Register (STCU_ERR_FM) . . . 1548
Figure 901. STCU2 Off-line LBIST Status Register (STCU_LBS) . . . 1549
Figure 902. STCU2 Off-line LBIST End Flag Register (STCU_LBE) . . . 1549
Figure 903. STCU2 LBIST Unrecoverable FM Register (STCU_LBUFM) . . . 1550
Figure 904. STCU2 Off-line MBIST Status Low Register (STCU_MBSL) . . . 1551
Figure 905. STCU2 Off-line MBIST Status Medium Register (STCU_MBSM) . . . 1551
Figure 906. STCU2 Off-line MBIST Status High Register (STCU_MBSH) . . . 1552
Figure 907. STCU2 Off-line MBIST End Flag Low Register (STCU_MBEL) . . . 1553
Figure 908. STCU2 Off-line MBIST End Flag Medium Register (STCU_MBEM) . . . 1553
Figure 909. STCU2 Off-line MBIST End Flag High Register (STCU_MBEH) . . . 1554
Figure 910. STCU2 MBIST Unrecoverable FM Low Register (STCU_MBUFML) . . . 1555
Figure 911. STCU2 MBIST Unrecoverable FM Medium Register (STCU_MBUFMM) . . . 1556
Figure 912. STCU2 MBIST Unrecoverable FM High Register (STCU_MBUFMH) . . . 1557

Figure 913. STCU2 LBIST Control Register n (STCU_LB_CTRLn) 1558
Figure 914. STCU2 LBIST PC Stop Register n (STCU_LB_PCSn) 1560
Figure 915. STCU2 LBIST PRPG Low Register n (STCU_LB_PRPGLn) 1560
Figure 916. STCU2 LBIST PRPG High Register n (STCU_LB_PRPGHn) 1561
Figure 917. STCU2 Off-line LBIST MISR Expected Low Register n (STCU_LB_MISRELn). 1562
Figure 918. STCU2 LBIST MISR Expected High Register n (STCU_LB_MISREHn) 1562
Figure 919. STCU2 Off-line LBIST MISR Read Low Register n (STCU_LB_MISRRLn). 1563
Figure 920. STCU2 Off-line LBIST MISR Read High Register n (STCU_LB_MISRRHn) 1564
Figure 921. STCU2 MBIST Control Register n (STCU_MB_CTRLn) 1565
Figure 922. REG_PROT Block Diagram 1572
Figure 923. REG_PROT Memory Diagram 1573
Figure 924. Soft Lock Bit Register (SLBRn) 1576
Figure 925. Global Configuration Register (GCR) 1577
Figure 926. Change lock settings directly via Area #4 1578
Figure 927. Change lock settings for 16-bit protected addresses 1579
Figure 928. Change lock settings for 32-bit protected addresses 1579
Figure 929. Change lock settings for mixed protection 1580
Figure 930. Enable Locking Via Mirror Module Space (Area #3). 1580
Figure 931. Enable locking for protected and unprotected addresses. 1581
Figure 932. PASS module block diagram 1583
Figure 933. Lifecycle Status register (LCSTAT) 1586
Figure 934. CHSEL register 1587
Figure 935. Challenge Status register (CSTAT) 1587
Figure 936. Challenge Input register (CINn) 1588
Figure 937. Password Group n - Lock 0 Status register (LOCK0_PGn) 1589
Figure 938. Password Group n - Lock 1 status register (LOCK1_PGn) 1590
Figure 939. Password Group n - Lock 2 status register (LOCK2_PGn) 1591
Figure 940. Password Group n - Lock 3 status register (LOCK3_PGn) 1592
Figure 941. Censorship register (not accessible) 1595

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

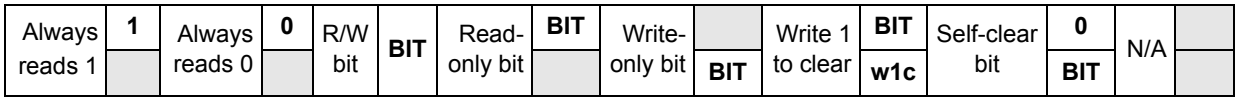


Table 1. Register conventions

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
u	Unaffected by reset
[*signal_name*]	Reset value is determined by polarity of indicated signal.

1.2 Acronyms and abbreviations

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

1.3 Reference documents

In addition to this reference manual, the following documents provide additional information on the operation of the SPC570Sx:

- IEEE-ISTO 5001-2003 Standard for a Global Embedded Processor Interface (Nexus)
- IEEE 1149.1-2001 standard - IEEE Standard Test Access Port and Boundary-Scan Architecture

2 Introduction

2.1 SPC570Sx microcontroller

The SPC570Sx microcontroller is a new member in a family of devices superseding the SPC56 family. SPC570Sx builds on the legacy of the SPC56 family, while introducing new features coupled with higher throughput to provide substantial reduction of cost per feature and significant power and performance improvement. The SPC570Sx device implements two e200z0h processor cores:

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

2.1.1 Target applications

This device is targeted at the following automotive applications:

- Antilock braking system (ABS)
- Airbag

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*, as well as in the SPC570Sx *Safety Manual* (application note AN4247).

2.1.2 Overall architecture

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

2.1.3 Core features

The architecture consists of an e200z0h CPU operating up to 80 MHz with a second identical CPU running as a safety checker core in delayed lockstep mode. The e200z0h is compatible with the Power Architecture® VLE instruction set, that supports code size reduction. The Power Architecture has enhancements that improve the architecture's fit in embedded applications.

2.1.4 Memory hierarchy

The SPC570Sx has two levels of memory hierarchy. At the top level is system RAM (SRAM) and flash memory. Two peripheral bridges provide access to the I/O peripherals.

2.2 Features

On-chip modules within the SPC570Sx include the following features:

- 2 main CPUs, single-issue, 32-bit CPU core complexes (e200z0h), running in lockstep
  - Power Architecture embedded specification compliance
  - Instruction set enhancement allowing variable length encoding (VLE), encoding a mix of 16-bit and 32-bit instructions, for code size footprint reduction
- Up to 544 KB (512 KB code + 32 KB data, suitable for EEPROM emulation) on-chip flash memory: supports read during program and erase operations, and multiple blocks allowing EEPROM emulation
- Up to 48 KB on-chip general-purpose SRAM
- Multi-channel direct memory access controller (eDMA paired in lockstep)
  - 16 channels per eDMA
- Interrupt controller (INTC) with dedicated interrupt source channels, including software interrupts and 32 priority levels
- Dual phase-locked loops with stable clock domain for peripherals and frequency modulation domain for computational shell
- Crossbar switch architecture for concurrent access to peripherals, flash memory, or SRAM from multiple bus masters with end-to-end ECC
- System integration unit lite (SIUL2)
- Boot Assist Flash (BAF) supports factory programming using serial bootload through ‘UART Serial Boot Mode Protocol’. Physical Interface (PHY) can be
  - UART / LIN
  - CAN
- Enhanced analog-to-digital converter system
  - 2 separate 12-bit SAR analog converters
  - 1.5 µs conversion time (at 12 MHz)
  - Up to 16 physical channels
- Temperature sensor
  - Range –40 to +150 °C
  - Sensitivity approximately 5.14 mV/°C
- STCU2
  - Support for Logic BIST and Memory BIST at power on
  - ASIL D
- 3 deserial serial peripheral interface (DSPI) modules
- Up to 2 LIN and UART communication interface (LINFlexD) modules
  - LINFlexD_0 (master/slave)
  - LINFlexD_1 (master)
- Up to 2 FlexCAN modules
- Nexus development interface (NDI) per IEEE-ISTO 5001-2003 standard, with partial support for 2010 standard
- Device and board test support per Joint Test Action Group (JTAG) (IEEE 1149.1)
- On-chip voltage regulator controller manages the supply voltage down to 1.2 V for core logic

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 SPC570Sx 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. SPC570Sx device feature summary (Family Superset Configuration)

Feature		Description
Process		55 nm
Main processor	Core	e200z0h
	Number of main cores	1
	Number of checker cores	1
	VLE	Yes
	Main processor frequency	80 MHz(1)
Interrupt controllers (including interrupt controller checker)		1
Software watchdog timer		1
System timers		1 AUTOSAR® STM 1 PIT with four 32-bit channels
DMA (including DMA checker)		1
DMA channels		16
SMPU		Yes (8 regions)(2)
System SRAM		Up to 48 KB
Code flash memory		Up to 512 KB
Data flash memory (suitable for EEPROM emulation)		32 KB
UTEST flash memory		8 KB
Boot assist flash (BAF)		8 KB
CRC		1
LINFlexD		Up to 2
FlexCAN		Up to 2
DSPI		3
eTimer		4 x 6 channels
ADC (SAR)		2(3)

Table 2. SPC570Sx device feature summary (Family Superset Configuration)(Continued)

Feature		Description
CTU (Cross Triggering Unit)		1
Temperature sensor		1
Self-test control unit (memory and logic BIST)		1
FCCU		1
MEMU		1
PLL		Dual PLL with FM
Nexus		3(4)
Sequence processing unit (SPU)		1
External power supplies		5 V(5) 3.3 V(5)
Junction temperature		–40 to 150 °C
Packages	Device SPC570SxxE3	eTQFP100
	Device SPC570SxxE1	eTQFP64

1. Includes user programmable CPU core and one safety core. The two e200z0h processors in the lockstep pair run at 80 MHz. The e200z0h is compatible with the Power Architecture embedded specification.
2. SMPU with process ID support extension
3. One ADC can be used as supervisor ADC
4. Including trace for the crossbar masters (data & instruction trace on core and data trace on eDMA). 4 MDO pin Nexus trace port
5. All I/Os can be supplied at 3.3 V or 5 V (mutually exclusive)

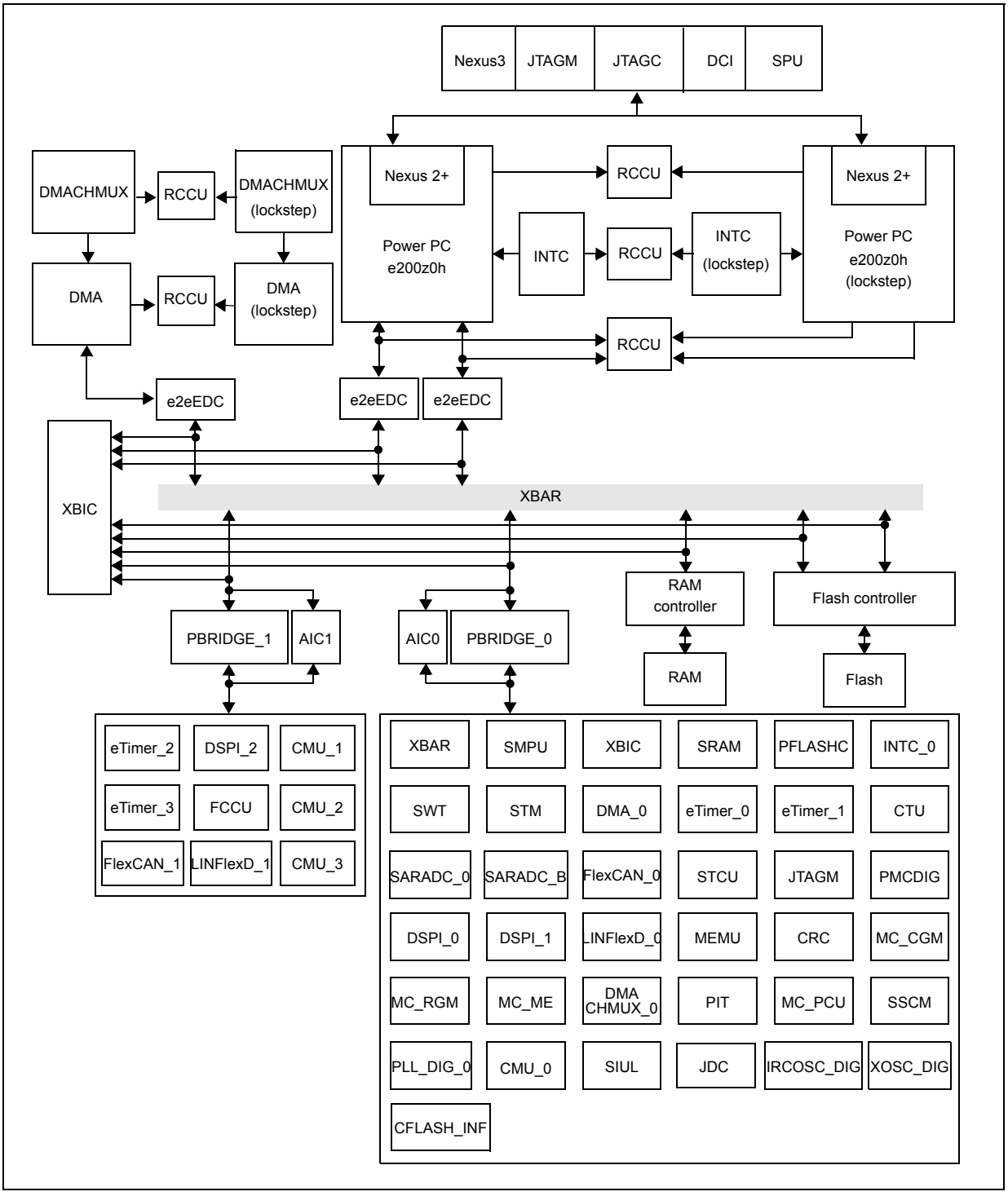
2.4 Packages

The SPC570Sx is available in two packages: eTQFP100 and eTQFP64.

2.5 Block diagram

Figure 2 shows the top-level block diagram.

Figure 2. Block diagram



3 Embedded memories

3.1 Overview

The embedded memory architecture for SPC570Sx devices is a significant departure from previous-generation devices. Features include:

- Onboard system SRAM
- Onboard system error correction code (ECC) flash memory
- Embedded memories in the peripherals[a]:
  - CAN
  - eDMA

3.2 SRAM

SPC570Sx devices include up to 48 KB of general-purpose on-chip ECC SRAM. The SRAM can be configured for either zero- or one-wait state read operation latency using the PRCR1 register in the PRAM controller.

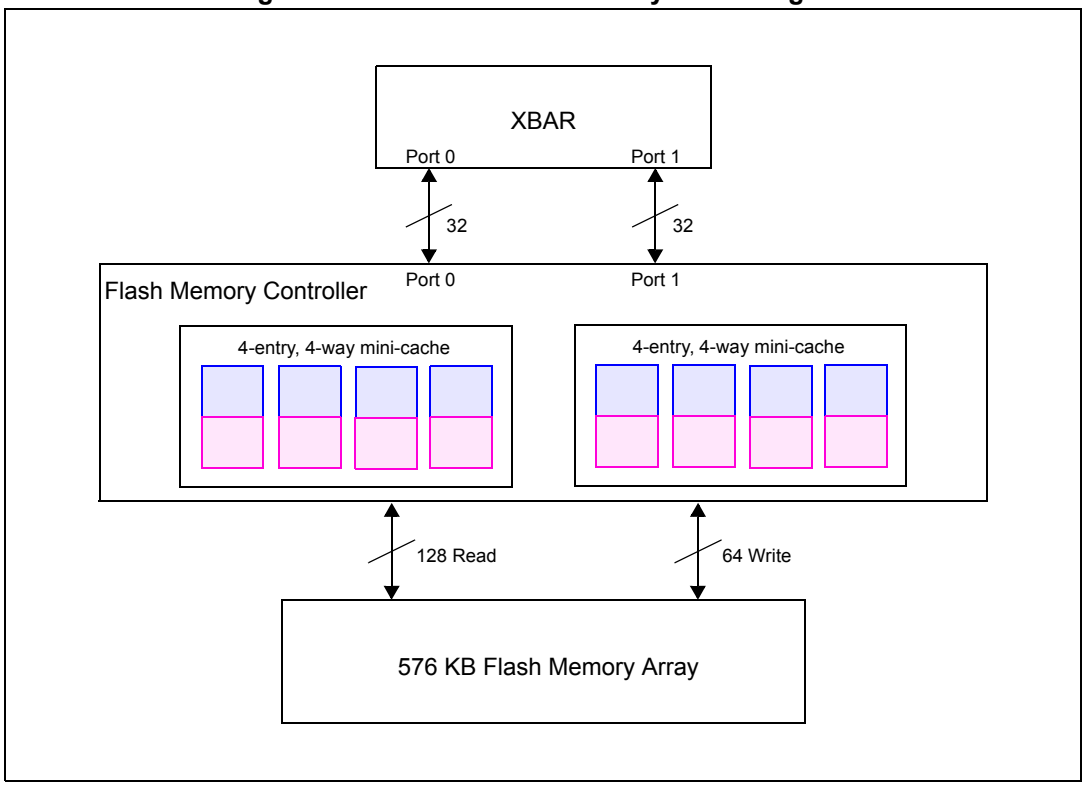
- See *Chapter 29: Platform RAM controller (PRAM)* for details on configuring the PRAM controller.
- See *Table 9: Memory map (XBAR slave ports)* in *Chapter 5: Memory map* for the SPC570Sx SRAM address range.

3.3 Flash memory

Flash memory on SPC570Sx 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 3* shows the memory architecture.

a. Refer to the respective module chapter for details.

Figure 3. SPC570Sx flash memory block diagram



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

3.3.1 Flash memory controller

The SPC570Sx flash memory controller acts as an interface between the system bus and the flash memory array.

The flash memory controller contains a four-entry, four-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 128-bit (16-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 30: Flash memory controller (PFLASH Controller)*, for details.

3.3.2 Flash memory array

The first partition in the array includes up to 512 KB of Code Flash and a 16 KB one-time programmable (OTP) region comprising UTEST (8 KB) and BAF (8 KB), included to support systems that require non-volatile memory for security features or system initialization information. The second 32-KB partition is EE-Emulated Data Flash.

Reads of the embedded flash memory return a 128-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.

3.3.2.1 Features

- Flash segmentation
  - 4x 16 KB, 2x 32 KB, 2x 64 KB, and 2x 128 KB blocks for code
  - 4x 8 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 8 KB OTP UTEST and BAF block
- ECC with double bit detection, single bit correction
- Erase suspend, program suspend, and erase-suspended program all supported

3.3.2.2 Flash organization

The embedded flash memory consists of three address spaces in five groupings:

- 16 KB block in low address space, 32 KB block in low address space and 64 KB block in low address space
- 8 KB block in high address space
- 128 KB block in 128 KB address space

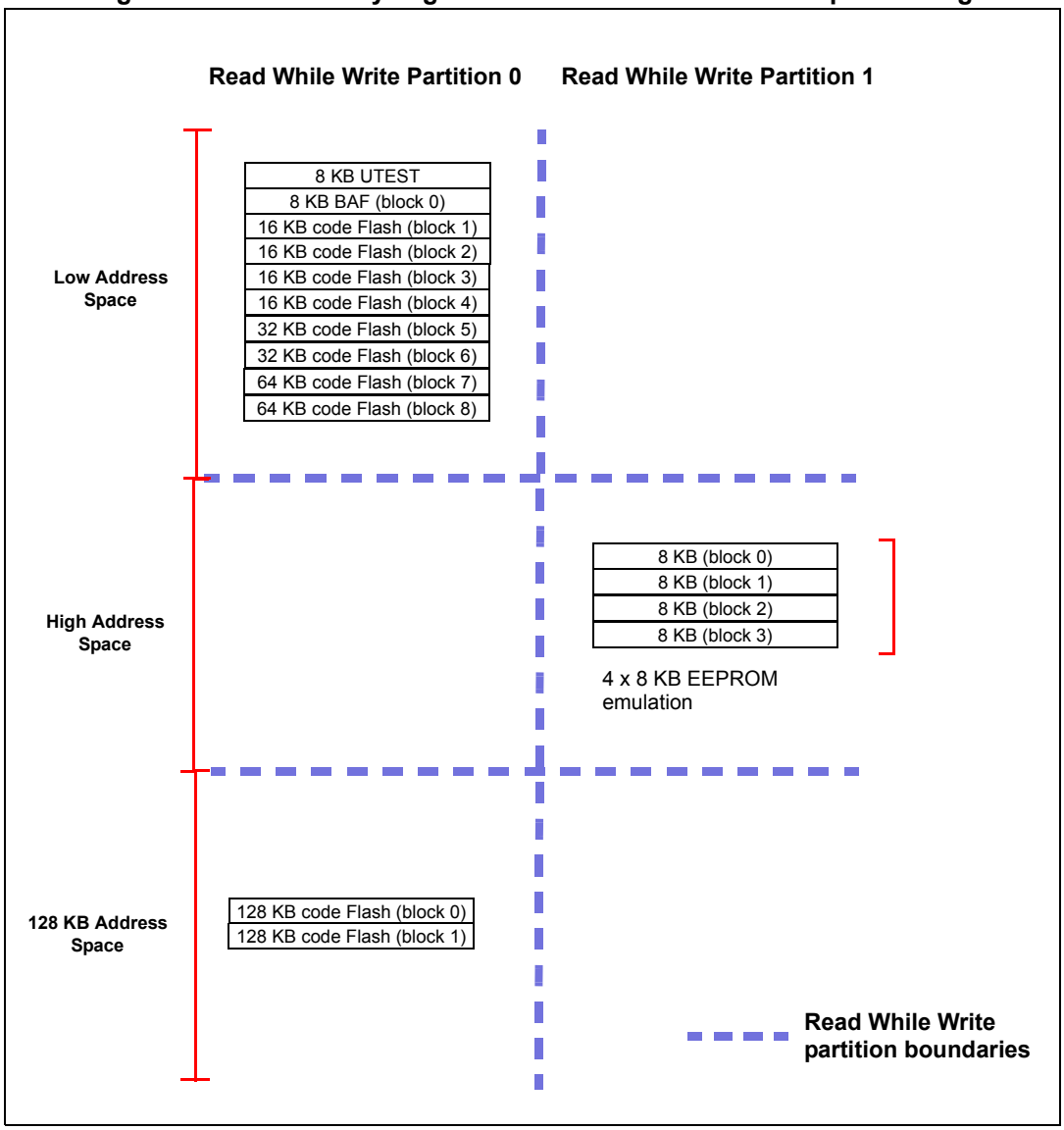
Address space group (low- or high-address space) determines which registers/fields are used to select blocks for modify operations or lock them from modify operations. Each address space is independent.

The flash module is further divided into two 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 the 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 4 shows the flash block segmentation and read-while-write partitioning for SPC570Sx devices.

Figure 4. Flash memory segmentation and read-while-write partitioning



Note: See Table 10: Flash memory map in Chapter 5: Memory map for the SPC570Sx flash memory map.

3.3.2.3 UTEST memory space

Devices in the SPC570Sx family contain an 8 KB area of OTP flash memory for storing test information and device configuration data. See *Table 13: UTest memory map* in *Chapter 5: Memory map* for the SPC570Sx UTEST flash memory map.

3.4 Security features

The SPC570Sx 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.

4 Signal description

4.1 Pin descriptions

The following sections provide signal descriptions and related information about the functionality and configuration of the SPC570Sx devices.

4.1.1 Pin startup and reset states

Table 3 provides startup state and reset state information for device pins.

Table 3. Pin startup and reset states

Pin	Startup state(1)	State during reset	State after reset
I/O pins(2)	Weak pullup	Input, weak pullup	Input, weak pullup
Analog Inputs(3)	High-impedance	High-impedance	High-impedance
F0 (ERROR0)	Weak pullup	Input, weak pullup	Input, weak pullup
F1 (ERROR1)	Weak pullup	Input, weak pullup	Input, weak pullup
PORST	Weak pulldown	Input, weak pulldown	Input, weak pulldown
TESTMODE(4)	Weak pulldown	Input, weak pulldown	Input, weak pulldown
TDI	High-impedance	Input, weak pullup	Input, weak pullup
TDO	High-impedance	High-impedance	High-impedance
TMS	High-impedance	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 4 contains information on power supply and reference pin functions for the devices.

Table 4. Power supply and reference pins

Supply			Pin No.	
Pad	Type	Description	eTQFP64	eTQFP100
VSS_HV	Ground	High voltage ground	Exposed pad	Exposed pad
VSS_LV	Ground	Low voltage ground	Exposed pad	Exposed pad

Table 4. Power supply and reference pins(Continued)

Supply			Pin No.	
Pad	**Type**	**Description**	**eTQFP64**	**eTQFP100**
VSS_HV_OSC_PMC	Ground	Ground supply for the internal power management unit and the oscillator	Exposed pad	Exposed pad
VDD_LV	Power	Low voltage power supply for device (PLL is also powered by this pin.)	12, 34, 45	19, 52, 68
VDD_HV_IO	Power	High voltage power supply for I/O	13, 33, 46, 54, 61(1)	20, 51, 85, 95(1)
VDD_HV_OSC_PMC	Power	High voltage power supply for internal power management unit and oscillator	37	55
VSS_HV_ADC_TSENS	Ground	Ground supply for ADC and TSENS	Exposed pad	Exposed pad
VDD_HV_ADC_TSENS	Power	High voltage supply for ADC and TSENS	25	39
VREFL_ADC	Reference	Ground reference of ADCs	Exposed pad	Exposed pad
VREFH_ADC	Reference	Voltage reference of ADCs	19	29

1. Pins 61 and 95 are also used as high voltage supply of flash

4.1.3 System pins

Table 5 contains information on system pin functions for the devices.

Table 5. System pins

Pad	Description	Direction	Pin No.	
			eTQFP64	**eTQFP100**
PORST	Power on reset with Schmitt trigger characteristics and noise filter. PORST is active low.	Bidirectional	44	66
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	43	64
XTAL	Analog output of the oscillator amplifier circuit—to be grounded if oscillator is used in bypass mode.	Output	36	54
EXTAL	– Analog input of the oscillator amplifier circuit when oscillator is not in bypass mode – Analog input for the clock generator when oscillator is in bypass mode	Input	35	53

4.1.4 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 13: System Integration Unit Lite2 (SIUL2).

Table 6. Pin multiplexing description

Port pin	SIUL2 MSCR		Signal	Module	Full signal name	Direction	Strength				GPIO_MAPPED	MSCR_MAPPED
	No.	SSS value					VeryStrong	Strong	Medium	Weak		
PA[0]	0	0000	GPIO0	SIUL2-GPIO0	GPIO0	I/O	-	-	x	x	1	1
		0001	CS0	DSPI_0	DSPI 0 - CS 0 (Master/Slave)	I/O	-	-	x	x		
		0010	Reserved				-	-	x	x		
		0011	CS1	DSPI_1	DSPI 1 - CS 1 (Master)	O	-	-	x	x		
		0100	ETC_0	ETIMER_0	Timer 0 - ch. 0 (IC/OC)	I/O	-	-	x	x		
		0101–1111	Reserved				-	-	x	x		
	626	00000010	CS0	DSPI_0	DSPI0 Slave Select Input	I	-	-	x	x		
	688	00000101	ETC_0	ETIMER_0	ETMR0 input channel 0	I	-	-	x	x		
	816	00000010	REQ0	SIUL	interrupt request channel 0	I	-	-	x	x		

Table 6. Pin multiplexing description(Continued)

Port pin	SIUL2 MSCR		Signal	Module	Full signal name	Direction	Strength				GPIO_MAPPED	MSCR_MAPPED
	No.	SSS value					VeryStrong	Strong	Medium	Weak		
PA[1]	1	0000	GPIO1	SIUL2-GPIO1	GPIO1	I/O	-	-	x	x	1	1
		0001	CS1	DSPI_1	DSPI 1 - CS 1 (Master)	O	-	-	x	x		
		0010	ETC_0	ETIMER_0	Timer 0 - ch. 0 (IC/OC)	I/O	-	-	x	x		
		0011	IOMUX_EVTI_B00	INTG_DEBUG	Nexus EVTI	I	-	-	x	x		
		0100	ETC_0	ETIMER_1	Timer 1 - ch. 0 (IC/OC)	I/O	-	-	x	x		
		0101–1111	Reserved				-	-	x	x		
	688	00000011	ETC_0	ETIMER_0	ETMR0 input channel 0	I	-	-	x	x		
	694	00000101	ETC_0	ETIMER_1	ETMR1 input channel 0	I	-	-	x	x		
PA[2]	2	0000	GPIO2	SIUL2-GPIO2	GPIO2	I/O	x	x	x	x	1	1
		0001	CS1	DSPI_2	DSPI 2 - CS 1 (Master)	O	x	x	x	x		
		0010	CS4	DSPI_0	DSPI 0 - CS 4 (Master)	O	x	x	x	x		
		0011	EVTO_B0	DCI	Nexus EVTO	O	x	x	x	x		
		0100	ETC_1	ETIMER_1	Timer 1 - ch. 1 (IC/OC)	I/O	x	x	x	x		
		0101–1111	Reserved				x	x	x	x		
	695	00000101	ETC_1	ETIMER_1	ETMR1 input channel 1	I	x	x	x	x		

Table 6. Pin multiplexing description(Continued)

Port pin	SIUL2 MSCR		Signal	Module	Full signal name	Direction	Strength				GPIO_MAPPED	MSCR_MAPPED
	No.	SSS value					VeryStrong	Strong	Medium	Weak		
PA[3]	3	0000	GPIO3	SIUL2-GPIO3	GPIO3	I/O	x	x	x	x	1	1
		0001	SCK	DSPI_0	DSPI 0 - CLK (Master/Slave)	I/O	x	x	x	x		
		0010	Reserved				x	x	x	x		
		0011	ETC_0	ETIMER_0	Timer 0 - ch. 0 (IC/OC)	I/O	x	x	x	x		
		0100	SCK	DSPI_1	DSPI 1 - CLK (Master/Slave)	I/O	x	x	x	x		
		0101–1111	Reserved				x	x	x	x		
	625	00000010	SCK	DSPI_0	DSPI0 Serial Clock Input	I	x	x	x	x		
	628	00000110	SCK	DSPI_1	DSPI1 Serial Clock Input	I	x	x	x	x		
	688	00000100	ETC_0	ETIMER_0	ETMR0 input channel 0	I	x	x	x	x		
	817	00000010	REQ1	SIUL	interrupt request channel 1	I	x	x	x	x		
PA[4]	4	0000	GPIO4	SIUL2-GPIO4	GPIO4	I/O	-	-	x	x	1	1
		0001	SOUT	DSPI_0	DSPI 0 - Serial Data	I/O	-	-	x	x		
		0010	Reserved				-	-	x	x		
		0011	ETC_1	ETIMER_0	Timer 0 - ch. 1 (IC/OC)	I/O	-	-	x	x		
		0100	SOUT	DSPI_1	DSPI 1 - Serial Data	I/O	-	-	x	x		
		0101–1111	Reserved				-	-	x	x		
	624	00000001	SIN	DSPI_0	DSPI0 Serial Data Input	I	-	-	x	x		
	627	00000101	SIN	DSPI_1	DSPI1 Serial Data Input	I	-	-	x	x		
	689	00000100	ETC_1	ETIMER_0	ETMR0 input channel 1	I	-	-	x	x		
	944	00000010	NMI	WKPU	NMI input	I	-	-	x	x		

Table 6. Pin multiplexing description(Continued)

Port pin	SIUL2 MSCR		Signal	Module	Full signal name	Direction	Strength				GPIO_MAPPED	MSCR_MAPPED
	No.	SSS value					VeryStrong	Strong	Medium	Weak		
PA[5]	5	0000	GPIO5	SIUL2-GPIO5	GPIO5	I/O	x	x	x	x	1	1
		0001	TXD	LINFlexD_1	LINFlexD 1 - TX	O	x	x	x	x		
		0010	ETC_1	ETIMER_0	Timer 0 - ch. 1 (IC/OC)	I/O	x	x	x	x		
		0011	MCKO	NPC_WRAPPER	Nexus MCK0	O	x	x	x	x		
		0100	ETC_2	ETIMER_1	Timer 1 - ch. 2 (IC/OC)	I/O	x	x	x	x		
		0101–1111	Reserved				x	x	x	x		
	689	00000011	ETC_1	ETIMER_0	ETMR0 input channel 1	I	x	x	x	x		
	696	00000101	ETC_2	ETIMER_1	ETMR1 input channel 2	I	x	x	x	x		
PA[6]	6	0000	GPIO6	SIUL2-GPIO6	GPIO6	I/O	x	x	x	x	1	1
		0001	Reserved				x	x	x	x		
		0010	ETC_2	ETIMER_0	Timer 0 - ch. 2 (IC/OC)	I/O	x	x	x	x		
		0011	MDO_0	NPC_WRAPPER	Nexus MDO 0	I/O	x	x	x	x		
		0100	ETC_3	ETIMER_1	Timer 1 - ch. 3 (IC/OC)	I/O	x	x	x	x		
		0101–1111	Reserved				x	x	x	x		
	561	00000001	RXD	LINFlexD_1	LINFlexD 1 Receive Data	I	x	x	x	x		
	690	00000011	ETC_2	ETIMER_0	ETMR0 input channel 2	I	x	x	x	x		
	697	00000101	ETC_3	ETIMER_1	ETMR1 input channel 3	I	x	x	x	x		

Table 6. Pin multiplexing description(Continued)

Port pin	SIUL2 MSCR		Signal	Module	Full signal name	Direction	Strength				GPIO_MAPPED	MSCR_MAPPED
	No.	SSS value					VeryStrong	Strong	Medium	Weak		
PA[7]	7	0000	GPIO7	SIUL2-GPIO7	GPIO7	I/O	x	x	x	x	1	1
		0001	SOUT	DSPI_0	DSPI 0 - Serial Data	I/O	x	x	x	x		
		0010	Reserved				x	x	x	x		
		0011	ETC_2	ETIMER_0	Timer 0 - ch. 2 (IC/OC)	I/O	x	x	x	x		
		0100	SOUT	DSPI_1	DSPI 1 - Serial Data	I/O	x	x	x	x		
		0101–1111	Reserved				x	x	x	x		
	624	00000010	SIN	DSPI_0	DSPI0 Serial Data Input	I	x	x	x	x		
	627	00000110	SIN	DSPI_1	DSPI1 Serial Data Input	I	x	x	x	x		
	690	00000100	ETC_2	ETIMER_0	ETMR0 input channel 2	I	x	x	x	x		
PA[8]	8	0000	GPIO8	SIUL2-GPIO8	GPIO8	I/O	-	-	x	x	1	1
		0001	CS1	DSPI_0	DSPI 0 - CS 1 (Master)	O	-	-	x	x		
		0010	Reserved				-	-	x	x		
		0011	TXD	LINFlexD_1	LINFlexD 1 - TX	O	-	-	x	x		
		0100	ETC_1	ETIMER_0	Timer 0 - ch. 1 (IC/OC)	I/O	-	-	x	x		
		0101–1111	Reserved				-	-	x	x		
	689	00000101	ETC_1	ETIMER_0	ETMR0 input channel 1	I	-	-	x	x		

Table 6. Pin multiplexing description(Continued)

Port pin	SIUL2 MSCR		Signal	Module	Full signal name	Direction	Strength				GPIO_MAPPED	MSCR_MAPPED
	No.	SSS value					VeryStrong	Strong	Medium	Weak		
PA[9]	9	0000	GPIO9	SIUL2-GPIO9	GPIO9	I/O	-	-	x	x	1	1
		0001	CS2	DSPI_0	DSPI 0 - CS 2 (Master)	O	-	-	x	x		
		0010	CS7	DSPI_0	DSPI 0 - CS 7 (Master)	O	-	-	x	x		
		0011	Reserved				-	-	x	x		
		0100	ETC_2	ETIMER_0	Timer 0 - ch. 2 (IC/OC)	I/O	-	-	x	x		
		0101–1111	Reserved				-	-	x	x		
	561	00000010	RXD	LINFlexD_1	LINFlexD 1 Receive Data	I	-	-	x	x		
	690	00000101	ETC_2	ETIMER_0	ETMR0 input channel 2	I	-	-	x	x		
PA[10]	10	0000	GPIO10	SIUL2-GPIO10	GPIO10	I/O	x	x	x	x	1	1
		0001	Reserved				x	x	x	x		
		0010	CS1	DSPI_1	DSPI 1 - CS 1 (Master)	O	x	x	x	x		
		0011	MDO_1	NPC_WRAPPER	Nexus MDO 1	I/O	x	x	x	x		
		0100–1111	Reserved				x	x	x	x		
	819	00000101	REQ3	SIUL	interrupt request channel 2	I	x	x	x	x		
PA[11]	11	0000	GPIO11	SIUL2-GPIO11	GPIO11	I/O	-	-	x	x	1	1
		0001	CS3	DSPI_0	DSPI 0 - CS 3 (Master)	O	-	-	x	x		
		0010	CS5	DSPI_0	DSPI 0 - CS 5 (Master)	O	-	-	x	x		
		0011	ETC_3	ETIMER_0	Timer 0 - ch. 3 (IC/OC)	—	-	-	x	x		
		0100–1111	Reserved				-	-	x	x		
	691	00000100	ETC_3	ETIMER_0	ETMR0 input channel 3	I	-	-	x	x		
	820	00000101	REQ4	SIUL	interrupt request channel 2	I	-	-	x	x		

Table 6. Pin multiplexing description(Continued)

Port pin	SIUL2 MSCR		Signal	Module	Full signal name	Direction	Strength				GPIO_MAPPED	MSCR_MAPPED
	No.	SSS value					VeryStrong	Strong	Medium	Weak		
PA[12]	12	0000	GPIO12	SIUL2-GPIO12	GPIO12	I/O	x	x	x	x	1	1
		0001–0011	Reserved				x	x	x	x		
		0100	ETC_3	ETIMER_0	Timer 0 - ch. 3 (IC/OC)	I/O	x	x	x	x		
		0101–1111	Reserved				x	x	x	x		
	513	00000010	RXD	FlexCAN_1	FlexCAN 1 Receive Data	I	x	x	x	x		
	560	00000001	RXD	LINFlexD_0	LINFlexD 0 Receive Data	I	x	x	x	x		
	561	00000011	RXD	LINFlexD_1	LINFlexD 1 Receive Data	I	x	x	x	x		
	691	00000101	ETC_3	ETIMER_0	ETMR0 input channel 3	I	x	x	x	x		
PA[13]	13	0000	GPIO13	SIUL2-GPIO13	GPIO13	I/O	x	x	x	x	1	1
		0001	TXD	LINFlexD_0	LINFlexD 0 - TX	O	x	x	x	x		
		0010	TXD	FlexCAN_1	FlexCAN 1 - TX	O	x	x	x	x		
		0011	TXD	LINFlexD_1	LINFlexD 1 - TX	O	x	x	x	x		
		0100	ETC_4	ETIMER_0	Timer 0 - ch. 4 (IC/OC)	I/O	x	x	x	x		
		0101–1111	Reserved				x	x	x	x		
	692	00000101	ETC_4	ETIMER_0	ETMR0 input channel 4	I	x	x	x	x		

Table 6. Pin multiplexing description(Continued)

Port pin	SIUL2 MSCR		Signal	Module	Full signal name	Direction	Strength				GPIO_MAPPED	MSCR_MAPPED
	No.	SSS value					VeryStrong	Strong	Medium	Weak		
PA[14]	14	0000	GPIO14	SIUL2-GPIO14	GPIO14	I/O	-	-	x	x	1	1
		0001	ETC_4	ETIMER_0	Timer 0 - ch. 4 (IC/OC)	I/O	-	-	x	x		
		0010	CS1	DSPI_1	DSPI 1 - CS 1 (Master)	O	-	-	x	x		
		0011	Reserved				-	-	x	x		
		0100	ETC_5	ETIMER_0	Timer 0 - ch. 5 (IC/OC)	I/O	-	-	x	x		
		0101–1111	Reserved				-	-	x	x		
	692	00000010	ETC_4	ETIMER_0	ETMR0 input channel 4	I	-	-	x	x		
	693	00000101	ETC_5	ETIMER_0	ETMR0 input channel 5	I	-	-	x	x		
	819	00000010	REQ3	SIUL	interrupt request channel 2	I	-	-	x	x		
PA[15]	15	0000	GPIO15	SIUL2-GPIO15	GPIO15	I/O	x	x	x	x	1	1
		0001	Reserved				x	x	x	x		
		0010	ETC_0	ETIMER_1	Timer 1 - ch. 0 (IC/OC)	I/O	x	x	x	x		
		0011	MDO_2	NPC_WRAPPER	Nexus MDO 2	I/O	x	x	x	x		
		0100	ETC_4	ETIMER_1	Timer 1 - ch. 4 (IC/OC)	I/O	x	x	x	x		
		0101–1111	Reserved				x	x	x	x		
	513	00000001	RXD	FlexCAN_1	FlexCAN 1 Receive Data	I	x	x	x	x		
	694	00000011	ETC_0	ETIMER_1	ETMR1 input channel 0	I	x	x	x	x		
	698	00000101	ETC_4	ETIMER_1	ETMR1 input channel 4	I	x	x	x	x		

Table 6. Pin multiplexing description(Continued)

Port pin	SIUL2 MSCR		Signal	Module	Full signal name	Direction	Strength				GPIO_MAPPED	MSCR_MAPPED
	No.	SSS value					VeryStrong	Strong	Medium	Weak		
PB[0]	16	0000	GPIO16	SIUL2-GPIO16	GPIO16	I/O	-	-	x	x	1	1
		0001	TXD	FlexCAN_1	FlexCAN 1 - TX	O	-	-	x	x		
		0010	ETC_1	ETIMER_1	Timer 1 - ch. 1 (IC/OC)	I/O	-	-	x	x		
		0011	MDO_3	NPC_WRAPPER	Nexus MDO 3	I/O	-	-	x	x		
		0100	ETC_5	ETIMER_1	Timer 1 - ch. 5 (IC/OC)	I/O	-	-	x	x		
		0101–1111	Reserved				-	-	x	x		
	695	00000011	ETC_1	ETIMER_1	ETMR1 input channel 1	I	-	-	x	x		
	699	00000101	ETC_5	ETIMER_1	ETMR1 input channel 5	I	-	-	x	x		
PB[1]	17	0000	GPIO17	SIUL2-GPIO17	GPIO17	I/O	x	x	x	x	1	1
		0001	ETC_5	ETIMER_1	Timer 1 - ch. 5 (IC/OC)	I/O	x	x	x	x		
		0010	CS6	DSPI_0	DSPI 0 - CS 6 (Master)	O	x	x	x	x		
		0011	MSEO_B_0	NPC_WRAPPER	Nexus MSEO0	O	x	x	x	x		
		0100	CS0	DSPI_1	DSPI 1 - CS 0 (Master/Slave)	I/O	x	x	x	x		
		0101–1111	Reserved				x	x	x	x		
	629	00000110	CS0	DSPI_1	DSPI1 Slave Select Input	I	x	x	x	x		
	699	00000001	ETC_5	ETIMER_1	ETMR1 input channel 5	I	x	x	x	x		
PB[2]	18	0000	GPIO18	SIUL2-GPIO18	GPIO18	I	-	-	-	-	1	1
		0001–1111	Reserved				-	-	-	-		
	512	00000010	RXD	FlexCAN_0	FlexCAN 0 Receive Data	I	-	-	-	-		
	692	00000001	ETC_4	ETIMER_0	ETMR0 input channel 4	I	-	-	-	-		
	819	00000011	REQ3	SIUL	interrupt request channel 2	I	-	-	-	-		
	—	—	ANP[15]	SAR_ADC	analog channel	I	-	-	-	-		

Table 6. Pin multiplexing description(Continued)

Port pin	SIUL2 MSCR		Signal	Module	Full signal name	Direction	Strength				GPIO_MAPPED	MSCR_MAPPED
	No.	SSS value					VeryStrong	Strong	Medium	Weak		
PB[3]	19	0000	GPIO19	SIUL2-GPIO19	GPIO19	I	-	-	-	-	1	1
		0001–1111	Reserved				-	-	-	-		
	624	00000111	SIN	DSPI_0	DSPI0 Serial Data Input	I	-	-	-	-		
	688	00000001	ETC_0	ETIMER_0	ETMR0 input channel 0	I	-	-	-	-		
	694	00000100	ETC_0	ETIMER_1	ETMR1 input channel 0	I	-	-	-	-		
	—	—	ANP[9]	SAR_ADC	analog channel	I	-	-	-	-		
PB[4]	20	0000	GPIO20	SIUL2-GPIO20	GPIO20	I	-	-	-	-	1	1
		0001–1111	Reserved				-	-	-	-		
	627	00000111	SIN	DSPI_1	DSPI1 Serial Data Input	I	-	-	-	-		
	689	00000001	ETC_1	ETIMER_0	ETMR0 input channel 1	I	-	-	-	-		
	695	00000100	ETC_1	ETIMER_1	ETMR1 input channel 1	I	-	-	-	-		
	—	—	ANP[8]	SAR_ADC	analog channel	I	-	-	-	-		
PB[5]	21	0000	GPIO21	SIUL2-GPIO21	GPIO21	I	-	-	-	-	1	1
		0001–1111	Reserved				-	-	-	-		
	630	00000111	SIN	DSPI_2	DSPI2 Serial Data Input	I	-	-	-	-		
	690	00000001	ETC_2	ETIMER_0	ETMR0 input channel 2	I	-	-	-	-		
	696	00000100	ETC_2	ETIMER_1	ETMR1 input channel 2	I	-	-	-	-		
	—	—	ANP[7]	SAR_ADC	analog channel	I	-	-	-	-		
PB[6]	22	0000	GPIO22	SIUL2-GPIO22	GPIO22	I	-	-	-	-	1	1
		0001–1111	Reserved				-	-	-	-		
	691	00000001	ETC_3	ETIMER_0	ETMR0 input channel 3	I	-	-	-	-		
	697	00000100	ETC_3	ETIMER_1	ETMR1 input channel 3	I	-	-	-	-		
	—	—	ANP[6]	SAR_ADC	analog channel	I	-	-	-	-		

Table 6. Pin multiplexing description(Continued)

Port pin	SIUL2 MSCR		Signal	Module	Full signal name	Direction	Strength				GPIO_MAPPED	MSCR_MAPPED
	No.	SSS value					VeryStrong	Strong	Medium	Weak		
PB[7]	23	0000	GPIO23	SIUL2-GPIO23	GPIO23	I	-	-	-	-	1	1
		0001–1111	Reserved				-	-	-	-		
	692	00000100	ETC_4	ETIMER_0	ETMR0 input channel 4	I	-	-	-	-		
	698	00000110	ETC_4	ETIMER_1	ETMR1 input channel 4	I	-	-	-	-		
	816	00000001	REQ0	SIUL	interrupt request channel 0	I	-	-	-	-		
	—	—	ANP[5]	SAR_ADC	analog channel	I	-	-	-	-		
PB[8]	24	0000	GPIO24	SIUL2-GPIO24	GPIO24	I	-	-	-	-	1	1
		0001–1111	Reserved				-	-	-	-		
	513	00000101	RXD	FlexCAN_1	FlexCAN 1 Receive Data	I	-	-	-	-		
	693	00000001	ETC_5	ETIMER_0	ETMR0 input channel 5	I	-	-	-	-		
	820	00000010	REQ4	SIUL	interrupt request channel 2	I	-	-	-	-		
	—	—	ANP[14]	SAR_ADC	analog channel	I	-	-	-	-		
PB[9]	25	0000	GPIO25	SIUL2-GPIO25	GPIO25	I	-	-	-	-	1	1
		0001–1111	Reserved				-	-	-	-		
	560	00000010	RXD	LINFlexD_0	LINFlexD 0 Receive Data	I	-	-	-	-		
	703	00000001	ETC_3	ETIMER_2	ETMR2 input channel 3	I	-	-	-	-		
	821	00000010	REQ5	SIUL	interrupt request channel 2	I	-	-	-	-		
	—	—	ANP[13]	SAR_ADC	analog channel	I	-	-	-	-		
PB[10]	26	0000	GPIO26	SIUL2-GPIO26	GPIO26	I	-	-	-	-	1	1
		0001–1111	Reserved				-	-	-	-		
	693	00000100	ETC_5	ETIMER_0	ETMR0 input channel 5	I	-	-	-	-		
	699	00000110	ETC_5	ETIMER_1	ETMR1 input channel 5	I	-	-	-	-		
	817	00000001	REQ1	SIUL	interrupt request channel 1	I	-	-	-	-		
	—	—	ANP[4]	SAR_ADC	analog channel	I	-	-	-	-		

Table 6. Pin multiplexing description(Continued)

Port pin	SIUL2 MSCR		Signal	Module	Full signal name	Direction	Strength				GPIO_MAPPED	MSCR_MAPPED
	No.	SSS value					VeryStrong	Strong	Medium	Weak		
PB[11]	27	0000	GPIO27	SIUL2-GPIO27	GPIO27	I	-	-	-	-	1	1
		0001–1111	Reserved				-	-	-	-		
	692	00000110	ETC_4	ETIMER_0	ETMR0 input channel 4	I	-	-	-	-		
	698	00000100	ETC_4	ETIMER_1	ETMR1 input channel 4	I	-	-	-	-		
	818	00000001	REQ2	SIUL	interrupt request channel 2	I	-	-	-	-		
	—	—	ANP[3]	SAR_ADC	analog channel	I	-	-	-	-		
PB[12]	28	0000	GPIO28	SIUL2-GPIO28	GPIO28	I	-	-	-	-	1	1
		0001–1111	Reserved				-	-	-	-		
	561	00000100	RXD	LINFlexD_1	LINFlexD 1 Receive Data	I	-	-	-	-		
	699	00000100	ETC_5	ETIMER_1	ETMR1 input channel 5	I	-	-	-	-		
	704	00000001	ETC_4	ETIMER_2	ETMR2 input channel 4	I	-	-	-	-		
	—	—	ANP[12]	SAR_ADC	analog channel	I	-	-	-	-		
PB[13]	29	0000	GPIO29	SIUL2-GPIO29	GPIO29	I	-	-	-	-	1	1
		0001–1111	Reserved				-	-	-	-		
	705	00000001	ETC_5	ETIMER_2	ETMR2 input channel 5	I	-	-	-	-		
	706	00000100	ETC_0	ETIMER_3	ETMR3 input channel 0	I	-	-	-	-		
	944	00000011	NMI	WKPU	NMI input	I	-	-	-	-		
	—	—	ANP[11]	SAR_ADC	analog channel	I	-	-	-	-		
PB[14]	30	0000	GPIO30	SIUL2-GPIO30	GPIO30	I	-	-	-	-	1	1
		0001–1111	Reserved				-	-	-	-		
	700	00000001	ETC_0	ETIMER_2	ETMR2 input channel 0	I	-	-	-	-		
	701	00000101	ETC_1	ETIMER_2	ETMR2 input channel 1	I	-	-	-	-		
	707	00000100	ETC_1	ETIMER_3	ETMR3 input channel 1	I	-	-	-	-		
	—	—	ANP[2]	SAR_ADC	analog channel	I	-	-	-	-		

Table 6. Pin multiplexing description(Continued)

Port pin	SIUL2 MSCR		Signal	Module	Full signal name	Direction	Strength				GPIO_MAPPED	MSCR_MAPPED
	No.	SSS value					VeryStrong	Strong	Medium	Weak		
PB[15]	31	0000	GPIO31	SIUL2-GPIO31	GPIO31	I	-	-	-	-	-	1
		0001–1111	Reserved				-	-	-	-	-	
	701	00000001	ETC_1	ETIMER_2	ETMR2 input channel 1	I	-	-	-	-	-	
	702	00000101	ETC_2	ETIMER_2	ETMR2 input channel 2	I	-	-	-	-	-	
	708	00000100	ETC_2	ETIMER_3	ETMR3 input channel 2	I	-	-	-	-	-	
	—	—	ANP[1]	SAR_ADC	analog channel	I	-	-	-	-	-	
PC[0]	32	0000	GPIO32	SIUL2-GPIO32	GPIO32	I	-	-	-	-	-	1
		0001–1111	Reserved				-	-	-	-	-	
	694	00000001	ETC_0	ETIMER_1	ETMR1 input channel 0	I	-	-	-	-	-	
	709	00000100	ETC_3	ETIMER_3	ETMR3 input channel 3	I	-	-	-	-	-	
	816	00000100	REQ0	SIUL	interrupt request channel 0	I	-	-	-	-	-	
	—	—	ANP[10]	SAR_ADC	analog channel	I	-	-	-	-	-	
PC[1]	33	0000	GPIO33	SIUL2-GPIO33	GPIO33	I	-	-	-	-	-	1
		0001–1111	Reserved				-	-	-	-	-	
	702	00000001	ETC_2	ETIMER_2	ETMR2 input channel 2	I	-	-	-	-	-	
	704	00000101	ETC_4	ETIMER_2	ETMR2 input channel 4	I	-	-	-	-	-	
	710	00000100	ETC_4	ETIMER_3	ETMR3 input channel 4	I	-	-	-	-	-	
	—	—	ANP[0]	SAR_ADC	analog channel	I	-	-	-	-	-	
PC[2]	34	0000	GPIO34	SIUL2-GPIO34	GPIO34	I/O	-	-	x	x	1	1
		0001	ETC_5	ETIMER_0	Timer 0 - ch. 5 (IC/OC)	I/O	-	-	x	x		
		0010	CS1	DSPI_2	DSPI 2 - CS 1 (Master)	O	-	-	x	x		
		0011–1111	Reserved				-	-	x	x		
	512	00000011	RXD	FlexCAN_0	FlexCAN 0 Receive Data	I	-	-	x	x		
	513	00000011	RXD	FlexCAN_1	FlexCAN 1 Receive Data	I	-	-	x	x		
	693	00000010	ETC_5	ETIMER_0	ETMR0 input channel 5	I	-	-	x	x		

Table 6. Pin multiplexing description(Continued)

Port pin	SIUL2 MSCR		Signal	Module	Full signal name	Direction	Strength				GPIO_MAPPED	MSCR_MAPPED
	No.	SSS value					VeryStrong	Strong	Medium	Weak		
PC[3]	35	0000	GPIO35	SIUL2-GPIO35	GPIO35	I/O	-	-	x	x	1	1
		0001	ETC_0	ETIMER_1	Timer 1 - ch. 0 (IC/OC)	I/O	-	-	x	x		
		0010	CS2	DSPI_2	DSPI 2 - CS 2 (Master)	O	-	-	x	x		
		0011	TXD	FlexCAN_1	FlexCAN 1 - TX	O	-	-	x	x		
		0100	TXD	FlexCAN_0	FlexCAN 0 - TX	O	-	-	x	x		
		0101–1111	Reserved				-	-	x	x		
	694	00000010	ETC_0	ETIMER_1	ETMR1 input channel 0	I	-	-	x	x		
PC[4]	36	0000	GPIO36	SIUL2-GPIO36	GPIO36	I/O	-	-	x	x	1	1
		0001	ETC_1	ETIMER_1	Timer 1 - ch. 1 (IC/OC)	I/O	-	-	x	x		
		0010	CS0	DSPI_1	DSPI 1 - CS 0 (Master/Slave)	I/O	-	-	x	x		
		0011–1111	Reserved				-	-	x	x		
	513	00000110	RXD	FlexCAN_1	FlexCAN 1 Receive Data	I	-	-	x	x		
	629	00000100	CS0	DSPI_1	DSPI1 Slave Select Input	I	-	-	x	x		
	695	00000001	ETC_1	ETIMER_1	ETMR1 input channel 1	I	-	-	x	x		
	817	00000011	REQ1	SIUL	interrupt request channel 1	I	-	-	x	x		

Table 6. Pin multiplexing description(Continued)

Port pin	SIUL2 MSCR		Signal	Module	Full signal name	Direction	Strength				GPIO_MAPPED	MSCR_MAPPED
	No.	SSS value					VeryStrong	Strong	Medium	Weak		
PC[5]	37	0000	GPIO37	SIUL2-GPIO37	GPIO37	I/O	x	x	x	x	1	1
		0001	CS0	DSPI_1	DSPI 1 - CS 0 (Master/Slave)	I/O	x	x	x	x		
		0010	ETC_2	ETIMER_1	Timer 1 - ch. 2 (IC/OC)	I/O	x	x	x	x		
		0011	RDY_B	INTG_DEBUG	Nexus RDY	I/O	x	x	x	x		
		0100	TXD	FlexCAN_1	FlexCAN 1 - TX	O	x	x	x	x		
		0101–1111	Reserved				x	x	x	x		
	629	00000010	CS0	DSPI_1	DSPI1 Slave Select Input	I	x	x	x	x		
	696	00000011	ETC_2	ETIMER_1	ETMR1 input channel 2	I	x	x	x	x		
PC[6]	38	0000	GPIO38	SIUL2-GPIO38	GPIO38	I/O	-	-	x	x	1	1
		0001	SOUT	DSPI_1	DSPI 1 - Serial Data	I/O	-	-	x	x		
		0010	ETC_3	ETIMER_1	Timer 1 - ch. 3 (IC/OC)	I/O	-	-	x	x		
		0011	CS4	DSPI_2	DSPI 2 - CS 4 (Master)	O	-	-	x	x		
		0100	SOUT	DSPI_0	DSPI 0 - Serial Data	I/O	-	-	x	x		
		0101–1111	Reserved				-	-	x	x		
	624	00000101	SIN	DSPI_0	DSPI0 Serial Data Input	I	-	-	x	x		
	627	00000001	SIN	DSPI_1	DSPI1 Serial Data Input	I	-	-	x	x		
	697	00000011	ETC_3	ETIMER_1	ETMR1 input channel 3	I	-	-	x	x		

Table 6. Pin multiplexing description(Continued)

Port pin	SIUL2 MSCR		Signal	Module	Full signal name	Direction	Strength				GPIO_MAPPED	MSCR_MAPPED
	No.	SSS value					VeryStrong	Strong	Medium	Weak		
PC[7]	39	0000	GPIO39	SIUL2-GPIO39	GPIO39	I/O	x	x	x	x	1	1
		0001	ETC_2	ETIMER_1	Timer 1 - ch. 2 (IC/OC)	I/O	x	x	x	x		
		0010	SOUT	DSPI_1	DSPI 1 - Serial Data	I/O	x	x	x	x		
		0011	CS5	DSPI_2	DSPI 2 - CS 5 (Master)	O	x	x	x	x		
		0100	CS0	DSPI_0	DSPI 0 - CS 0 (Master/Slave)	I/O	x	x	x	x		
		0101–1111	Reserved				x	x	x	x		
	626	00000110	CS0	DSPI_0	DSPI0 Slave Select Input	I	x	x	x	x		
	627	00000100	SIN	DSPI_1	DSPI1 Serial Data Input	I	x	x	x	x		
	696	00000001	ETC_2	ETIMER_1	ETMR1 input channel 2	I	x	x	x	x		
PC[8]	40	0000	GPIO40	SIUL2-GPIO40	GPIO40	I/O	x	x	x	x	1	1
		0001	ETC_3	ETIMER_1	Timer 1 - ch. 3 (IC/OC)	I/O	x	x	x	x		
		0010	SOUT	DSPI_1	DSPI 1 - Serial Data	I/O	x	x	x	x		
		0011	CS6	DSPI_2	DSPI 2 - CS 6 (Master)	O	x	x	x	x		
		0100	CS1	DSPI_0	DSPI 0 - CS 1 (Master)	O	x	x	x	x		
		0101–1111	Reserved				x	x	x	x		
	627	00000011	SIN	DSPI_1	DSPI1 Serial Data Input	I	x	x	x	x		
	697	00000001	ETC_3	ETIMER_1	ETMR1 input channel 3	I	x	x	x	x		

Table 6. Pin multiplexing description(Continued)

Port pin	SIUL2 MSCR		Signal	Module	Full signal name	Direction	Strength				GPIO_MAPPED	MSCR_MAPPED
	No.	SSS value					VeryStrong	Strong	Medium	Weak		
PC[9]	41	0000	GPIO41	SIUL2-GPIO41	GPIO41	I/O	x	x	x	x	1	1
		0001	SOUT	DSPI_1	DSPI 1 - Serial Data	I/O	x	x	x	x		
		0010	ETC_4	ETIMER_1	Timer 1 - ch. 4 (IC/OC)	I/O	x	x	x	x		
		0011	CS7	DSPI_2	DSPI 2 - CS 7 (Master)	O	x	x	x	x		
		0100	SOUT	DSPI_0	DSPI 0 - Serial Data	I/O	x	x	x	x		
		0101–1111	Reserved				x	x	x	x		
	624	00000110	SIN	DSPI_0	DSPI0 Serial Data Input	I	x	x	x	x		
	627	00000010	SIN	DSPI_1	DSPI1 Serial Data Input	I	x	x	x	x		
	698	00000011	ETC_4	ETIMER_1	ETMR1 input channel 4	I	x	x	x	x		
PC[10]	42	0000	GPIO42	SIUL2-GPIO42	GPIO42	I/O	x	x	x	x	1	1
		0001	SCK	DSPI_1	DSPI 1 - CLK (Master/Slave)	I/O	x	x	x	x		
		0010	ETC_5	ETIMER_1	Timer 1 - ch. 5 (IC/OC)	I/O	x	x	x	x		
		0011	Reserved				x	x	x	x		
		0100	SCK	DSPI_0	DSPI 0 - CLK (Master/Slave)	I/O	x	x	x	x		
		0101–1111	Reserved				x	x	x	x		
	625	00000110	SCK	DSPI_0	DSPI0 Serial Clock Input	I	x	x	x	x		
	628	00000010	SCK	DSPI_1	DSPI1 Serial Clock Input	I	x	x	x	x		
	699	00000011	ETC_5	ETIMER_1	ETMR1 input channel 5	I	x	x	x	x		

Table 6. Pin multiplexing description(Continued)

Port pin	SIUL2 MSCR		Signal	Module	Full signal name	Direction	Strength				GPIO_MAPPED	MSCR_MAPPED
	No.	SSS value					VeryStrong	Strong	Medium	Weak		
PC[11]	43	0000	GPIO43	SIUL2-GPIO43	GPIO43	I/O	x	x	x	x	1	1
		0001	ETC_4	ETIMER_1	Timer 1 - ch. 4 (IC/OC)	I/O	x	x	x	x		
		0010	SCK	DSPI_1	DSPI 1 - CLK (Master/Slave)	I/O	x	x	x	x		
		0011	Reserved				x	x	x	x		
		0100	CS2	DSPI_0	DSPI 0 - CS 2 (Master)	O	x	x	x	x		
		0101–1111	Reserved				x	x	x	x		
	628	00000100	SCK	DSPI_1	DSPI1 Serial Clock Input	I	x	x	x	x		
	698	00000001	ETC_4	ETIMER_1	ETMR1 input channel 4	I	x	x	x	x		
PC[12]	44	0000	GPIO44	SIUL2-GPIO44	GPIO44	I/O	-	-	x	x	1	1
		0001	ETC_0	ETIMER_0	Timer 0 - ch. 0 (IC/OC)	I/O	-	-	x	x		
		0010	CS3	DSPI_1	DSPI 1 - CS 3 (Master)	O	-	-	x	x		
		0011–1111	Reserved				-	-	x	x		
	560	00000011	RXD	LINFlexD_0	LINFlexD 0 Receive Data	I	-	-	x	x		
	688	00000010	ETC_0	ETIMER_0	ETMR0 input channel 0	I	-	-	x	x		
PC[13]	45	0000	GPIO45	SIUL2-GPIO45	GPIO45	I/O	-	-	x	x	1	1
		0001	ETC_1	ETIMER_0	Timer 0 - ch. 1 (IC/OC)	I/O	-	-	x	x		
		0010	CS4	DSPI_1	DSPI 1 - CS 4 (Master)	O	-	-	x	x		
		0011	Reserved				-	-	x	x		
		0100	TXD	LINFlexD_0	LINFlexD 0 - TX	O	-	-	x	x		
		0101–1111	Reserved				-	-	x	x		
	689	00000010	ETC_1	ETIMER_0	ETMR0 input channel 1	I	-	-	x	x		

Table 6. Pin multiplexing description(Continued)

Port pin	SIUL2 MSCR		Signal	Module	Full signal name	Direction	Strength				GPIO_MAPPED	MSCR_MAPPED
	No.	SSS value					VeryStrong	Strong	Medium	Weak		
PC[14]	46	0000	GPIO46	SIUL2-GPIO46	GPIO46	I/O	-	-	x	x	1	1
		0001	ETC_2	ETIMER_0	Timer 0 - ch. 2 (IC/OC)	I/O	-	-	x	x		
		0010	CS5	DSPI_1	DSPI 1 - CS 5 (Master)	O	-	-	x	x		
		0011	Reserved				-	-	x	x		
		0100	CS3	DSPI_0	DSPI 0 - CS 3 (Master)	O	-	-	x	x		
		0101–1111	Reserved				-	-	x	x		
	690	00000010	ETC_2	ETIMER_0	ETMR0 input channel 2	I	-	-	x	x		
PC[15]	47	0000	GPIO47	SIUL2-GPIO47	GPIO47	I/O	-	-	x	x	1	1
		0001	Reserved				-	-	x	x		
		0010	CS2	DSPI_1	DSPI 1 - CS 2 (Master)	O	-	-	x	x		
		0011	Reserved				-	-	x	x		
		0100	ETC_0	ETIMER_2	Timer 2 - ch. 0 (IC/OC)	I/O	-	-	x	x		
		0101	ESR0	MC_RGM	—	O	-	-	x	x		
		0110–1111	Reserved				-	-	x	x		
	700	00000110	ETC_0	ETIMER_2	ETMR2 input channel 0	I	-	-	x	x		
	820	00000011	REQ4	SIUL	interrupt request channel 2	I	-	-	x	x		
	944	00000001	NMI	WKPU	NMI input	I	-	-	x	x		

Table 6. Pin multiplexing description(Continued)

Port pin	SIUL2 MSCR		Signal	Module	Full signal name	Direction	Strength				GPIO_MAPPED	MSCR_MAPPED
	No.	SSS value					VeryStrong	Strong	Medium	Weak		
PD[0]	48	0000	GPIO48	SIUL2-GPIO48	GPIO48	I/O	-	-	x	x	1	1
		0001	CS6	DSPI_1	DSPI 1 - CS 6 (Master)	O	-	-	x	x		
		0010-0011	Reserved				-	-	x	x		
		0100	ETC_1	ETIMER_2	Timer 2 - ch. 1 (IC/OC)	I/O	-	-	x	x		
		0101–1111	Reserved				-	-	x	x		
	701	00000110	ETC_1	ETIMER_2	ETMR2 input channel 1	I	-	-	x	x		
	816	00000011	REQ0	SIUL	interrupt request channel 0	I	-	-	x	x		
PD[1]	49	0000	GPIO49	SIUL2-GPIO49	GPIO49	I/O	-	-	x	x	1	1
		0001	ETC_3	ETIMER_0	Timer 0 - ch. 3 (IC/OC)	I/O	-	-	x	x		
		0010	CS7	DSPI_1	DSPI 1 - CS 7 (Master)	O	-	-	x	x		
		0011	Reserved				-	-	x	x		
		0100	CS4	DSPI_0	DSPI 0 - CS 4 (Master)	O	-	-	x	x		
		0101–1111	Reserved				-	-	x	x		
	691	00000010	ETC_3	ETIMER_0	ETMR0 input channel 3	I	-	-	x	x		
PD[2]	50	0000	GPIO50	SIUL2-GPIO50	GPIO50	I/O	-	-	x	x	1	1
		0001	ETC_0	ETIMER_2	Timer 2 - ch. 0 (IC/OC)	I/O	-	-	x	x		
		0010	CS1	DSPI_2	DSPI 2 - CS 1 (Master)	O	-	-	x	x		
		0011	CS6	DSPI_1	DSPI 1 - CS 6 (Master)	O	-	-	x	x		
		0100	ETC_0	ETIMER_3	Timer 3 - ch. 0 (IC/OC)	I/O	-	-	x	x		
		0101–1111	Reserved				-	-	x	x		
	700	00000010	ETC_0	ETIMER_2	ETMR2 input channel 0	I	-	-	x	x		
	706	00000110	ETC_0	ETIMER_3	ETMR3 input channel 0	I	-	-	x	x		

Table 6. Pin multiplexing description(Continued)

Port pin	SIUL2 MSCR		Signal	Module	Full signal name	Direction	Strength				GPIO_MAPPED	MSCR_MAPPED
	No.	SSS value					VeryStrong	Strong	Medium	Weak		
PD[3]	51	0000	GPIO51	SIUL2-GPIO51	GPIO51	I/O	-	-	x	x	1	1
		0001	ETC_1	ETIMER_2	Timer 2 - ch. 1 (IC/OC)	I/O	-	-	x	x		
		0010	CS2	DSPI_2	DSPI 2 - CS 2 (Master)	O	-	-	x	x		
		0011	CS4	DSPI_1	DSPI 1 - CS 4 (Master)	O	-	-	x	x		
		0100	ETC_1	ETIMER_3	Timer 3 - ch. 1 (IC/OC)	I/O	-	-	x	x		
		0101–1111	Reserved				-	-	x	x		
	701	00000010	ETC_1	ETIMER_2	ETMR2 input channel 1	I	-	-	x	x		
	707	00000110	ETC_1	ETIMER_3	ETMR3 input channel 1	I	-	-	x	x		
PD[4]	52	0000	GPIO52	SIUL2-GPIO52	GPIO52	I/O	-	-	x	x	1	1
		0001	ETC_2	ETIMER_2	Timer 2 - ch. 2 (IC/OC)	I/O	-	-	x	x		
		0010	CS3	DSPI_2	DSPI 2 - CS 3 (Master)	O	-	-	x	x		
		0011	CS7	DSPI_1	DSPI 1 - CS 7 (Master)	O	-	-	x	x		
		0100	ETC_2	ETIMER_3	Timer 3 - ch. 2 (IC/OC)	I/O	-	-	x	x		
		0101–1111	Reserved				-	-	x	x		
	702	00000010	ETC_2	ETIMER_2	ETMR2 input channel 2	I	-	-	x	x		
	708	00000110	ETC_2	ETIMER_3	ETMR3 input channel 2	I	-	-	x	x		

Table 6. Pin multiplexing description(Continued)

Port pin	SIUL2 MSCR		Signal	Module	Full signal name	Direction	Strength				GPIO_MAPPED	MSCR_MAPPED
	No.	SSS value					VeryStrong	Strong	Medium	Weak		
PD[5]	53	0000	GPIO53	SIUL2-GPIO53	GPIO53	I/O	x	x	x	x	1	1
		0001	CS0	DSPI_2	DSPI2 - CS 0 (Master/Slave)	I/O	x	x	x	x		
		0010	ETC_1	ETIMER_2	Timer 2 - ch. 1 (IC/OC)	I/O	x	x	x	x		
		0011	CS6	DSPI_1	DSPI 1 - CS 6 (Master)	O	x	x	x	x		
		0100	ETC_3	ETIMER_3	Timer 3 - ch. 3 (IC/OC)	I/O	x	x	x	x		
		0101–1111	Reserved				x	x	x	x		
	632	00000010	CS0	DSPI_2	DSPI2 Slave Select Input	I	x	x	x	x		
	701	00000011	ETC_1	ETIMER_2	ETMR2 input channel 1	I	x	x	x	x		
	709	00000110	ETC_3	ETIMER_3	ETMR3 input channel 3	I	x	x	x	x		
PD[6]	54	0000	GPIO54	SIUL2-GPIO54	GPIO54	I/O	x	x	x	x	1	1
		0001	SOUT	DSPI_2	DSPI2 - Serial Data	I/O	x	x	x	x		
		0010	ETC_2	ETIMER_2	Timer 2 - ch. 2 (IC/OC)	I/O	x	x	x	x		
		0011	CS5	DSPI_1	DSPI 1 - CS 5 (Master)	O	x	x	x	x		
		0100	CS5	DSPI_0	DSPI 0 - CS 5 (Master)	O	x	x	x	x		
		0101–1111	Reserved				x	x	x	x		
	630	00000001	SIN	DSPI_2	DSPI2 Serial Data Input	I	x	x	x	x		
	702	00000011	ETC_2	ETIMER_2	ETMR2 input channel 2	I	x	x	x	x		

Table 6. Pin multiplexing description(Continued)

Port pin	SIUL2 MSCR		Signal	Module	Full signal name	Direction	Strength				GPIO_MAPPED	MSCR_MAPPED
	No.	SSS value					VeryStrong	Strong	Medium	Weak		
PD[7]	55	0000	GPIO55	SIUL2-GPIO55	GPIO55	I/O	-	-	x	x	1	1
		0001	ETC_0	ETIMER_3	Timer 3 - ch. 0 (IC/OC)	I/O	-	-	x	x		
		0010	Reserved				-	-	x	x		
		0011	CS2	DSPI_1	DSPI 1 - CS 2 (Master)	O	-	-	x	x		
		0100–1111	Reserved				-	-	x	x		
	561	00000101	RXD	LINFlexD_1	LINFlexD 1 Receive Data	I	-	-	x	x		
	706	00000001	ETC_0	ETIMER_3	ETMR3 input channel 0	I	-	-	x	x		
	767	00000010	EXT_IN15	CTU_0	CTU input trigger 15	I	-	-	x	x		
PD[8]	56	0000	GPIO56	SIUL2-GPIO56	GPIO56	I/O	-	-	x	x	1	1
		0001	ETC_1	ETIMER_3	Timer 3 - ch. 1 (IC/OC)	I/O	-	-	x	x		
		0010	EXT_TRG	CTU_0	CTU External Trigger	O	-	-	x	x		
		0011	CS6	DSPI_1	DSPI 1 - CS 6 (Master)	O	-	-	x	x		
		0100	TXD	LINFlexD_1	LINFlexD 1 - TX	O	-	-	x	x		
		0101–1111	Reserved				-	-	x	x		
	707	00000001	ETC_1	ETIMER_3	ETMR3 input channel 1	I	-	-	x	x		

Table 6. Pin multiplexing description(Continued)

Port pin	SIUL2 MSCR		Signal	Module	Full signal name	Direction	Strength				GPIO_MAPPED	MSCR_MAPPED
	No.	SSS value					VeryStrong	Strong	Medium	Weak		
PD[9]	57	0000	GPIO57	SIUL2-GPIO57	GPIO57	I/O	-	-	x	x	1	1
		0001	Reserved				-	-	x	x		
		0010	CS1	DSPI_2	DSPI 2 - CS 1 (Master)	O	-	-	x	x		
		0011	Reserved				-	-	x	x		
		0100	ETC_2	ETIMER_2	Timer 2 - ch. 2 (IC/OC)	I/O	-	-	x	x		
		0101–1111	Reserved				-	-	x	x		
	512	00000001	RXD	FlexCAN_0	FlexCAN 0 Receive Data	I	-	-	x	x		
	513	00000100	RXD	FlexCAN_1	FlexCAN 1 Receive Data	I	-	-	x	x		
	702	00000110	ETC_2	ETIMER_2	ETMR2 input channel 2	I	-	-	x	x		
PD[10]	58	0000	GPIO58	SIUL2-GPIO58	GPIO58	I/O	-	-	x	x	1	1
		0001	TXD	FlexCAN_0	FlexCAN 0 - TX	O	-	-	x	x		
		0010	Reserved				-	-	x	x		
		0011	TXD	FlexCAN_1	FlexCAN 1 - TX	O	-	-	x	x		
		0100	ETC_3	ETIMER_2	Timer 2 - ch. 3 (IC/OC)	I/O	-	-	x	x		
		0101–1111	Reserved				-	-	x	x		
	703	00000110	ETC_3	ETIMER_2	ETMR2 input channel 3	I	-	-	x	x		
PD[11]	59	0000	GPIO59	SIUL2-GPIO59	GPIO59	I/O	-	-	x	x	1	1
		0001	ETC_2	ETIMER_3	Timer 3 - ch. 2 (IC/OC)	I/O	-	-	x	x		
		0010	SCK	DSPI_2	DSPI 2 - CLK (Master/Slave)	I/O	-	-	x	x		
		0011	CS7	DSPI_1	DSPI 1 - CS 7 (Master)	O	-	-	x	x		
		0100–1111	Reserved				-	-	x	x		
	631	00000100	SCK	DSPI_2	DSPI2 Serial Clock Input	I	-	-	x	x		
	708	00000001	ETC_2	ETIMER_3	ETMR3 input channel 2	I	-	-	x	x		

Table 6. Pin multiplexing description(Continued)

Port pin	SIUL2 MSCR		Signal	Module	Full signal name	Direction	Strength				GPIO_MAPPED	MSCR_MAPPED
	No.	SSS value					VeryStrong	Strong	Medium	Weak		
PD[12]	60	0000	GPIO60	SIUL2-GPIO60	GPIO60	I/O	x	x	x	x	1	1
		0001	SOUT	DSPI_2	DSPI2 - Serial Data	I/O	x	x	x	x		
		0010	ETC_3	ETIMER_2	Timer 2 - ch. 3 (IC/OC)	I/O	x	x	x	x		
		0011	CS2	DSPI_2	DSPI 2 - CS 2 (Master)	O	x	x	x	x		
		0100–1111	Reserved				x	x	x	x		
	630	00000010	SIN	DSPI_2	DSPI2 Serial Data Input	I	x	x	x	x		
	703	00000011	ETC_3	ETIMER_2	ETMR2 input channel 3	I	x	x	x	x		
PD[13]	61	0000	GPIO61	SIUL2-GPIO61	GPIO61	I/O	x	x	x	x	1	1
		0001	SCK	DSPI_2	DSPI2 - CLK (Master/Slave)	I/O	x	x	x	x		
		0010	ETC_4	ETIMER_2	Timer 2 - ch. 4 (IC/OC)	I/O	x	x	x	x		
		0011	CS3	DSPI_2	DSPI 2 - CS 3 (Master)	O	x	x	x	x		
		0100–1111	Reserved				x	x	x	x		
	631	00000010	SCK	DSPI_2	DSPI2 Serial Clock Input	I	x	x	x	x		
	704	00000011	ETC_4	ETIMER_2	ETMR2 input channel 4	I	x	x	x	x		

Table 6. Pin multiplexing description(Continued)

Port pin	SIUL2 MSCR		Signal	Module	Full signal name	Direction	Strength				GPIO_MAPPED	MSCR_MAPPED
	No.	SSS value					VeryStrong	Strong	Medium	Weak		
PD[14]	62	0000	GPIO62	SIUL2-GPIO62	GPIO62	I/O	-	-	x	x	1	1
		0001	ETC_3	ETIMER_2	Timer 2 - ch. 3 (IC/OC)	I/O	-	-	x	x		
		0010	SOUT	DSPI_2	DSPI 2 - Serial Data	I/O	-	-	x	x		
		0011	ETC_3	ETIMER_3	Timer 3 - ch. 3 (IC/OC)	I/O	-	-	x	x		
		0100–1111	Reserved				-	-	x	x		
	630	00000011	SIN	DSPI_2	DSPI2 Serial Data Input	I	-	-	x	x		
	703	00000010	ETC_3	ETIMER_2	ETMR2 input channel 3	I	-	-	x	x		
	709	00000101	ETC_3	ETIMER_3	ETMR3 input channel 3	I	-	-	x	x		
PD[15]	63	0000	GPIO63	SIUL2-GPIO63	GPIO63	I/O	x	x	x	x	1	1
		0001	ETC_4	ETIMER_2	Timer 2 - ch. 4 (IC/OC)	I/O	x	x	x	x		
		0010	SOUT	DSPI_2	DSPI 2 - Serial Data	I/O	x	x	x	x		
		0011	ETC_4	ETIMER_3	Timer 3 - ch. 4 (IC/OC)	I/O	x	x	x	x		
		0100–1111	Reserved				x	x	x	x		
	630	00000100	SIN	DSPI_2	DSPI2 Serial Data Input	I	x	x	x	x		
	704	00000010	ETC_4	ETIMER_2	ETMR2 input channel 4	I	x	x	x	x		
	710	00000101	ETC_4	ETIMER_3	ETMR3 input channel 4	I	x	x	x	x		

Table 6. Pin multiplexing description(Continued)

Port pin	SIUL2 MSCR		Signal	Module	Full signal name	Direction	Strength				GPIO_MAPPED	MSCR_MAPPED
	No.	SSS value					VeryStrong	Strong	Medium	Weak		
PE[0]	64	0000	GPIO64	SIUL2-GPIO64	GPIO64	I/O	-	-	x	x	1	1
		0001	ETC_3	ETIMER_3	Timer 3 - ch. 3 (IC/OC)	I/O	-	-	x	x		
		0010-0011	Reserved				-	-	x	x		
		0100	ETC_4	ETIMER_2	Timer 2 - ch. 4 (IC/OC)	I/O	-	-	x	x		
		0101–1111	Reserved				-	-	x	x		
	704	00000110	ETC_4	ETIMER_2	ETMR2 input channel 4	I	-	-	x	x		
	709	00000001	ETC_3	ETIMER_3	ETMR3 input channel 3	I	-	-	x	x		
	818	00000010	REQ2	SIUL	interrupt request channel 2	I	-	-	x	x		
PE[1]	65	0000	GPIO65	SIUL2-GPIO65	GPIO65	I/O	-	-	x	x	1	1
		0001	ETC_4	ETIMER_3	Timer 3 - ch. 4 (IC/OC)	I/O	-	-	x	x		
		0010-0011	Reserved				-	-	x	x		
		0100	ETC_5	ETIMER_2	Timer 2 - ch. 5 (IC/OC)	I/O	-	-	x	x		
		0101–1111	Reserved				-	-	x	x		
	705	00000110	ETC_5	ETIMER_2	ETMR2 input channel 5	I	-	-	x	x		
	710	00000001	ETC_4	ETIMER_3	ETMR3 input channel 4	I	-	-	x	x		

Table 6. Pin multiplexing description(Continued)

Port pin	SIUL2 MSCR		Signal	Module	Full signal name	Direction	Strength				GPIO_MAPPED	MSCR_MAPPED
	No.	SSS value					VeryStrong	Strong	Medium	Weak		
PE[2]	66	0000	GPIO66	SIUL2-GPIO66	GPIO66	I/O	x	x	x	x	1	1
		0001	ETC_5	ETIMER_2	Timer 2 - ch. 5 (IC/OC)	I/O	x	x	x	x		
		0010	CS0	DSPI_2	DSPI 2 - CS 0 (Master/Slave)	I/O	x	x	x	x		
		0011	CS3	DSPI_0	DSPI 0 - CS 3 (Master)	O	x	x	x	x		
		0100–1111	Reserved				x	x	x	x		
	632	00000100	CS0	DSPI_2	DSPI2 Slave Select Input	I	x	x	x	x		
	705	00000010	ETC_5	ETIMER_2	ETMR2 input channel 5	I	x	x	x	x		
PE[3]	67	0000	GPIO67	SIUL2-GPIO67	GPIO67	I/O	x	x	x	x	1	1
		0001	MSEO_B_0	NPC_WRAPPER	Nexus MSEO	O	x	x	x	x		
		0010	Reserved				x	x	x	x		
		0011	CS4	DSPI_0	DSPI 0 - CS 4 (Master)	O	x	x	x	x		
		0100	SCK	DSPI_2	DSPI 2 - CLK (Master/Slave)	I/O	x	x	x	x		
		0101–1111	Reserved				x	x	x	x		
	631	00000110	SCK	DSPI_2	DSPI2 Serial Clock Input	I	x	x	x	x		

Table 6. Pin multiplexing description(Continued)

Port pin	SIUL2 MSCR		Signal	Module	Full signal name	Direction	Strength				GPIO_MAPPED	MSCR_MAPPED
	No.	SSS value					VeryStrong	Strong	Medium	Weak		
PE[4]	68	0000	GPIO68	SIUL2-GPIO68	GPIO68	I/O	-	-	x	x	1	1
		0001	ETC_5	ETIMER_3	Timer 3 - ch. 5 (IC/OC)	—	-	-	x	x		
		0010	CS2	DSPI_2	DSPI 2 - CS 2 (Master/Slave)	O	-	-	x	x		
		0011	ETC_4	ETIMER_2	Timer 2 - ch. 4 (IC/OC)	I/O	-	-	x	x		
		0100	CS0	DSPI_2	DSPI 2 - CS 0 (Master/Slave)	I/O	-	-	x	x		
		0101–1111	Reserved				-	-	x	x		
	632	00000110	CS0	DSPI_2	DSPI2 Slave Select Input	I	-	-	x	x		
	704	00000100	ETC_4	ETIMER_2	ETMR2 input channel 4	I	-	-	x	x		
	711	00000001	ETC_5	ETIMER_3	ETMR3 input channel 5	I	-	-	x	x		
PE[5]	69	0000	GPIO69	SIUL2-GPIO69	GPIO69	I/O	x	x	x	x	1	1
		0001	MDO_3	NPC_WRAPPER	Nexus MDO 3	O	x	x	x	x		
		0010	Reserved				x	x	x	x		
		0011	CLKOUT0	MC_CGL	CLOCKOUT	O	x	x	x	x		
		0100	SOUT	DSPI_2	DSPI 2 - Serial Data	I/O	x	x	x	x		
		0101–1111	Reserved				x	x	x	x		
	630	00000101	SIN	DSPI_2	DSPI2 Serial Data Input	I	x	x	x	x		

Table 6. Pin multiplexing description(Continued)

Port pin	SIUL2 MSCR		Signal	Module	Full signal name	Direction	Strength				GPIO_MAPPED	MSCR_MAPPED
	No.	SSS value					VeryStrong	Strong	Medium	Weak		
PE[6]	70	0000	GPIO70	SIUL2-GPIO70	GPIO70	I/O	x	x	x	x	1	1
		0001	MDO_2	NPC_WRAPPER	Nexus MDO 2	O	x	x	x	x		
		0010	Reserved				x	x	x	x		
		0011	CS6	DSPI_0	DSPI 0 - CS 6 (Master)	O	x	x	x	x		
		0100	SOUT	DSPI_2	DSPI 2 - Serial Data	I/O	x	x	x	x		
		0101–1111	Reserved				x	x	x	x		
	630	00000110	SIN	DSPI_2	DSPI2 Serial Data Input	I	x	x	x	x		
PE[7]	71	0000	GPIO71	SIUL2-GPIO71	GPIO71	I/O	x	x	x	x	1	1
		0001	MDO_1	NPC_WRAPPER	Nexus MDO 1	O	x	x	x	x		
		0010	Reserved				x	x	x	x		
		0011	CS7	DSPI_0	DSPI 0 - CS 7 (Master)	O	x	x	x	x		
		0100	ETC_4	ETIMER_3	Timer 3 - ch. 4 (IC/OC)	I/O	x	x	x	x		
		0101–1111	Reserved				x	x	x	x		
	710	00000110	ETC_4	ETIMER_3	ETMR3 input channel 4	I	x	x	x	x		

Table 6. Pin multiplexing description(Continued)

Port pin	SIUL2 MSCR		Signal	Module	Full signal name	Direction	Strength				GPIO_MAPPED	MSCR_MAPPED
	No.	SSS value					VeryStrong	Strong	Medium	Weak		
PE[8]	72	0000	GPIO72	SIUL2-GPIO72	GPIO72	I/O	x	x	x	x	1	1
		0001	MDO_0	NPC_WRAPPER	Nexus MDO 0	O	x	x	x	x		
		0010	CS0	DSPI_0	DSPI0 - CS 0 (Master/Slave)	I/O	x	x	x	x		
		0011	Reserved				x	x	x	x		
		0100	ETC_5	ETIMER_3	Timer 3 - ch. 5 (IC/OC)	I/O	x	x	x	x		
		0101–1111	Reserved				x	x	x	x		
	626	00000100	CS0	DSPI_0	DSPI0 Slave Select Input	I	x	x	x	x		
	711	00000110	ETC_5	ETIMER_3	ETMR3 input channel 5	I	x	x	x	x		
	819	00000100	REQ3	SIUL	interrupt request channel 2	I	x	x	x	x		
PE[9]	73	0000	GPIO73	SIUL2-GPIO73	GPIO73	I/O	x	x	x	x	1	1
		0001	Reserved				x	x	x	x		
		0010	ETC_2	ETIMER_3	Timer 3 - ch. 2 (IC/OC)	I/O	x	x	x	x		
		0011	Reserved				x	x	x	x		
		0100	CS1	DSPI_2	DSPI 2 - CS 1 (Master)	O	x	x	x	x		
		0101–1111	Reserved				x	x	x	x		
	708	00000011	ETC_2	ETIMER_3	ETMR3 input channel 2	I	x	x	x	x		
	820	00000100	REQ4	SIUL	interrupt request channel 2	I	x	x	x	x		

Table 6. Pin multiplexing description(Continued)

Port pin	SIUL2 MSCR		Signal	Module	Full signal name	Direction	Strength				GPIO_MAPPED	MSCR_MAPPED
	No.	SSS value					VeryStrong	Strong	Medium	Weak		
PE[10]	74	0000	GPIO74	SIUL2-GPIO74	GPIO74	I/O	x	x	x	x	1	1
		0001	Reserved				x	x	x	x		
		0010	ETC_3	ETIMER_3	Timer 3 - ch. 3 (IC/OC)	I/O	x	x	x	x		
		0011	CS5	DSPI_0	DSPI 0 - CS 5 (Master)	O	x	x	x	x		
		0100	CS2	DSPI_2	DSPI 2 - CS 2 (Master)	O	x	x	x	x		
		0101–1111	Reserved				x	x	x	x		
	709	00000011	ETC_3	ETIMER_3	ETMR3 input channel 3	I	x	x	x	x		
	821	00000100	REQ5	SIUL	interrupt request channel 2	I	x	x	x	x		
PE[11]	75	0000	GPIO75	SIUL2-GPIO75	GPIO75	I/O	x	x	x	x	1	1
		0001	MCKO	NPC_WRAPPER	Nexus MCK0	O	x	x	x	x		
		0010	SCK	DSPI_0	—	I/O	x	x	x	x		
		0011	CS1	DSPI_0	DSPI 0 - CS 1 (Master)	O	x	x	x	x		
		0100	CS3	DSPI_1	DSPI 1 - CS 3 (Master)	O	x	x	x	x		
		0101–1111	Reserved				x	x	x	x		
	625	00000100	SCK	DSPI_0	DSPI0 Serial Clock Input	I	x	x	x	x		

Table 6. Pin multiplexing description(Continued)

Port pin	SIUL2 MSCR		Signal	Module	Full signal name	Direction	Strength				GPIO_MAPPED	MSCR_MAPPED
	No.	SSS value					VeryStrong	Strong	Medium	Weak		
PE[12]	76	0000	GPIO76	SIUL2-GPIO76	GPIO76	I/O	-	-	x	x	1	1
		0001	Reserved				-	-	x	x		
		0010	ETC_4	ETIMER_3	Timer 3 - ch. 4 (IC/OC)	I/O	-	-	x	x		
		0011	CS0	DSPI_2	DSPI 2 - CS 0 (Master/Slave)	I/O	-	-	x	x		
		0100	CS2	DSPI_1	DSPI 1 - CS 2 (Master)	O	-	-	x	x		
		0101–1111	Reserved				-	-	x	x		
	632	00000101	CS0	DSPI_2	DSPI2 Slave Select Input	I	-	-	x	x		
	710	00000011	ETC_4	ETIMER_3	ETMR3 input channel 4	I	-	-	x	x		
PE[13]	77	0000	GPIO77	SIUL2-GPIO77	GPIO77	I/O	-	-	x	x	1	1
		0001	Reserved				-	-	x	x		
		0010	ETC_5	ETIMER_3	Timer 3 - ch. 5 (IC/OC)	I/O	-	-	x	x		
		0011	CS1	DSPI_2	DSPI 2 - CS 1 (Master)	O	-	-	x	x		
		0100	CS1	DSPI_1	DSPI 1 - CS 1 (Master)	O	-	-	x	x		
		0101–1111	Reserved				-	-	x	x		
	711	00000011	ETC_5	ETIMER_3	ETMR3 input channel 5	I	-	-	x	x		
PE[14]	78	0000	GPIO78	SIUL2-GPIO78	GPIO78	I/O	x	x	x	x	1	1
		0001	EVTO_B0	DCI	Nexus EVTO	O	x	x	x	x		
		0010	SOUT	DSPI_0	DSPI 0 - Serial Data	I/O	x	x	x	x		
		0011	CS2	DSPI_0	DSPI 0 - CS 2 (Master)	O	x	x	x	x		
		0100	CS3	DSPI_2	DSPI 2 - CS 3 (Master)	O	x	x	x	x		
		0101–1111	Reserved				x	x	x	x		
	624	00000011	SIN	DSPI_0	DSPI0 Serial Data Input	I	x	x	x	x		

Table 6. Pin multiplexing description(Continued)

Port pin	SIUL2 MSCR		Signal	Module	Full signal name	Direction	Strength				GPIO_MAPPED	MSCR_MAPPED
	No.	SSS value					VeryStrong	Strong	Medium	Weak		
PE[15]	79	0000	GPIO79	SIUL2-GPIO79	GPIO79	I/O	x	x	x	x	1	1
		0001	IOMUX_EVTI_B01	INTG_DEBUG	Nexus EVTI	I	x	x	x	x		
		0010	SOUT	DSPI_0	DSPI 0 - Serial Data	I/O	x	x	x	x		
		0011	CS3	DSPI_1	DSPI 1 - CS 3 (Master)	O	x	x	x	x		
		0100–1111	Reserved				x	x	x	x		
	624	00000100	SIN	DSPI_0	DSPI0 Serial Data Input	I	x	x	x	x		
PAD_FCCU_F0	480	0000	F0	FCCU	FCCU_F0	I/O	x	x	x	x	1	1
		0001–1111	Reserved				x	x	x	x		
PAD_FCCU_F1	481	0000	F1	FCCU	FCCU_F1	I/O	-	-	x	x	1	1
		0001–1111	Reserved				-	-	x	x		
PAD_TDI	482	0000	TDI	DCI	TDI	I	-	-	x	x	1	1
		0001–1111	Reserved				-	-	x	x		
PAD_TMS	483	0000	TMS	DCI	TMS	I	-	-	x	x	1	1
		0001–1111	Reserved				-	-	x	x		
PAD_TDO	484	0000	TDO	DCI	TDO	I/O	x	x	x	x	1	1
		0001–1111	Reserved				x	x	x	x		
PAD_TCK	485	0000	TCK	MC_CGL	TCK	I	-	-	x	x	1	1
		0001–1111	Reserved				-	-	x	x		
PAD_RDY_B	486	0000	PAD_RDY_B	INTG_DEBUG	Nexus RDY	I/O	x	x	x	x	1	1
		0001–1111	Reserved				x	x	x	x		

Table 6. Pin multiplexing description(Continued)

Port pin	SIUL2 MSCR		Signal	Module	Full signal name	Direction	Strength				GPIO_MAPPED	MSCR_MAPPED
	No.	SSS value					VeryStrong	Strong	Medium	Weak		
PAD_MSEO_B[0]	487	0000	PAD_MSEO_B_0	NPC_WRAPPER	Nexus MSEO	I/O	x	x	x	x	1	1
		0001–1111	Reserved				x	x	x	x		
PAD_MDO[3]	488	0000	PAD_MDO_3	NPC_WRAPPER	Nexus MDO 3	I/O	x	x	x	x	1	1
		0001–1111	Reserved				x	x	x	x		
PAD_MDO[2]	489	0000	PAD_MDO_2	NPC_WRAPPER	Nexus MDO 2	I/O	x	x	x	x	1	1
		0001–1111	Reserved				x	x	x	x		
PAD_MDO[1]	490	0000	PAD_MDO_1	NPC_WRAPPER	Nexus MDO 1	I/O	x	x	x	x	1	1
		0001–1111	Reserved				x	x	x	x		
PAD_MDO[0]	491	0000	PAD_MDO_0	NPC_WRAPPER	Nexus MDO 0	I/O	x	x	x	x	1	1
		0001–1111	Reserved				x	x	x	x		
PAD_MCKO	492	0000	PAD_MCKO	NPC_WRAPPER	Nexus MCKO	I/O	x	x	x	x	1	1
		0001–1111	Reserved				x	x	x	x		
PAD_EVTO_B	493	0000	PAD_EVTO_B	DCI	Nexus EVTO	I/O	x	x	x	x	1	1
		0001–1111	Reserved				x	x	x	x		
PAD_EVTI_B	494	0000	IOMUX_EVTI_B02	INTG_DEBUG	Nexus EVTI	I/O	-	-	x	x	1	1
		0001–1111	Reserved				-	-	x	x		

Table 7. Peripheral multiplexing description

Destination			MSCR	Alternate functions	Source		GPIO_MAPPED	MSCR_MAPPED
Instance	Peripherals	Functions			Peripherals	Functions		
FlexCAN_0	FlexCAN	RXD	MSCR[512]	0000_0000	—	disable-low	0	1
				0000_0001	I/O pad	PD[9]		
				0000_0010	I/O pad	PB[2]		
				0000_0011	I/O pad	PC[2]		
				0000_0100–1111_1111	—	Reserved		
FlexCAN_1	FlexCAN	RXD	MSCR[513]	0000_0000	—	disable-low	0	1
				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_0111–1111_1111	—	Reserved		
LINFlexD_0	LINFlexD3	RXD	MSCR[560]	0000_0000	—	disable-low	0	1
				0000_0001	I/O pad	PA[12]		
				0000_0010	I/O pad	PB[9]		
				0000_0011	I/O pad	PC[12]		
				0000_0100–1111_1111	—	Reserved		
LINFlexD_1	LINFlexD3	RXD	MSCR[561]	0000_0000	—	disable-low	0	1
				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_0110–1111_1111	—	Reserved		

Table 7. Peripheral multiplexing description(Continued)

Destination			MSCR	Alternate functions	Source		GPIO_MAPPED	MSCR_MAPPED
Instance	Peripherals	Functions			Peripherals	Functions		
DSPI_0	DSPI_IPI	SIN	MSCR[624]	0000_0000	—	disable-low	0	1
				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_1000–1111_1111	—	Reserved		
DSPI_0	DSPI_IPI	SCK	MSCR[625]	0000_0000	—	disable-low	0	1
				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_0111–1111_1111	—	Reserved		
DSPI_0	DSPI_IPI	CS0	MSCR[626]	0000_0000	—	disable-low	0	1
				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_0111–1111_1111	—	Reserved		

Table 7. Peripheral multiplexing description(Continued)

Destination			MSCR	Alternate functions	Source		GPIO_MAPPED	MSCR_MAPPED
Instance	Peripherals	Functions			Peripherals	Functions		
DSPI_1	DSPI_IPI	SIN	MSCR[627]	0000_0000	—	disable-low	0	1
				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_1000–1111_1111	—	Reserved		
DSPI_1	DSPI_IPI	SCK	MSCR[628]	0000_0000	—	disable-low	0	1
				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	0	1
				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_0111–1111_1111	—	Reserved		

Table 7. Peripheral multiplexing description(Continued)

Destination			MSCR	Alternate functions	Source		GPIO_MAPPED	MSCR_MAPPED
Instance	Peripherals	Functions			Peripherals	Functions		
DSPI_2	DSPI_IPI	SIN	MSCR[630]	0000_0000	—	disable-low	0	1
				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_1000–1111_1111	—	Reserved		
DSPI_2	DSPI_IPI	SCK	MSCR[631]	0000_0000	—	disable-low	0	1
				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_0111–1111_1111	—	Reserved		
DSPI_2	DSPI_IPI	CS0	MSCR[632]	0000_0000	—	disable-low	0	1
				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_0111–1111_1111	—	Reserved		

Table 7. Peripheral multiplexing description(Continued)

Destination			MSCR	Alternate functions	Source		GPIO_MAPPED	MSCR_MAPPED
Instance	Peripherals	Functions			Peripherals	Functions		
eTimer_0	ETIMER	ETC_0	MSCR[688]	0000_0000	—	disable-low	0	1
				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	ETIMER0_TRG		
				0000_1000–1111_1111	—	Reserved		
eTimer_0	ETIMER	ETC_1	MSCR[689]	0000_0000	—	disable-low	0	1
				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	ETIMER1_TRG		
				0000_1000–1111_1111	—	Reserved		
eTimer_0	ETIMER	ETC_2	MSCR[690]	0000_0000	—	disable-low	0	1
				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_0110–1111_1111	—	Reserved		

Table 7. Peripheral multiplexing description(Continued)

Destination			MSCR	Alternate functions	Source		GPIO_MAPPED	MSCR_MAPPED
Instance	Peripherals	Functions			Peripherals	Functions		
eTimer_0	ETIMER	ETC_3	MSCR[691]	0000_0000	—	disable-low	0	1
				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_0110–1111_1111	—	Reserved		
eTimer_0	ETIMER	ETC_4	MSCR[692]	0000_0000	—	disable-low	0	1
				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	ETC_5	MSCR[693]	0000_0000	—	disable-low	0	1
				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_0110–1111_1111	—	Reserved		

Table 7. Peripheral multiplexing description(Continued)

Destination			MSCR	Alternate functions	Source		GPIO_MAPPED	MSCR_MAPPED
Instance	Peripherals	Functions			Peripherals	Functions		
eTimer_1	ETIMER	ETC_0	MSCR[694]	0000_0000	—	disable-low	0	1
				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	ETIMER0_TRG		
				0000_1000–1111_1111	—	Reserved		
eTimer_1	ETIMER	ETC_1	MSCR[695]	0000_0000	—	disable-low	0	1
				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	ETIMER1_TRG		
				0000_1000–1111_1111	—	Reserved		
eTimer_1	ETIMER	ETC_2	MSCR[696]	0000_0000	—	disable-low	0	1
				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_0110–1111_1111	—	Reserved		

Table 7. Peripheral multiplexing description(Continued)

Destination			MSCR	Alternate functions	Source		GPIO_MAPPED	MSCR_MAPPED
Instance	Peripherals	Functions			Peripherals	Functions		
eTimer_1	ETIMER	ETC_3	MSCR[697]	0000_0000	—	disable-low	0	1
				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_0110–1111_1111	—	Reserved		
eTimer_1	ETIMER	ETC_4	MSCR[698]	0000_0000	—	disable-low	0	1
				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_0111–1111_1111	—	Reserved		
eTimer_1	ETIMER	ETC_5	MSCR[699]	0000_0000	—	disable-low	0	1
				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_0111–1111_1111	—	Reserved		

Table 7. Peripheral multiplexing description(Continued)

Destination			MSCR	Alternate functions	Source		GPIO_MAPPED	MSCR_MAPPED
Instance	Peripherals	Functions			Peripherals	Functions		
eTimer_2	ETIMER	ETC_0	MSCR[700]	0000_0000	—	disable-low	0	1
				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	ETIMER0_TRG		
				0000_1000–1111_1111	—	Reserved		
eTimer_2	ETIMER	ETC_1	MSCR[701]	0000_0000	—	disable-low	0	1
				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	ETIMER1_TRG		
				0000_1000–1111_1111	—	Reserved		
eTimer_2	ETIMER	ETC_2	MSCR[702]	0000_0000	—	disable-low	0	1
				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_0111–1111_1111	—	Reserved		

Table 7. Peripheral multiplexing description(Continued)

Destination			MSCR	Alternate functions	Source		GPIO_MAPPED	MSCR_MAPPED
Instance	Peripherals	Functions			Peripherals	Functions		
eTimer_2	ETIMER	ETC_3	MSCR[703]	0000_0000	—	disable-low	0	1
				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_0111–1111_1111	—	Reserved		
eTimer_2	ETIMER	ETC_4	MSCR[704]	0000_0000	—	disable-low	0	1
				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_0111–1111_1111	—	Reserved		
eTimer_2	ETIMER	ETC_5	MSCR[705]	0000_0000	—	disable-low	0	1
				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		

Table 7. Peripheral multiplexing description(Continued)

Destination			MSCR	Alternate functions	Source		GPIO_MAPPED	MSCR_MAPPED
Instance	Peripherals	Functions			Peripherals	Functions		
eTimer_3	ETIMER	ETC_0	MSCR[706]	0000_0000	—	disable-low	0	1
				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	ETIMER0_TRG		
				0000_1000–1111_1111	—	Reserved		
eTimer_3	ETIMER	ETC_1	MSCR[707]	0000_0000	—	disable-low	0	1
				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	ETIMER1_TRG		
				0000_1000–1111_1111	—	Reserved		
eTimer_3	ETIMER	ETC_2	MSCR[708]	0000_0000	—	disable-low	0	1
				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_0111–1111_1111	—	Reserved		

Table 7. Peripheral multiplexing description(Continued)

Destination			MSCR	Alternate functions	Source		GPIO_MAPPED	MSCR_MAPPED
Instance	Peripherals	Functions			Peripherals	Functions		
eTimer_3	ETIMER	ETC_3	MSCR[709]	0000_0000	—	disable-low	0	1
				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_0111–1111_1111	—	Reserved		
eTimer_3	ETIMER	ETC_4	MSCR[710]	0000_0000	—	disable-low	0	1
				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_0111–1111_1111	—	Reserved		
eTimer_3	ETIMER	ETC_5	MSCR[711]	0000_0000	—	disable-low	0	1
				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_0111–1111_1111	—	Reserved		
eTimer_0	ETIMER	FAULT_0	MSCR[712]	0000_0000	—	disable-high	0	1
				0000_0001	etimer_0	ETC_4 output		
				0000_0010-1111_1111	—	Reserved		

Table 7. Peripheral multiplexing description(Continued)

Destination			MSCR	Alternate functions	Source		GPIO_MAPPED	MSCR_MAPPED
Instance	Peripherals	Functions			Peripherals	Functions		
eTimer_0	ETIMER	FAULT_1	MSCR[713]	0000_0000	—	disable-high	0	1
				0000_0001	etimer_0	ETC_5 output		
				0000_0010-1111_1111	—	Reserved		
eTimer_1	ETIMER	FAULT_0	MSCR[714]	0000_0000	—	disable-high	0	1
				0000_0001	etimer_0	ETC_4 output		
				0000_0010-1111_1111	—	Reserved		
eTimer_1	ETIMER	FAULT_1	MSCR[715]	0000_0000	—	disable-high	0	1
				0000_0001	etimer_0	ETC_5 output		
				0000_0010-1111_1111	—	Reserved		
eTimer_2	ETIMER	FAULT_0	MSCR[716]	0000_0000	—	disable-high	0	1
				0000_0001	etimer_0	ETC_4 output		
				0000_0010-1111_1111	—	Reserved		
eTimer_2	ETIMER	FAULT_1	MSCR[717]	0000_0000	—	disable-high	0	1
				0000_0001	etimer_0	ETC_5 output		
				0000_0010-1111_1111	—	Reserved		
eTimer_3	ETIMER	FAULT_0	MSCR[718]	0000_0000	—	disable-high	0	1
				0000_0001	etimer_0	ETC_4 output		
				0000_0010-1111_1111	—	Reserved		
eTimer_3	ETIMER	FAULT_1	MSCR[719]	0000_0000	—	disable-high	0	1
				0000_0001	etimer_0	ETC_5 output		
				0000_0010-1111_1111	—	Reserved		

Table 7. Peripheral multiplexing description(Continued)

Destination			MSCR	Alternate functions	Source		GPIO_MAPPED	MSCR_MAPPED
Instance	Peripherals	Functions			Peripherals	Functions		
CTU_0	CTU	EXT_IN0	MSCR[752]	0000_0000	—	disable-low	0	1
				0000_0001	etimer_0	ETC_0		
				0000_0010	etimer_1	ETC_0		
				0000_0011	etimer_2	ETC_0		
				0000_0100	etimer_3	ETC_0		
				0000_0101–1111_1111	—	Reserved		
CTU_0	CTU	EXT_IN1	MSCR[753]	0000_0000	—	disable-low	0	1
				0000_0001	etimer_0	ETC_1		
				0000_0010	etimer_1	ETC_1		
				0000_0011	etimer_2	ETC_1		
				0000_0100	etimer_3	ETC_1		
				0000_0101–1111_1111	—	Reserved		
CTU_0	CTU	EXT_IN2	MSCR[754]	0000_0000	—	disable-low	0	1
				0000_0001	etimer_0	ETC_2		
				0000_0010	etimer_1	ETC_2		
				0000_0011	etimer_2	ETC_2		
				0000_0100	etimer_3	ETC_2		
				0000_0101–1111_1111	—	Reserved		
CTU_0	CTU	EXT_IN3	MSCR[755]	0000_0000	—	disable-low	0	1
				0000_0001	etimer_0	ETC_3		
				0000_0010	etimer_1	ETC_3		
				0000_0011	etimer_2	ETC_3		
				0000_0100	etimer_3	ETC_3		
				0000_0101–1111_1111	—	Reserved		

Table 7. Peripheral multiplexing description(Continued)

Destination			MSCR	Alternate functions	Source		GPIO_MAPPED	MSCR_MAPPED
Instance	Peripherals	Functions			Peripherals	Functions		
CTU_0	CTU	EXT_IN4	MSCR[756]	0000_0000	—	disable-low	0	1
				0000_0001	etimer_0	ETC_4		
				0000_0010	etimer_1	ETC_4		
				0000_0011	etimer_2	ETC_4		
				0000_0100	etimer_3	ETC_4		
				0000_0101–1111_1111	—	Reserved		
CTU_0	CTU	EXT_IN5	MSCR[757]	0000_0000	—	disable-low	0	1
				0000_0001	etimer_0	ETC_5		
				0000_0010	etimer_1	ETC_5		
				0000_0011	etimer_2	ETC_5		
				0000_0100	etimer_3	ETC_5		
				0000_0101–1111_1111	—	Reserved		
CTU_0	CTU	EXT_IN6	MSCR[758]	0000_0000	—	disable-low	0	1
				0000_0001	etimer_0	ETC_0		
				0000_0010	etimer_1	ETC_0		
				0000_0011	etimer_2	ETC_0		
				0000_0100	etimer_3	ETC_0		
				0000_0101–1111_1111	—	Reserved		
CTU_0	CTU	EXT_IN7	MSCR[759]	0000_0000	—	disable-low	0	1
				0000_0001	etimer_0	ETC_1		
				0000_0010	etimer_1	ETC_1		
				0000_0011	etimer_2	ETC_1		
				0000_0100	etimer_3	ETC_1		
				0000_0101–1111_1111	—	Reserved		

Table 7. Peripheral multiplexing description(Continued)

Destination			MSCR	Alternate functions	Source		GPIO_MAPPED	MSCR_MAPPED
Instance	Peripherals	Functions			Peripherals	Functions		
CTU_0	CTU	EXT_IN8	MSCR[760]	0000_0000	—	disable-low	0	1
				0000_0001	etimer_0	ETC_2		
				0000_0010	etimer_1	ETC_2		
				0000_0011	etimer_2	ETC_2		
				0000_0100	etimer_3	ETC_2		
				0000_0101–1111_1111	—	Reserved		
CTU_0	CTU	EXT_IN9	MSCR[761]	0000_0000	—	disable-low	0	1
				0000_0001	etimer_0	ETC_3		
				0000_0010	etimer_1	ETC_3		
				0000_0011	etimer_2	ETC_3		
				0000_0100	etimer_3	ETC_3		
				0000_0101–1111_1111	—	Reserved		
CTU_0	CTU	EXT_IN10	MSCR[762]	0000_0000	—	disable-low	0	1
				0000_0001	etimer_0	ETC_4		
				0000_0010	etimer_1	ETC_4		
				0000_0011	etimer_2	ETC_4		
				0000_0100	etimer_3	ETC_4		
				0000_0101–1111_1111	—	Reserved		
CTU_0	CTU	EXT_IN11	MSCR[763]	0000_0000	—	disable-low	0	1
				0000_0001	etimer_0	ETC_5		
				0000_0010	etimer_1	ETC_5		
				0000_0011	etimer_2	ETC_5		
				0000_0100	etimer_3	ETC_5		
				0000_0101–1111_1111	—	Reserved		

Table 7. Peripheral multiplexing description(Continued)

Destination			MSCR	Alternate functions	Source		GPIO_MAPPED	MSCR_MAPPED
Instance	Peripherals	Functions			Peripherals	Functions		
CTU_0	CTU	EXT_IN15	MSCR[767]	0000_0000	—	disable-low	0	1
				0000_0001	—	Reserved		
				0000_0010	I/O pad	PD[7]		
				0000_0011–1111_1111	—	Reserved		
SIUL	SIUL	EIRQ0	MSCR[816]	0000_0000	—	disable-low	0	1
				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	0	1
				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		
SIUL	SIUL	EIRQ2	MSCR[818]	0000_0000	—	disable-low	0	1
				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	0	1
				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_0110–1111_1111	—	Reserved		

Table 7. Peripheral multiplexing description(Continued)

Destination			MSCR	Alternate functions	Source		GPIO_MAPPED	MSCR_MAPPED
Instance	Peripherals	Functions			Peripherals	Functions		
SIUL	SIUL	EIRQ4	MSCR[820]	0000_0000	—	disable-low	0	1
				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	0	1
				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		
wkpufilter	wkpufilter	NMI	MSCR[944]	0000_0000	—	disable-high	0	1
				0000_0001	I/O pad	PC[15]		
				0000_0010	I/O pad	PA[4]		
				0000_0011	I/O pad	PB[13]		
				0000_0100–1111_1111	—	Reserved		

4.2 Package pads/pins

Table 8 shows the cross-mapping between pads and eTQFP64, eTQFP100 pinouts.

Table 8. Cross-mapping between pads and package pins

Pad	Port pin name	Pin No.	
		eTQFP64	eTQFP100
PAD_FCCU_F0	—	1	1
PAD[0]	PA[0]	2	2
PAD[1]	PA[1]	—	3
PAD[2]	PA[2]	—	4
PAD[3]	PA[3]	3	5
PAD[4]	PA[4]	4	6
PAD[5]	PA[5]	—	7
PAD[6]	PA[6]	—	8
PAD[7]	PA[7]	5	9
PAD[8]	PA[8]	6	10
PAD[9]	PA[9]	7	11
PAD[10]	PA[10]	—	12
PAD[11]	PA[11]	8	13
PAD[12]	PA[12]	9	14
PAD[13]	PA[13]	10	15
PAD[14]	PA[14]	11	16
PAD[15]	PA[15]	—	17
PAD[16]	PB[0]	—	18
PAD[17]	PB[1]	—	21
PAD[18]	PB[2]	—	22
PAD[19]	PB[3]	14	23
PAD[20]	PB[4]	15	24
PAD[21]	PB[5]	16	25
PAD[22]	PB[6]	17	26
PAD[23]	PB[7]	18	27
PAD[24]	PB[8]	—	28
PAD[25]	PB[9]	—	30
PAD[26]	PB[10]	20	31
PAD[27]	PB[11]	21	32
PAD[28]	PB[12]	—	33
PAD[29]	PB[13]	—	34

Table 8. Cross-mapping between pads and package pins(Continued)

Pad	Port pin name	Pin No.	
		eTQFP64	eTQFP100
PAD[30]	PB[14]	22	35
PAD[31]	PB[15]	23	36
PAD[32]	PC[0]	—	37
PAD[33]	PC[1]	24	38
PAD[34]	PC[2]	26	40
PAD[35]	PC[3]	27	41
PAD[36]	PC[4]	28	42
PAD[37]	PC[5]	—	43
PAD[38]	PC[6]	—	44
PAD[39]	PC[7]	29	45
PAD[40]	PC[8]	30	46
PAD[41]	PC[9]	—	47
PAD[42]	PC[10]	—	48
PAD[43]	PC[11]	31	49
PAD_FCCU_F1	—	32	50
PAD[44]	PC[12]	—	56
PAD[45]	PC[13]	—	57
PAD[46]	PC[14]	—	58
PAD_TDI	—	38	59
PAD_TMS	—	39	60
PAD_TDO	—	40	61
PAD[47]	PC[15]	41	62
PAD_TCK	—	42	63
PAD[48]	PD[0]	—	65
PAD[49]	PD[1]	—	67
PAD[50]	PD[2]	—	69
PAD[51]	PD[3]	—	70
PAD[52]	PD[4]	—	71
PAD[53]	PD[5]	—	72
PAD[54]	PD[6]	—	73
PAD[55]	PD[7]	47	74

Table 8. Cross-mapping between pads and package pins(Continued)

Pad	Port pin name	Pin No.	
		eTQFP64	eTQFP100
PAD[56]	PD[8]	48	75
PAD[57]	PD[9]	49	76
PAD[58]	PD[10]	50	77
PAD[59]	PD[11]	51	78
PAD[60]	PD[12]	—	79
PAD[61]	PD[13]	—	80
PAD[62]	PD[14]	52	81
PAD[63]	PD[15]	53	82
PAD[64]	PE[0]	—	83
PAD[65]	PE[1]	—	84
PAD[66]	PE[2]	55	86
PAD[67]	PE[3]	56	87
PAD[68]	PE[4]	—	88
PAD[69]	PE[5]	57	89
PAD[70]	PE[6]	58	90
PAD[71]	PE[7]	59	91
PAD[72]	PE[8]	60	92
PAD[73]	PE[9]	—	93
PAD[74]	PE[10]	—	94
PAD[75]	PE[11]	62	96
PAD[76]	PE[12]	—	97
PAD[77]	PE[13]	—	98
PAD[78]	PE[14]	63	99
PAD[79]	PE[15]	64	100

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 9. Memory map (XBAR slave ports)

Start address	End address	Description
0x0000_0000	0x003F_FFFF	Reserved
0x0040_0000	0x3FFF_FFFF	Flash memory
0x4000_0000	0x4000_BFFF	System RAM
0x4000_C000	0xF7FF_FFFF	Reserved
0xF800_0000	0xFBFF_FFFF	Peripherals 1 (see *Table 11*)
0xFC00_0000	0xFFFF_FFFF	Peripherals 0 (see *Table 12*)

Table 10. Flash memory map

Start address	End address	Description	RWW partition ID
0x0040_0000	0x0040_1FFF	Flash 8 KB UTest block	1
0x0040_2000	0x0040_3FFF	Reserved	
0x0040_4000	0x0040_5FFF	Flash 8 KB BAF block	1
0x0040_6000	0x0040_7FFF	Reserved	
0x0040_8000	0x0040_BFFF	Flash 16 KB Test block(1)	1
0x0040_C000	0x007F_FFFF	Reserved	
0x0080_0000	0x0080_1FFF	Flash 8 KB Data Block 0	2
0x0080_2000	0x0080_3FFF	Flash 8 KB Data Block 1	2
0x0080_4000	0x0080_5FFF	Flash 8 KB Data Block 2	2
0x0080_6000	0x0080_7FFF	Flash 8 KB Data Block 3	2
0x0080_8000	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

Table 10. Flash memory map(Continued)

Start address	End address	Description	RWW partition ID
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	0x101_FFFF	Flash 128 KB Code Block 0	1
0x0102_0000	0x0103_FFFF	Flash 128 KB Code Block 1	1
0x0104_0000	0x3FFF_FFFF	Reserved	

1. The Test Flash Block is not accessible directly through the Flash controller. It can be accessed only through a private bus by the SSCM module during the reset and by PASS module when a password match comparison is requested. The table reports the test Flash Block address, as it were visible from the Flash controller. In reality, the SSCM and PASS modules access the Test Flash block using the real Flash block address, which is 0x80_0000 till 0x80_BFFF.

Table 11. Peripheral bridge 1 memory map

Start address	End address	Peripheral	On-Platform Peripheral slot	Off-Platform Peripheral slot
0xF800_0000	0xF800_3FFF	PBRIDGE_1	0	—
0xF800_4000	0xFBC1_FFFF	Reserved	—	—
0xF800_4000	0xF807_FFFF	Reserved	—	—
0xF808_0000	0xF808_3FFF(1)	Reserved	—	—
0xF808_4000	0xFBC1_FFFF	Reserved	—	—
0xFBC2_0000	0xFBC2_3FFF	eTimer_2	—	247
0xFBC2_4000	0xFBC2_7FFF	eTimer_3	—	246
0xFBC2_8000	0xFBE6_FFFF	Reserved	—	—
0xFBE7_0000	0xFBE7_3FFF	DSPI_2	—	99
0xFBE7_4000	0xFBE8_BFFF	Reserved	—	—
0xFBE8_C000	0xFBE8_FFFF	LINFlexD_1	—	92
0xFBE9_0000	0xFBEB_FFFF	Reserved	—	—
0xFBEC_0000	0xFBEC_3FFF	FlexCAN_1	—	79
0xFBEC_4000	0xFBF5_7FFF	Reserved	—	—
0xFBF5_8000	0xFBF5_BFFF	FCCU	—	41
0xFBF5_C000	0xFBFB_01FF	Reserved	—	—
0xFBFB_0200	0xFBFB_023F	CMU_1 (System Clock)	—	19
0xFBFB_0240	0xFBFB_027F	CMU_2 (PBRIDGE 0/1)	—	19
0xFBFB_0280	0xFBFB_02BF	CMU_3 (Peripheral Clock)	—	19
0xFBFB_02C0	0xFBFF_FFFF	Reserved	—	—

1. Access to this region will not terminate in ERROR

Table 12. Peripheral bridge 0 memory map

Start address	End address	Description	On-Platform Peripheral slot	Off-Platform Peripheral slot
0xFC00_0000	0xFC00_3FFF	PBRIDGE_0	0	—
0xFC00_4000	0xFC00_7FFF	XBAR_0	1	—
0xFC00_8000	0xFC00_BFFF	EIM	2	—
0xFC00_C000	0xFC00_FFFF	Reserved	—	—
0xFC01_0000	0xFC01_3FFF	SMPU_0	4	—
0xFC01_4000	0xFC01_7FFF	Reserved	—	—
0xFC01_8000	0xFC01_BFFF	XBIC_0	6	—
0xFC01_C000	0xFC01_FFFF	Reserved	—	—
0xFC02_0000	0xFC02_3FFF	PRAM_0	8	—
0xFC02_4000	0xFC02_FFFF	Reserved	—	—
0xFC03_0000	0xFC03_3FFF	PFLASH_0	12	—
0xFC03_4000	0xFC03_FFFF	Reserved	—	—
0xFC04_0000	0xFC04_3FFF	INTC_0	16	—
0xFC04_4000	0xFC05_7FFF	Reserved	—	—
0xFC05_8000	0xFC05_BFFF	SWT_0	22	—
0xFC05_C000	0xFC06_7FFF	Reserved	—	—
0xFC06_8000	0xFC06_BFFF	STM_0	26	—
0xFC06_C000	0xFC09_FFFF	Reserved	—	—
0xFC06_0000	0xFC07_FFFF	Reserved	—	—
0xFC08_0000	0xFC08_3FFF(1)	Reserved	—	—
0xFC08_4000	0xFC09_FFFF	Reserved	—	—
0xFC0A_0000	0xFC0A_3FFF	eDMA_0	40	—
0xFC0A_4000	0xFFC1_FFFF	Reserved	—	—
0xFFC2_0000	0xFFC2_3FFF	eTimer_0	—	247
0xFFC2_4000	0xFFC2_7FFF	eTimer_1	—	246
0xFFC2_8000	0xFFC2_BFFF	Reserved	—	—
0xFFC2_C000	0xFFC2_FFFF	CTU_0	—	244
0xFFC3_0000	0xFFDF_FFFF	Reserved	—	—
0xFFE0_0000	0xFFE0_3FFF	SAR_ADC_12BIT_0	—	127
0xFFE3_C000	0xFFE3_FFFF	SAR_ADC_12BIT_B0	—	126
0xFFE0_8000	0xFFE3_7FFF	Reserved	—	—
0xFFE3_8000	0xFFE3_BFFF	Reserved	—	—
0xFFE3_C000	0xFFE3_FFFF	Reserved	—	—
0xFFE4_0000	0xFFE6_FFFF	Reserved	—	—

Table 12. Peripheral bridge 0 memory map(Continued)

Start address	End address	Description	On-Platform Peripheral slot	Off-Platform Peripheral slot
0xFFE7_0000	0xFFE7_3FFF	DSPI_0	—	99
0xFFE7_4000	0xFFE7_7FFF	DSPI_1	—	98
0xFFE7_8000	0xFFE8_BFFF	Reserved	—	—
0xFFE8_C000	0xFFE8_FFFF	LINFlexD_0	—	92
0xFFE9_0000	0xFFEB_FFFF	Reserved		
0xFFEC_0000	0xFFEC_3FFF	FlexCAN_0	—	79
0xFFEC_4000	0xFFF3_BFFF	Reserved	—	—
0xFFF3_C000	0xFFF3_FFFF	JDC	—	48
0xFFF4_0000	0xFFF4_3FFF	Reserved	—	—
0xFFF4_4000	0xFFF4_7FFF	STCU2	—	46
0xFFF4_8000	0xFFF4_BFFF	JTAGM	—	45
0xFFF4_C000	0xFFF4_FFFF	Reserved	—	—
0xFFF5_0000	0xFFF5_3FFF	MEMU	—	43
0xFFF5_4000	0xFFF6_3FFF	Reserved	—	—
0xFFF6_4000	0xFFF6_7FFF	CRC_0	—	38
0xFFF6_8000	0xFFF6_BFFF	Reserved	—	—
0xFFF6_C000	0xFFF6_FFFF	DMAMUX_0	—	36
0xFFF7_0000	0xFFF8_3FFF	Reserved	—	—
0xFFF8_4000	0xFFF8_7FFF	PIT_0	—	30
0xFFF8_8000	0xFFF9_7FFF	Reserved	—	—
0xFFF9_8000	0xFFF9_BFFF	WKPU	—	25
0xFFF9_C000	0xFFF9_FFFF	Reserved	—	—
0xFFFA_0000	0xFFFA_03FF	MC_PCU	—	23
0xFFFA_0400	0xFFFA_3FFF	PMCDIG	—	23
0xFFFA_4000	0xFFFA_7FFF	Reserved	—	—
0xFFFA_8000	0xFFFA_BFFF	MC_RGM	—	21
0xFFFA_C000	0xFFFA_FFFF	Reserved	—	—
0xFFFB_0000	0xFFFB_003F	RC16M_DIG (IRCOSC)	—	19
0xFFFB_0040	0xFFFB_007F	Reserved	—	—
0xFFFB_0080	0xFFFB_00BF	OSC16M_DIG (XOSC)	—	19
0xFFFB_00C0	0xFFFB_00FF	Reserved	—	—
0xFFFB_0100	0xFFFB_013F	PLLDIG	—	19
0xFFFB_0140	0xFFFB_01FF	Reserved	—	—
0xFFFB_0200	0xFFFB_023F	CMU_0 (PLL)	—	19

Table 12. Peripheral bridge 0 memory map(Continued)

Start address	End address	Description	On-Platform Peripheral slot	Off-Platform Peripheral slot
0xFFFB_0240	0xFFFB_06FF	Reserved	—	—
0xFFFB_0700	0xFFFB_3FFF	MC_CGM	—	19
0xFFFB_4000	0xFFFB_7FFF	Reserved	—	—
0xFFFB_8000	0xFFFB_BFFF	MC_ME	—	17
0xFFFB_C000	0xFFFB_FFFF	Reserved	—	—
0xFFFC_0000	0xFFFC_3FFF	SIUL2	—	15
0xFFFC_4000	0xFFFD_FFFF	Reserved	—	—
0xFFFE_0000	0xFFFE_3FFF	FLASH_0	—	7
0xFFFE_4000	0xFFFF_3FFF	Reserved	—	—
0xFFFE_4000	0xFFFF_7FFF	PASS	—	2
0xFFFF_8000	0xFFFF_BFFF	SSCM	—	1
0xFFFF_C000	0xFFFF_FFFF	Reserved	—	—

1. Access to this region will not terminate in ERROR.

Table 13. 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
0x0180	0x019F	PASS password - Group 2

Table 13. UTest memory map(Continued)

Start offset	End offset	Description
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 SPC570Sx microcontroller unit (MCU).

6.2 Assumptions and usage constraints

The SPC570Sx offers 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 SPC570Sx *Safety Manual* (AN4247) as constraints on the allowed usage of the MCU in order to satisfy the requested safety level.

The MCU can be 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 SPC570Sx *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 safety goal of the SPC570Sx is that does not leave its safe states for intervals greater than or equal to the fault tolerant time interval (FTTI) (10 ms) unless configured by the application software to do so.

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

- Completely unpowered
- In reset
- Operating correctly
- 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 known 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 from a system point of view, pull-ups or pull-downs need to be added at system level to safety-critical outputs depending on application requirements.

Safety analysis assumes that the system implements 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 shall 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 software.

6.3 Module categorization

The definition of the SPC570Sx 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 14*.

Table 14. 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, eDMA, DMACHMUX, flash, and the bus interconnect). Examples of ViMo are the core, 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 14. 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 SPC570Sx are shown in *Table 15*.

Table 15. Module classification

Category abbreviation	Description
ViMo	Master Safety Core, eDMA, DMACHMUX, 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 and PIT
PeMo	eTimers, STM, CTU, PIT, SAR ADC, LinFlexD, DSPI, FlexCAN, SIUL2 and PBRIDGEn
NoSaMo	Nexus, JTAG, WKPU and SPU
SuMo	RCCU, Checker Safety Core, Checker DMA, Checker DMACHMUX, Checker INTC, e2eEDC, AIC, XBIC, SMPU, SWT, SARADC_B, STCU, CRC Units, Temperature Sensors, IRCOSC, CMUs, HVDs, LVDs, Bandgap reference, LBIST controllers, REG_PROT, SWT, FCCU and MEMU

6.4 Safety architecture overview

This chapter gives an overview of the safety implementation for the SPC570Sx, 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 additional measures customized for individual modules.
- The safety of the data path from bus masters to storage and periphery is ensured by End-to-End EDC (E2E EDC) with address encoding and additional measures for individual modules. For peripherals, E2E EDC terminates after 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 non 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, nonfunctional signal monitoring, supply monitoring, physical isolation and triple voting).

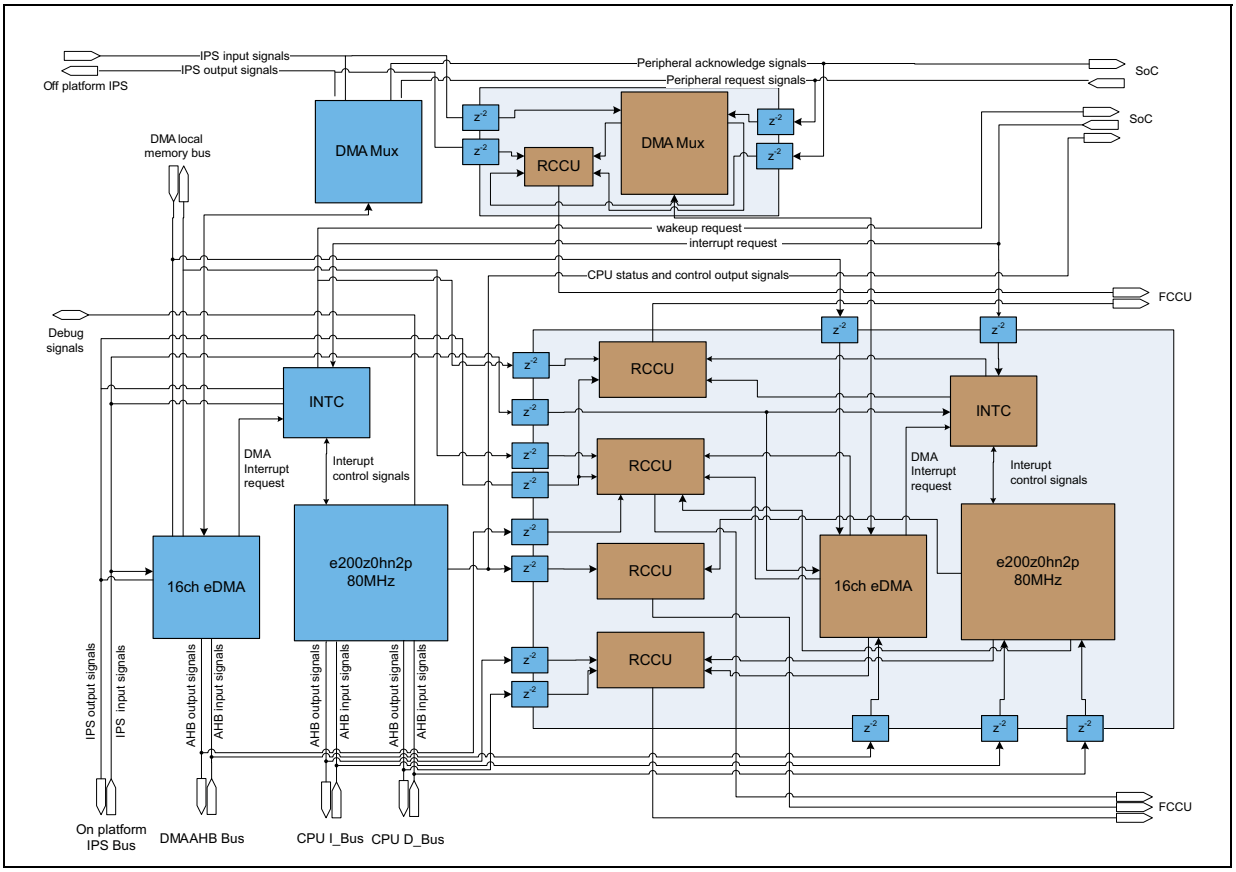
6.4.1 Lockstep

SPC570Sx 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 5. Replication architecture



Equivalent operation of both channels is supervised by checker units (RCCU) on all functional signals leaving the channels to 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 mismatch between the supervised signals.

There is a two-clock delay between the two channels to reduce the probability of common causes of failure such as clock and power glitches.

The replicated core (also known as 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. This reconfiguration takes effect 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 shall check that 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

Data from the safety masters must correctly reach all clients, and data from all clients must be transferred to safety masters reliably. We denote the connections between masters and slaves (clients) as data paths.

The data and address lines are protected on the data path with an End-to-End EDC.

EDC bits are generated on writes by masters and checked by the target slaves (namely platform RAM controller, PFLASH controller, PBRIDGE). For slaves other than memories, which cannot store checkbits, EDC bits are checked before being discarded. For flash memories EDC bits are checked and new EDC bits are calculated based only on the data before storing it into flash. For RAM memories, no check is performed for 32-bit writes and checkbits are stored directly into memory[b]. For less than 32-bit write, RAM controller performs ECC check & regeneration of checkbits.

EDC bits are provided on reads by XBAR slaves and checked by the requesting master. For XBAR slaves other than memories which cannot store checkbits, EDC bits are generated before data is provided to the master. For memories, checkbits are first used to correct possible single-bit errors (ECC increases availability). Once error correction has been performed, checkbits are not discarded, but provided to the requesting master that checks for errors introduced on the data paths (EDC only).

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. The RAM controller, of RAM with address coverage requirements, compares 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.

The characteristics of the ECC engines distributed around the system is shown in *Table 16*. Each engine can correct 1 bit and detect 2 bits.

b. Hardware verifies the coherence between data and checkbits during the next read access to the memory.

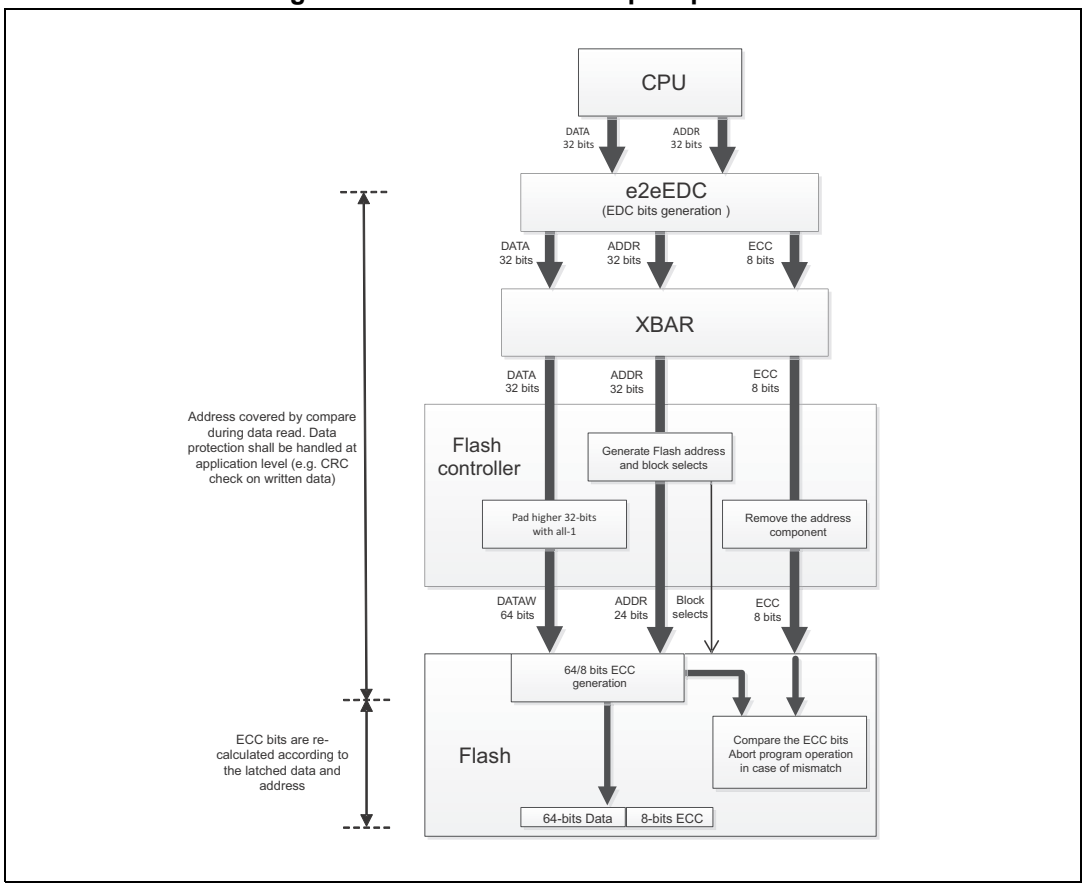
Table 16. ECC Engine Summary

ID	Interface	Address covered by ECC	Reaction upon ECC error within the address	Report to
1	CPU EDC	Yes	Reported as uncorrectable multi-bit error	FCCU
2	DMA AHB I/F	Yes	Reported as uncorrectable multi-bit error	FCCU
3	DMA local RAM I/F	Yes	Indicated by a dedicated output signal at platform top	MEMU/FCCU
4	Flash controller	Yes	Reported as uncorrectable multi-bit error	MEMU/FCCU
5	Flash	No	Encoded address checks performed	MEMU/FCCU (through Flash controller)
6	RAM controller	Yes	Reported as uncorrectable multi-bit error	MEMU/FCCU
7	PBRIDGE0/1	Yes	Reported as fault to FCCU.	FCCU

6.4.2.1 CPU to Flash write path protection

Figure 6 shows the CPU to the Flash memory write path. There is no straight EDC check on the path from the CPU to the Flash. The data integrity is verified during the reading operation. The Flash memory controller translates the address into the Flash block sector and block enable. The Flash controller removes the address component from the ECC bits and transmits them to the Flash memory. The Flash memory checks them and aborts the writing operation in case a mismatch is detected. The Flash memory packs the incoming 32-bit data and covers them with the ECC bit generated within the Flash memory itself. The XBIC block, in another protection layer, guarantees the signal integrity during the transaction across the XBAR.

Figure 6. CPU to Flash write path protection



6.4.2.2 CPU to Flash read path protection

Figure 7 shows the CPU to the Flash memory read path. The data is stored into the Flash memory together with the ECC bits needed to protect the data and address. Upon data reading, the Flash memory executes an ECC correction, followed by an EDC check. The Flash memory is able to correct 1-bit failure and to detect 2-bit failures.

The occurrence of an EDC error after ECC error is reported to the FCCU, whereas the occurrence of an ECC error is reported to the Flash memory controller, which in turn will report to the MEMU which will trigger the FCCU. ECC errors detected during access to data Flash (data for EEPROM emulation) address range are suppressed and not reported to FCCU: refer to *Section 30.5.12.1.1: e2eEDC and flash accesses* of the *Chapter 30: Flash memory controller (PFLASH Controller)*. In addition, the Flash executes a compare between the latched address and the address used internally to address the memory array. A mismatch between them is reported to 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.

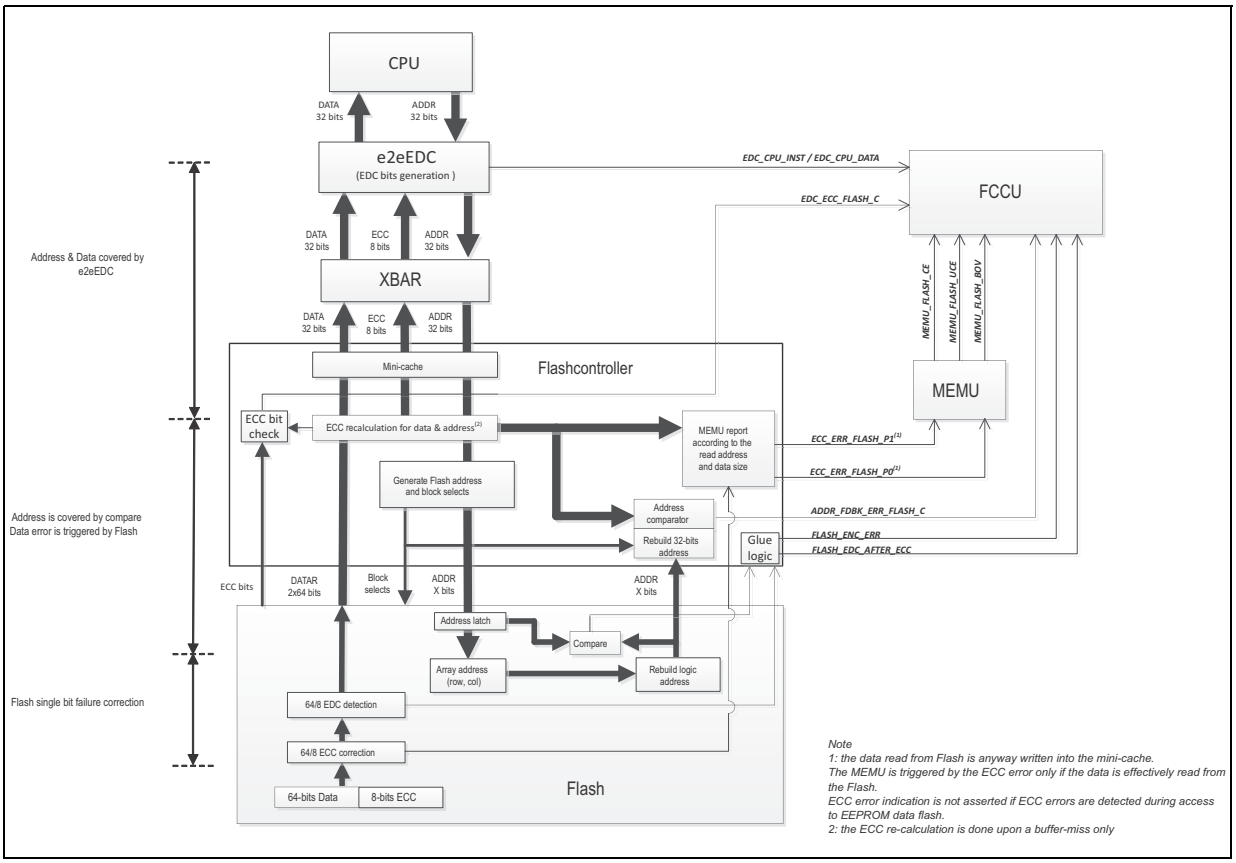
Before storing the data into the internal mini-cache, the Flash memory controller re-calculate the ECC bits in order to match the different address bus width towards the XBAR. This operation has to be done in parallel on every 32-bit portion of the data read from the Flash,

so that subsequent accesses to the mini-cache will not incur in any penalties needed to re-calculate the ECC bits.

After this ECC re-calculation, the Flash controller executes an EDC checks and reports to the FCCU in case of failure.

The data is finally transferred to the e2eEDC module, which detects any single or double bit fails and reports them to the FCCU. The XBIC block is used to guarantee the signal integrity during the transaction across the XBAR.

Figure 7. CPU to Flash read path protection

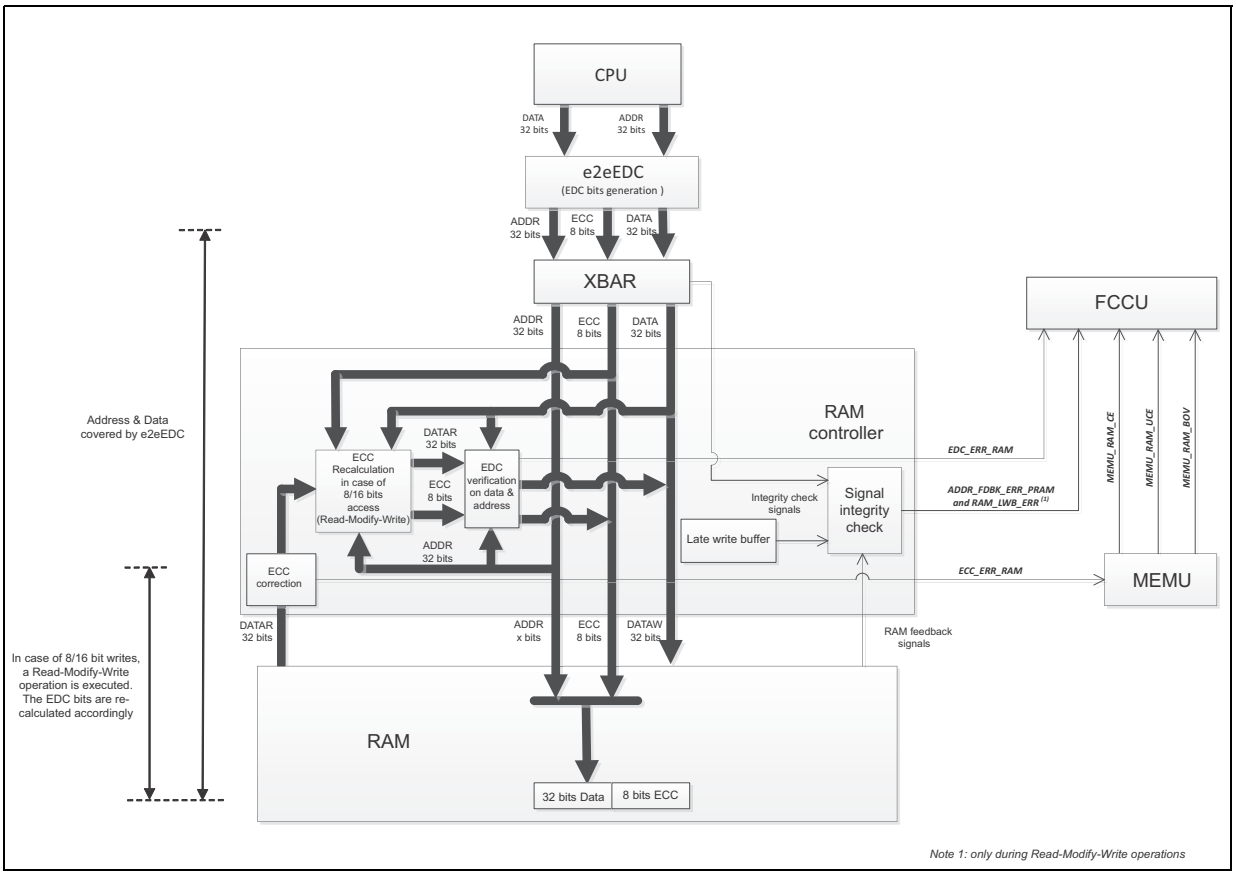


6.4.2.3 CPU to RAM write path protection

Figure 8 shows the CPU to the RAM system memory write path. The ECC bits are generated by the e2eEDC module connected to the XBAR master port and cover both the address and data (8 ECC bits are used). The RAM controller checks the size of the data being written. In case the size is 32 bit, then the data and the ECC bits are directly stored into the RAM. Vice versa, in case less than 32 bits shall be written, the RAM controller will execute a Read-Modify-Write operation. The old 32-bit data is read first and, in case of a single bit correction, the data is repaired and a signal is triggered to the MEMU, which in turn will trigger the FCCU. Afterwards, the data read from the RAM is merged with the data coming from the CPU and the ECC bits are recalculated. Then, an EDC check is done on the recalculated data bits and an error is eventually reported to the FCCU. Finally the data and the ECC bits are written into the RAM. 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 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. The controller, of RAM whose address is covered by the ECC, compares the value fed back from the latches with the values which were written to the latches when it evaluated/forwarded the results from the access to the RAM array. The comparison shall only be performed when a change of the latched values provided by the array can no longer influence the result of the RAM access. If a deviation is detected during the comparison it is signaled to the FCCU. The XBIC block, in another protection layer, guarantees the signal integrity during the transaction across the XBAR.

Figure 8. CPU to RAM write path protection

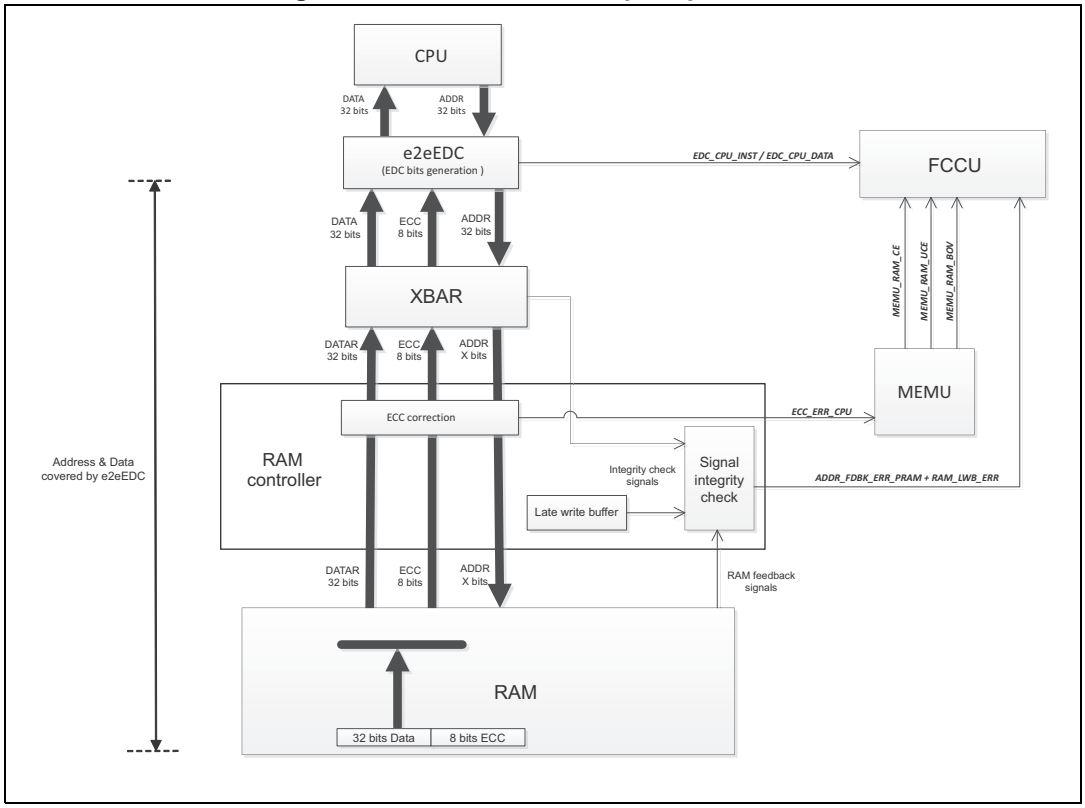


6.4.2.4 CPU to RAM read path protection

Figure 9 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 transferred to the e2eEDC module, which detects any single or double bit fails and reports them to the FCCU. The XBIC block is used to guarantee the signal integrity during the transaction across the XBAR. 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. It is assured that any flip of such a latch which occurs late enough to influence the operation conducted by the array will be fed to the controller in time to be visible together with the operation results. RAM controller of RAM compares 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. The comparison is performed when a change of the latched values provided

by the array can no longer influence the result of the RAM access. If a deviation is detected during the comparison, it is signaled to the FCCU.

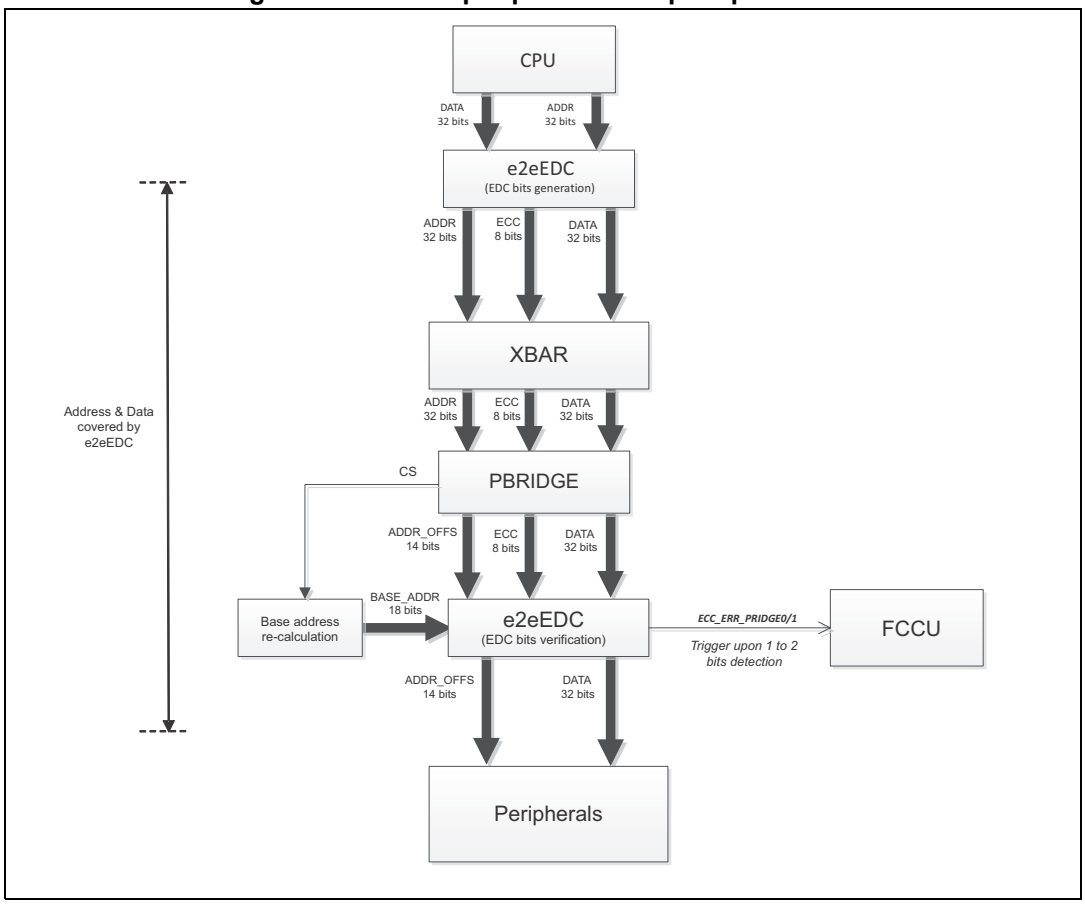
Figure 9. CPU to RAM read path protection



6.4.2.5 CPU to peripheral write path protection

Figure 10 shows the CPU to the peripheral write path. The ECC bits are generated by the e2eEDC module connected to the XBAR master port and cover both the address and data (8 ECC bits are used). The XBIC block is used to guarantee the signal integrity before and after the XBAR. The e2eEDC block after the PBRIDGE checks the received data and address for any bit failure and eventually triggers a signal to the FCCU.

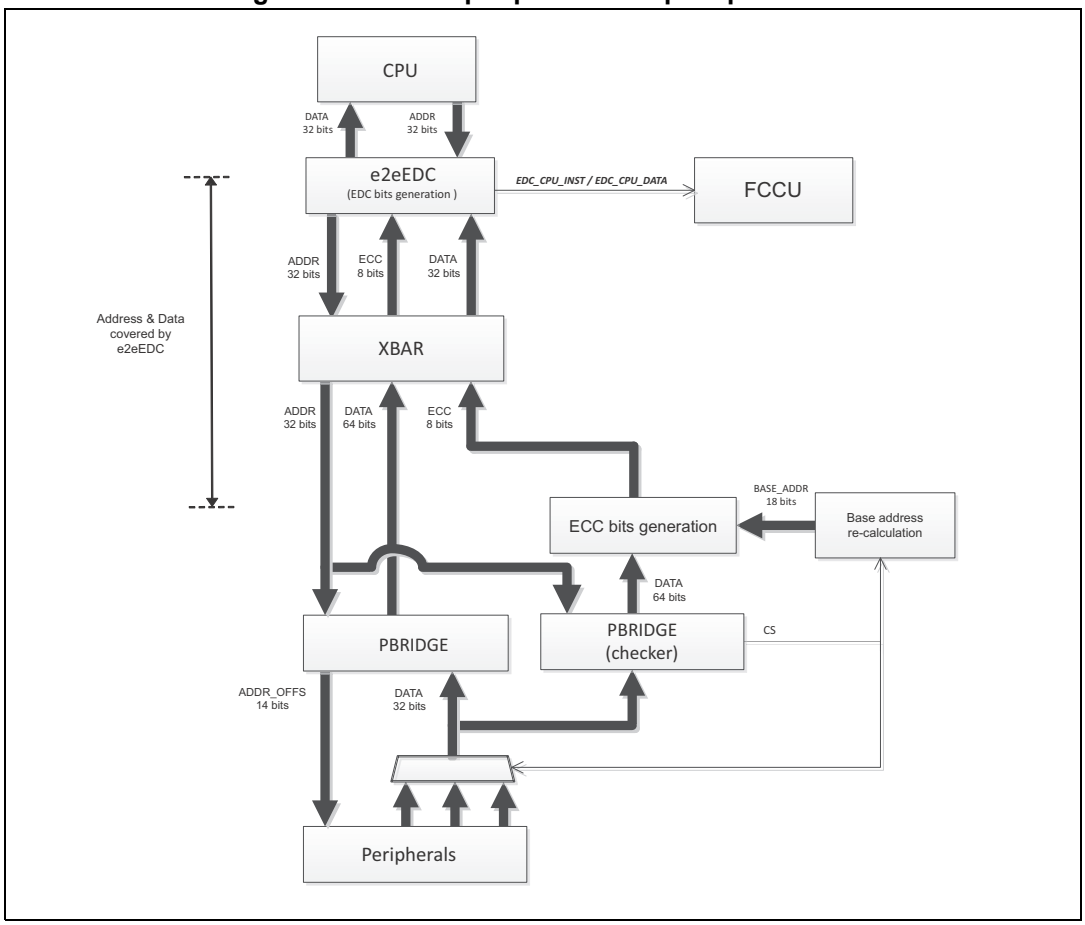
Figure 10. CPU to peripheral write path protection



6.4.2.6 CPU to peripheral read path protection

Figure 11 shows the CPU to the peripheral read path. The ECC bits are generated by the e2eEDC module connected under the PBRIDGE 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 XBIC block is used to guarantee the signal integrity before and after the XBAR. The e2eEDC block connected to the XBAR Master port checks the received data and address vs. single or double bit error and eventually triggers a signal to the FCCU.

Figure 11. CPU to peripheral read path protection



The mapping is different in read access than in write access.

6.4.3 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, in the case of DSPI, 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 they are accessible via different addresses.

6.4.4 Clock

The PLL provides error indications to the FCCU.

CMUs monitor the frequency of different clocks. If an error is detected, a error is forwarded to the FCCU.

CMU status is forwarded to the FCCU.

To decrease the probability of common cause failures, CMUs are driven by the safe clock (IRCOSC) and no elements (specifically no flip-flop) are clocked by the supervised clock on the path from a CMU to FCCU.

Modules are not directly clocked by the XOSC.

Diagnostic measures use a different clock to the function they are supervising. In particular, the FCCU and internal SWT run on the safe clock (IRCOSC) that is a different clock to the system clock.

In the case of master and replicated channels, a CMU monitors the clock of the master channel after the branch of the clock line to the master and checker channels.

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

6.4.5 Power

There are different types of voltage supervisors on the SPC570Sx: Low Voltage Detection (LVD), Minimum Voltage Detection (MVD), High Voltage Detection (HVD) and Upper Voltage Detection (UVD) monitors, that verify that safety relevant voltages are within the expected range. 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 47: Power Control Unit (MC_PCU)*.

6.4.6 Failure handling

To prevent the failure of a safety related component 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 collecting and reacting to all failure indications.

For each failure source a different reaction can be configured.

Failure sources include:

- All failure indication signals from modules within the MCU
- Control logic and signals monitored by the FCCU itself (see *Chapter 60: 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 (for example FlexCAN)

Note: Disabling the transmission capabilities 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. A functional reset does not influence the status of the FCCU registers including the storage of the failure source. This, however, is not true in the case of a destructive reset.

In order to prevent FCCU failures from affecting the overall failure reaction capability the FCCU itself is monitored by a module called FOSU (FCCU output supervisor unit).

A memory error management unit (MEMU) records the location, and additional details, about ECC errors detected in Flash and RAM.

The MEMU receives the following data from the ECC units:

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

Note: EDC logic does not report errors to the MEMU but failure indications are 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 already in the MEMU tables, 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 17. FCCU failure inputs

FCCU input	IP	Input trigger	Description	Default reaction after Reset	Error reaction path check	HW fault behavior	How to clear the fault
0	PMC	TEMP_ERROR	Temperature out of range	No reaction	—	Pulse, no SW intervention needed	Source fault is cleared itself as it is a pulse
1		LVD_ERROR	Voltage out of range from LVDs	No reaction	Selftest by tap point manipulation	Pulse, no SW intervention needed	Source fault is cleared itself as it is a pulse
2		HVD_ERROR	Voltage out of range from HVDs	No reaction	Selftest by tap point manipulation	Pulse, no SW intervention needed	Source fault is cleared itself as it is a pulse
3		DPMC_DCF_SAFETY_ERR	DCF read error (PMC trimming configuration)(1)	No reaction	LBIST	Persistent till error negation	Error is negated by clearing error status flag by SW
4	RAM	MEMORY_DCF_SAFETY_ERR	DCF read error (RAM redundancy data)(2)	No reaction	LBIST	Persistent till error negation	Error is negated by clearing error status flag by SW
5	SSCM	SSCM_XFER_ERR	SSCM transfer error (during DCF reading used to configure the SSCM and the STCU).	No reaction	Not testable (but wrong configuration can trigger other detection)	SSCM gets cleared by SW. Fatal error pulse no SW needed.	SSCM_XFER_ERR is cleared Fatal error is a pulse that clears itself

Table 17. FCCU failure inputs(Continued)

FCCU input	IP	Input trigger	Description	Default reaction after Reset	Error reaction path check	HW fault behavior	How to clear the fault
6	STCU2	STCU_UF[3]	STCU2 Unrecoverable Fault during the BIST execution, corresponding to the STCU_ERR_STAT.UFSF error bit	Functional Reset[4]	SW check of BIST MISR	Clear status through FCCU	FCCU channel status to be cleared, this clears the STCU status bit as well as the fault
7		STCU_RF[3]	STCU2 Recoverable Fault during the BIST execution, corresponding to the STCU_ERR_STAT.RFSF error bit	No reaction	SW check of BIST MISR	Clear status through FCCU	FCCU channel status to be cleared, this clears the STCU status bit as well as the fault
8		STCU_LMBIST_USR_ERR	STCU2 fault in case LBIST or MBIST[5] control signals go to wrong condition during user application	No reaction	STCU activation during User application running	Clear status through FCCU	FCCU channel status to be cleared, this clears the STCU status bit as well as the fault
9	Debug logic	JTAG_NPC_MON	JTAG or NPC not in reset or activation of dangerous debug functionality	No reaction	Feedback loop via FCCU Fake bit	Clear status through FCCU	FCCU channel status to be cleared, this clears the fault
10	RCCU	RCCU_CPU_NON_AHB	CPU non AHB RCCU error trigger	No reaction	LBIST + Feedback loop via FCCU Fake bit and additional RCCU input	Clear status through FCCU	FCCU channel status to be cleared, this clears the fault
11		RCCU_CPU_AHB	CPU AHB RCCU error trigger	No reaction	LBIST + Feedback loop via FCCU Fake bit and additional RCCU input	Clear status through FCCU	FCCU channel status to be cleared, this clears the fault

Table 17. FCCU failure inputs(Continued)

FCCU input	IP	Input trigger	Description	Default reaction after Reset	Error reaction path check	HW fault behavior	How to clear the fault
12	SWT	SWT0_ResetReq	Software watchdog timer, 2nd timeout	Functional Reset	LBIST + SW test by not triggering SWT during startup	Cleared by reset only	Cleared by reset
13		SWT1_InterruptReq	Software watchdog timer, 1st timeout	No reaction	LBIST + SW test by not triggering SWT during startup	Needs to be cleared by SW, wc1 in SWT register	Needs to be cleared by SW, wc1 in SWT register
14	RCCU	RCCU_PBRIDGE_0	PBRIDGE 0 protocol checker	No reaction	LBIST + Feedback loop via FCCU Fake bit and additional RCCU input	Clear status through FCCU	FCCU channel status to be cleared, this clears the fault
15		RCCU_PBRIDGE_1	PBRIDGE 1 protocol checker	No reaction	LBIST + Feedback loop via FCCU Fake bit and additional RCCU input	Clear status through FCCU	FCCU channel status to be cleared, this clears the fault

Table 17. FCCU failure inputs(Continued)

FCCU input	IP	Input trigger	Description	Default reaction after Reset	Error reaction path check	HW fault behavior	How to clear the fault
16	MEMU	MEMU_RAM_CE	MEMU System RAM correctable error	No reaction	LBIST + Injection of RAM ECC error via ECC bit access and later reading	Clear status in MEMU	The fault is cleared by clearing the MEMU status flag, MEMU_ERR_FLAG [SR_CE]
17		MEMU_RAM_UCE	MEMU System RAM uncorrectable error	No reaction	LBIST + Injection of RAM ECC error via ECC bit access and later reading	Clear status in MEMU	The fault is cleared by clearing the MEMU status flag, MEMU_ERR_FLAG [SR_CE]
18		MEMU_RAM_BOV	MEMU System RAM Buffer Overflow	No reaction	LBIST + Injection of Overflow error indication by MEMU debug register	Clear status in MEMU	The fault is cleared by clearing the MEMU status flag, MEMU_ERR_FLAG [SR_CE]
19		MEMU_PER_CE	MEMU DMA RAM correctable error	No reaction	LBIST + Injection of correctable error entry by writing one into the MEMU	Clear status in MEMU	The fault is cleared by clearing the MEMU status flag, MEMU_ERR_FLAG [SR_CE]
20		MEMU_PER_UCE	MEMU DMA RAM uncorrectable error	No reaction	LBIST + Injection of uncorrectable error entry by writing one into the MEMU	Clear status in MEMU	The fault is cleared by clearing the MEMU status flag, MEMU_ERR_FLAG [SR_CE]
21		MEMU_PER_BOV	MEMU DMA RAM Buffer Overflow	No reaction	LBIST + Injection of Overflow error indication by MEMU debug register	Clear status in MEMU	The fault is cleared by clearing the MEMU status flag, MEMU_ERR_FLAG [SR_CE]

Table 17. FCCU failure inputs(Continued)

FCCU input	IP	Input trigger	Description	Default reaction after Reset	Error reaction path check	HW fault behavior	How to clear the fault
22	MEMU	MEMU_FLASH_CE	MEMU Flash Correctable Error	No reaction	LBIST + Reading preprogrammed Correctable ECC error from UTestFlash	Clear status in MEMU	The fault is cleared by clearing the MEMU status flag, MEMU_ERR_FLAG [SR_CE]
23		MEMU_FLASH_UCE	MEMU Flash Uncorrectable Error	No reaction	LBIST + Reading preprogrammed uncorrectable ECC error from UTestFlash	Clear status in MEMU	The fault is cleared by clearing the MEMU status flag, MEMU_ERR_FLAG [SR_CE]
24		MEMU_FLASH_BOV	MEMU Flash Buffer Overflow	No reaction	LBIST + Injection of Overflow error indication by MEMU debug register	Clear status in MEMU	The fault is cleared by clearing the MEMU status flag, MEMU_ERR_FLAG [SR_CE]
25	RCCU	RCCU_INTC	INTC RCCU error trigger	No reaction	LBIST + Feedback loop via FCCU Fake bit and additional RCCU input	Clear status through FCCU	FCCU channel status to be cleared, this clears the fault
26		RCCU_DMA	DMA RCCU error trigger	No reaction	LBIST + Feedback loop via FCCU Fake bit and additional RCCU input	Clear status through FCCU	FCCU channel status to be cleared, this clears the fault
27	ADC	SAR_WDG	SAR0…SAR7 watchdog (ORed together)	No reaction	LBIST + ADC conversion outside the watchdog range	SW clear required in ADCDIG	SW clear required in ADCDIG
28		SARB_WDG	Reference SAR watchdog	No reaction	LBIST + ADC conversion outside the watchdog range	SW clear required in ADCDIG	SW clear required in ADCDIG

Table 17. FCCU failure inputs(Continued)

FCCU input	IP	Input trigger	Description	Default reaction after Reset	Error reaction path check	HW fault behavior	How to clear the fault
29	PLL	FM_PLL0	Loss of lock	No reaction	Forced loss by PLL reconfiguration	SW clear required in PLLDIG	SW clear required in PLLDIG
30		FM_PLL1	Loss of lock	No reaction	Forced loss by PLL reconfiguration	SW clear required in PLLDIG	SW clear required in PLLDIG
31	CGM	CMU_0_OSC	Loss of XOSC clock	No reaction	—	Clear status in CMU	Clear status bit in CMU by SW
32		CMU_0_PLL0	PLL0 frequency out of range	No reaction	Test by misconfiguration	Clear status in CMU	Clear status bit in CMU by SW
33		CMU_1	AHB system clock out of range	No reaction	Test by misconfiguration	Clear status in CMU	Clear status bit in CMU by SW
34		CMU_2	IPS clock out of range	No reaction	Test by misconfiguration	Clear status in CMU	Clear status bit in CMU by SW
35	Flash	REF_ERR_FLASH	Flash Read Reference Error	No reaction	LBIST	Clear status through FCCU	FCCU channel status to be cleared, this clears the fault
36	Flash controller	EDC_ECC_FLASH	EDC after ECC for Flash array	No reaction	LBIST	Clear status through FCCU	FCCU channel status to be cleared, this clears the fault
37		EDC_ECC_FLASH_C	EDC after ECC for FLASH Controller	No reaction	LBIST	Clear status through FCCU	FCCU channel status to be cleared, this clears the fault

Table 17. FCCU failure inputs(Continued)

FCCU input	IP	Input trigger	Description	Default reaction after Reset	Error reaction path check	HW fault behavior	How to clear the fault
38	Flash	ENC_ERR_FLASH	Flash encoding error(6)	No reaction	LBIST	Clear status through FCCU	FCCU channel status to be cleared, this clears the fault
39	Flash controller	ADDR_FDBK_ERR_FLASH_C	Flash Controller Address Feedback Error	No reaction	LBIST	Clear status through FCCU	FCCU channel status to be cleared, this clears the fault
40	XBIC	XBAR	XBAR transaction monitor mismatch	No reaction	LBIST	Clear status through FCCU	FCCU channel status to be cleared, this clears the fault
41	PMC	PMU5 VD BIST	PMU voltage detectors BIST (enabled via SW)	No reaction	LBIST	Clear status through FCCU	FCCU channel status to be cleared, this clears the fault
42	SSCM	SSCM_ACTIVE	Indication of Spurious activation of SSCM (during application runtime)	No reaction	—	Pulse	Self clearing pulse
43	RAM controller	ADDR_FDBK_ERR_PRAM + RAM_LWB_ERR	System RAM Address Feedback + RAM Late-Write Buffer mismatch	No reaction	LBIST	Persistent until SW clearing from FCCU	FCCU channel status to be cleared, this clears the fault
44	SMPU	SMPU Checker Error	SMPU monitor error alarm	No reaction	See Note (7)	Pulse, no SW intervention needed	Self clearing pulse

Table 17. FCCU failure inputs(Continued)

FCCU input	IP	Input trigger	Description	Default reaction after Reset	Error reaction path check	HW fault behavior	How to clear the fault
45	RAM controller	EDC_ERR_PRAM	EDC after ECC for RAM (read-modify-write ECC)	No reaction	—	Persistent until SW clearing from FCCU	FCCU channel status to be cleared, this clears the fault
46	TCU	DFTx	Test circuitry unexpected activation	No reaction	—	Persistent until reset	Fault cleared after reset
47		DFT_1	Test circuitry unexpected activation	No reaction	—	Persistent until reset	Fault cleared after reset
48		DFT_2	Test circuitry unexpected activation	No reaction	—	Persistent until reset	Fault cleared after reset
49		DFT_3	Test circuitry unexpected activation	No reaction	—	Persistent until reset	Fault cleared after reset
50	CPU	Main Core	Safety Core exception indication	No reaction	LBIST + Willfully cause an appropriate exception (for example, address a non-existing address)	Pulse, no SW intervention needed	Self clearing pulse
51	SOC	Lockstep Mode	Indication of disablement of Checker Core or RCCUs	No reaction	Feedback loop via FCCU Fake bit	Persistent until cleared by HW (coming out of debug mode)	Fault cleared by HW as soon as the device comes out of debug mode
52	MC_ME	Safe Mode	Safe Mode entry indication	No reaction	LBIST + (in principle requesting an RGM "safe mode" but that is not typically something want to do)	Persistent as long as the system remains in safe mode	Fault cleared by HW as soon as the device comes out of safe mode
53	IO	Compensation Disable	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 until cleared by HW	Fault is cleared by HW as soon as pad compensation is enabled
54	CGM	CMU_3	Peripheral clock out of range	No reaction	Test by misconfiguration	Clear status in CMU	Clear status bit in CMU by SW

Table 17. FCCU failure inputs(Continued)

FCCU input	IP	Input trigger	Description	Default reaction after Reset	Error reaction path check	HW fault behavior	How to clear the fault
55	SMPU	SMPU FCCU alarm	SMPU protection error	No reaction	Test by intentionally performing an access, violating the programmed configuration. See Note (8)	Pulse, no SW intervention needed	Self clearing pulse
56	e2eEDC wrappers	EDC_ERR_PBRIDGE0	PBRIDGE 0 EDC error	No reaction	LBIST	Pulse, no SW intervention needed	Self clearing pulse
57		EDC_ERR_PBRIDGE1	PBRIDGE 1 EDC error	No reaction	LBIST	Pulse, no SW intervention needed	Self clearing pulse
58		EDC_ERR_DMA	DMA EDC error	No reaction	LBIST	Pulse, no SW intervention needed	Self clearing pulse
59		EDC_ERR_CPU_INST	CPU instruction AHB EDC error	No reaction	LBIST	Pulse, no SW intervention needed	Self clearing pulse
60		EDC_ERR_CPU_DATA	CPU data AHB EDC error	No reaction	LBIST	Pulse, no SW intervention needed	Self clearing pulse

Table 17. FCCU failure inputs(Continued)

FCCU input	IP	Input trigger	Description	Default reaction after Reset	Error reaction path check	HW fault behavior	How to clear the fault
61	RCCU	RCCU_DMA_MUX	DMA Mux error trigger (It will become active, when there would be found difference in DMA MUX lock step)	No reaction	LBIST + Feedback loop via FCCU Fake bit and additional RCCU input	Persistent until SW clearing from FCCU	FCCU channel status to be cleared, this clears the fault
62		RCCU_EIM	RCCU EIM ECC trigger	No reaction	LBIST + Feedback loop via FCCU Fake bit and additional RCCU input	Persistent until SW clearing from FCCU	FCCU channel status to be cleared, this clears the fault

1. DCF clients are triple voted, if any bit flips then it will generate a safety error and some PMC DCF clients are write once if we try to access it multiple times then also it will generate a safety error.
2. These faults are generated when reading DCF during boot. In case of fault, the flag will already be set when application will start.
3. The STCU2 error conditions can be configured as unrecoverable or recoverable. The effect is to trigger two different output triggers, which are mapped to separate fault inputs of the FCCU.
4. Destructive reset in case of offline LBIST from RGM.
5. For the SPC570Sx device, the STCU2 supports off-line MBIST. On-line/off-line LBIST and on-line MBIST are not supported. Please see *Chapter 7: Device configuration* - *Chapter 61: Self-Test Control Unit (STCU2)* for additional information.
6. The Flash recalculates the ECC bits and compares 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.
7. SMPU checker monitors the protocol violation due to faults in the SMPU. Assertion of this fault indicates that SMPU failed to provide error response to the bus master in the event of access violation.
8. SMPU flags a fault to the FCCU if it detects a bus access violating the permissions as configured in SMPU.

6.4.7 Failure indication

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

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

Independent of the chosen protocol, at least one of the failure indication pins is high to indicate “operational”.

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

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

After power-on, the error out pins go to “operational” state 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.

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

Software can activate error pins by injecting fake faults to verify the integrity of the reaction path.

6.4.8 Latent fault detection

SPC570Sx 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 incorrect function.

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 X single-bit
- Any error in any other selftest module including the selftest controller itself

Self tests can be executed during startup or during operation.

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)

6.4.9 Common Cause Failures (CCFs)

Various prevention and/or detection measures have been included to cover CCFs. These measures include the supervision of configuration registers and the 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).

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.

6.4.10 Resets

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

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

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

6.4.11 LBIST/MBIST architecture partitioning

The Logic BIST partitioning is provided in *Figure 12*.

Figure 12. LBIST partitioning

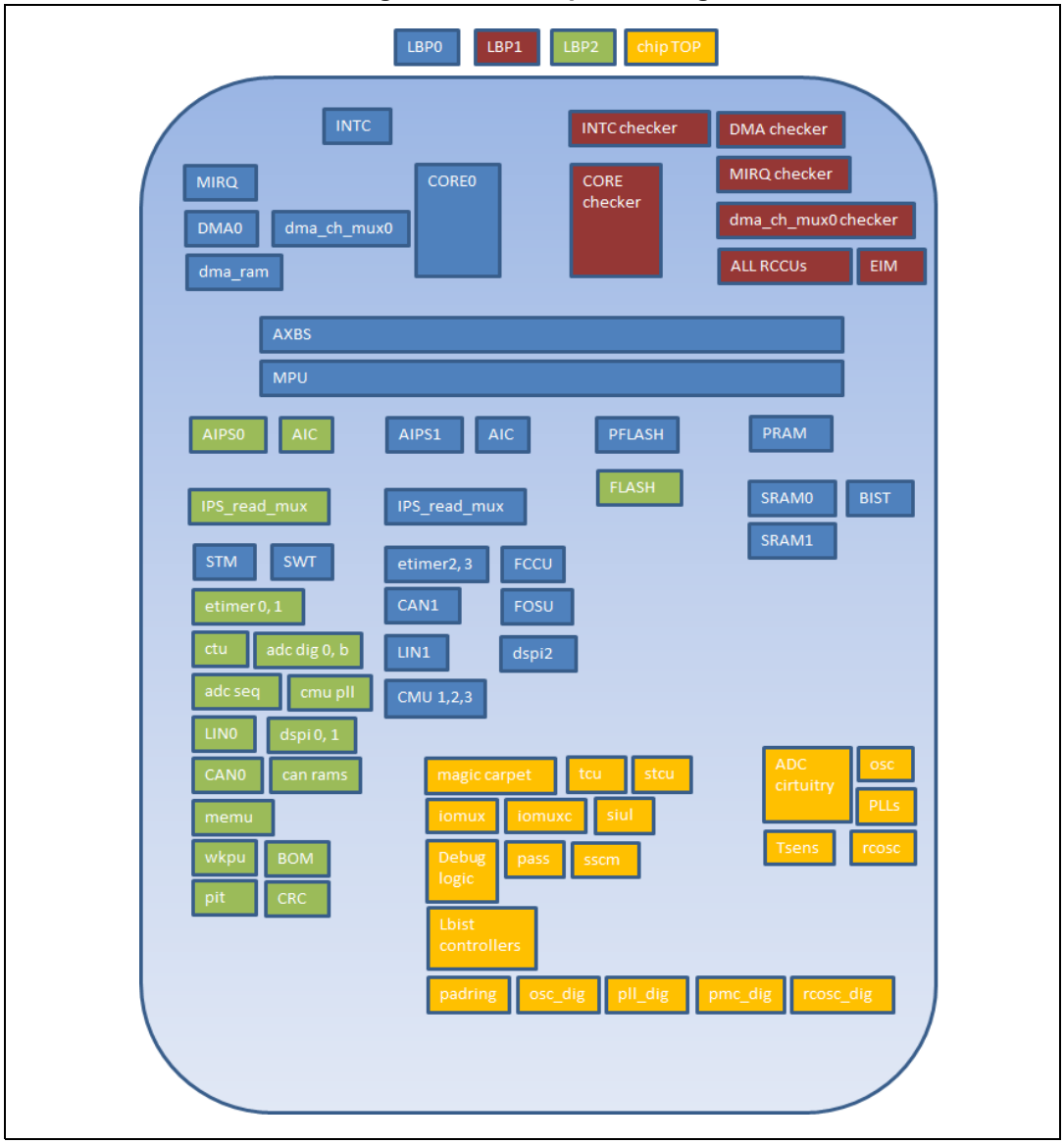


Table 18. MBIST partitioning

M_BIST	Partition
0	ram_sram_1
1	ram_sram_10
2	ram_dma
3	ram_can_rxim_1
4	ram_can_rxim_0

Table 18. MBIST partitioning(Continued)

M_BIST	Partition
5	ram_can_mb_1
6	ram_can_mb_0
7	ram_mpflash
8	bam_rom

The settings of LBIST/MBIST configurations described as preliminary data in *Table 18* are provided in *Section 7.9.3: Self-Test Control Unit (STCU2) configuration*.

Process time considers the IRC as the frequency source. The power consumption data can be found in the latest version of the datasheet.

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.

Table 19. Self test configurations

Config. Number	Comment	BIST	Mode	Freq. setting	number of patterns	Process time (ms)
1	*MBIST configuration*	MBIST	Parallel	73.3 MHz (PLL)	—	10.4
2	*LBIST configuration*	LBIST	Parallel	18.3 MHz (PLL)	2K (0x7D0)	23.7
3	*Full self-test configuration*	LBIST	Sequential	16 MHz (RC)	2K (0x7D0)	38
		MBIST	Parallel	18.3 MHz (PLL)	—	
4	*Bypass configuration*	No BIST	—	—	—	0.5

Table 20. MBIST configuration

Register name	Register value	Comment	Reference
STCU_SKC	0xD3FEA98B	Key 1	*Section 61.5.2*
STCU_SKC	0x2C015674	Key 2	*Section 61.5.2*
STCU_CFG	0x10000001	Pointer set for MBIST0, MBIST reduced algorithm (PMOSEN bit set to '0'), PMOSEN and MBU can be played with to chose different algorithms and MBIST coverage, WRP bit set to '0' to allow IPS access.	*Section 61.5.3*
STCU_PLL_CFG	0x06020038	PLL programmed for 73.3 MHz with input frequency as RC (16 MHz ± 8%), PLL output freq = (Input_freq * LDF) / (ODF * IDF), = (Input_freq * 'h38) / (06 * 02), Play with ODF factor(06) to reduce frequency	*Section 61.5.4*
STCU_WDG	0x0003D090		*Section 61.5.5*

Table 20. MBIST configuration(Continued)

Register name	Register value	Comment	Reference
STCU_MBUFML	0x00000000	Programming '1' to the respective MBIST in this register generates an unrecoverable fault in the case of MBIST failure. If the fault is permanent, then the device will never recover from this programming. So it is safe to program '0' in this register and later STCU status can be read by software.	*Section 61.5.12*
STCU_ERR_FM	0x00000000	Programming '1' to any corresponding bit of this register can make the device unrecoverable if the fault is permanent.	*Section 61.5.9*
STCU_MB_CTRL0	0x91000000	Next pointer set to MBIST1, CSM BIST set to '1' to indicate parallel running of MBIST0 and MBIST1	*Section 61.5.30*
STCU_MB_CTRL1	0x92000000	Next pointer set to MBIST2, CSM BIST set to '1' to indicate parallel running of MBIST1 and MBIST2	*Section 61.5.30*
STCU_MB_CTRL2	0x93000000	Next pointer set to MBIST3, CSM BIST set to '1' to indicate parallel running of MBIST2 and MBIST3	*Section 61.5.30*
STCU_MB_CTRL3	0x94000000	Next pointer set to MBIST4, CSM BIST set to '1' to indicate parallel running of MBIST3 and MBIST4	*Section 61.5.30*
STCU_MB_CTRL4	0x95000000	Next pointer set to MBIST5, CSM BIST set to '1' to indicate parallel running of MBIST4 and MBIST5	*Section 61.5.30*
STCU_MB_CTRL5	0x96000000	Next pointer set to MBIST6, CSM BIST set to '1' to indicate parallel running of MBIST5 and MBIST6	*Section 61.5.30*
STCU_MB_CTRL6	0x97000000	Next pointer set to MBIST7, CSM BIST set to '1' to indicate parallel running of MBIST6 and MBIST7	*Section 61.5.30*
STCU_MB_CTRL7	0x98000000	Next pointer set to MBIST8, CSM BIST set to '1' to indicate parallel running of MBIST7 and MBIST8	*Section 61.5.30*
STCU_MB_CTRL8	0x7F000000	Next pointer set to MBIST9, CSM BIST set to '1' to indicate parallel running of MBIST8 and MBIST9	*Section 61.5.30*
STCU_RUN	0x00000201	Selftest triggered with MBIST run on PLL frequency	*Section 61.5.1*

Table 21. LBIST configuration

Register name	Register value	Comment	Reference
STCU_SKC	0xD3FEA98B	Key 1	*Section 61.5.2*
STCU_SKC	0x2C015674	Key 2	*Section 61.5.2*
STCU_CFG	0x005A0001		*Section 61.5.3*
STCU_PLL_CFG	0x18020038	PLL programmed for 35 MHz with input frequency as RC (16 MHz ± 8%), PLL output freq = (Input_freq * LDF) / (ODF * IDF), = (Input_freq * 'h46) / (10 * 02), Play with ODF factor(10) to reduce frequency	*Section 61.5.4*
STCU_WDG	0x0001E848		*Section 61.5.5*

Table 21. LBIST configuration(Continued)

Register name	Register value	Comment	Reference
STCU_LBUFM	0x00000000	Programming '1' to the respective LBIST in this register generates an unrecoverable fault in the case of LBIST failure. If the fault is permanent, then the device will never recover from this programming. So it is safe to program '0' in this register and later STCU status can be read by software.	*Section 61.5.12*
STCU_LB_CTRL0	0x81001100	Next pointer to LBIST partition1, parallel mode, CSM bit (31st) can be played with to run	*Section 61.5.22*
STCU_LB_PCS0	0x000007D0	Number of patterns -2000 for partition0	*Section 61.5.23*
STCU_LB_MISREL0	0x9CCC9484	MISR lower 32 bits for 2000th pattern for partition0. Will change according to the number of patterns to be executed.	*Section 61.5.26*
STCU_LB_MISREH0	0x36565224	MISR upper 32 bits for 2000th pattern for partition0. Will change according to the number of patterns to be executed.	*Section 61.5.27*
STCU_LB_CTRL1	0x82001100	Next pointer to LBIST partition2, parallel mode, CSM bit (31st) can be played with to run	*Section 61.5.22*
STCU_LB_PCS1	0x000007D0	Number of patterns -2000 for partition1	*Section 61.5.23*
STCU_LB_MISREL1	0xCFC6629E	MISR lower 32 bits for 2000th pattern for partition1. Will change according to the number of patterns to be executed.	*Section 61.5.26*
STCU_LB_MISREH1	0x10E9A429	MISR upper 32 bits for 2000th pattern for partition1. Will change according to the number of patterns to be executed.	*Section 61.5.27*
STCU_LB_CTRL2	0x7F001100	Next pointer to null to indicate end of selftest	*Section 61.5.22*
STCU_LB_PCS2	0x000007D0	Number of patterns -2000 for partition2	*Section 61.5.23*
STCU_LB_MISREL2	0xA582B986	MISR lower 32 bits for 2000th pattern for partition2. Will change according to the number of patterns to be executed.	*Section 61.5.26*
STCU_LB_MISREH2	0xD654207A	MISR upper 32 bits for 2000th pattern for partition2. Will change according to the number of patterns to be executed.	*Section 61.5.27*
STCU_RUN	0x00001010	STCU triggered with LBIST programmed to run on PLL frequency	*Section 61.5.1*

Table 22. Full self-test configuration

Register name	Register value	Comment	Reference
STCU_SKC	0xD3FEA98B	Key 1	*Section 61.5.2*
STCU_SKC	0x2C015674	Key 2	*Section 61.5.2*

Table 22. Full self-test configuration(Continued)

Register name	Register value	Comment	Reference
STCU_CFG	0x005A0001	Pointer set for MBIST0, MBIST reduced algorithm (PMOSEN bit set to '0'), PMOSEN and MBU can be played with to chose different algorithms and MBIST coverage, WRP bit set to '0' to allow IPS access.	*Section 61.5.3*
STCU_PLL_CFG	0x06020038	PLL programmed for 73.3 MHz with input frequency as RC (16 MHz ± 8%), PLL output freq = (Input_freq * LDF) / (ODF * IDF), = (Input_freq * 'h38) / (06 * 02), Play with ODF factor(06) to reduce frequency	*Section 61.5.4*
STCU_WDG	0x0003D090		*Section 61.5.5*
STCU_LB_CTRL0	0x01001100	Next pointer to LBIST partition1, sequential mode, CSM bit (31st) can be played with to run in serial mode	*Section 61.5.22*
STCU_LB_PCS0	0x000007D0	Number of patterns -2000 for partition0	*Section 61.5.23*
STCU_LB_MISREL0	0x9CCC9484	MISR lower 32 bits for 2000th pattern for partition0. Will change according to the number of patterns to be executed.	*Section 61.5.26*
STCU_LB_MISREH0	0x36565224	MISR upper 32 bits for 2000th pattern for partition0. Will change according to the number of patterns to be executed.	*Section 61.5.27*
STCU_LB_CTRL1	0x02001100	Next pointer to LBIST partition2, sequential mode, CSM bit (31st) can be played with to run in serial mode	*Section 61.5.22*
STCU_LB_PCS1	0x000007D0	Number of patterns -2000 for partition1	*Section 61.5.23*
STCU_LB_MISREL1	0xCFC6629E	MISR lower 32 bits for 2000th pattern for partition1. Will change according to the number of patterns to be executed.	*Section 61.5.26*
STCU_LB_MISREH1	0x10E9A429	MISR upper 32 bits for 2000th pattern for partition1. Will change according to the number of patterns to be executed.	*Section 61.5.27*
STCU_LB_CTRL2	0x10001100	Next pointer to MBIST0 to indicate MBIST run after LBIST	*Section 61.5.22*
STCU_LB_PCS2	0x000007D0	Number of patterns -2000 for partition2	*Section 61.5.23*
STCU_LB_MISREL2	0xA582B986	MISR lower 32 bits for 2000th pattern for partition2. Will change according to the number of patterns to be executed.	*Section 61.5.26*
STCU_LB_MISREH2	0xD654207A	MISR upper 32 bits for 2000th pattern for partition2. Will change according to the number of patterns to be executed.	*Section 61.5.27*
STCU_MB_CTRL0	0x91000000	Next pointer set to MBIST1, CSM BIST set to '1' to indicate parallel running of MBIST0 and MBIST1	*Section 61.5.30*
STCU_MB_CTRL1	0x92000000	Next pointer set to MBIST2, CSM BIST set to '1' to indicate parallel running of MBIST1 and MBIST2	*Section 61.5.30*
STCU_MB_CTRL2	0x93000000	Next pointer set to MBIST3, CSM BIST set to '1' to indicate parallel running of MBIST2 and MBIST3	*Section 61.5.30*

Table 22. Full self-test configuration(Continued)

Register name	Register value	Comment	Reference
STCU_MB_CTRL3	0x94000000	Next pointer set to MBIST4, CSM BIST set to '1' to indicate parallel running of MBIST3 and MBIST4	*Section 61.5.30*
STCU_MB_CTRL4	0x95000000	Next pointer set to MBIST5, CSM BIST set to '1' to indicate parallel running of MBIST4 and MBIST5	*Section 61.5.30*
STCU_MB_CTRL5	0x96000000	Next pointer set to MBIST6, CSM BIST set to '1' to indicate parallel running of MBIST5 and MBIST6	*Section 61.5.30*
STCU_MB_CTRL6	0x97000000	Next pointer set to MBIST7, CSM BIST set to '1' to indicate parallel running of MBIST6 and MBIST7	*Section 61.5.30*
STCU_MB_CTRL7	0x98000000	Next pointer set to MBIST8, CSM BIST set to '1' to indicate parallel running of MBIST7 and MBIST8	*Section 61.5.30*
STCU_MB_CTRL8	0x99000000	Next pointer set to MBIST9, CSM BIST set to '1' to indicate parallel running of MBIST8 and MBIST9	*Section 61.5.30*
STCU_MB_CTRL9	0x7F000000	Next pointer set to null pointer to indicate the end of the selftest	*Section 61.5.30*
STCU_RUN	0x00000201	Selftest triggered with MBIST run PLL, LBIST with RC	*Section 61.5.1*

Table 23. Bypass configuration

Register name	Register value	Comment	Reference
STCU_SKC	0xD3FEA98B	Key 1	*Section 61.5.2*
STCU_SKC	0x2C015674	Key 2	*Section 61.5.2*
STCU_RUN	0x00000400		*Section 61.5.1*

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

SPC570Sx has two separate cores that perform various computational and control functions.

Table 24. SPC570Sx core modules

Instance	Description
Main Core_0	The main computational shell includes the Main Core_0, which uses an e200z0h core. This core is used for general computational functions.
Checker Core_0s	Checker Core_0s uses an e200z0h core. When enabled, Checker Core_0s operates in lock step mode with Main Core_0 executing exactly the same instructions as Main Core_0. Thus, Checker Core_0s checks to ensure that Main Core_0 is executing correctly.

For more information, please see *Chapter 12: e200z0h Core description*.

7.2.1 Core reset settings

Table 25 shows the state of the *PowerISA 2.06* architected registers and other optional resources immediately following a system reset.

Table 25. Reset settings for e200z0h 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)

Table 25. Reset settings for e200z0h resources(Continued)

Resource	System reset setting
DBCR1	0x0000_0000(3)
DBCR2	0x0000_0000(3)
DBSR	0x1000_0000(3)
DEAR	Unaffected(2)
DSRR0	Unaffected(2)
DSRR1	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)
IVPR	Unaffected(2)
LR	Unaffected(2)
L1CFG0(4)	0x0000_0000
MCSR	0x0000_0000
MMUCFG(4)	0x0000_0003(5)
MSR	0x0000_0000
PID0	0x0000_0000
PIR	0x0000_0000
PVR(4)	0x817F_0000
SPRG0	Unaffected(2)
SPRG1	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 **POR** assertion, unchanged on **external PORESET** assertion.
3. Reset by processor reset **external PORESET** if DBCR0[EDM] = 0, as well as unconditionally by **POR**.
4. Read-only registers.
5. MMU Architecture version 3 and a null MMU.

7.2.2 Special purpose register (SPR) summary

PowerISA 2.06 and implementation-specific SPRs for the e200z0h cores are listed in *Table 26*. 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 may be read or written with the **mfspr** and **mtspr** instructions. In the instruction syntax, compilers should recognize the mnemonic name given in *Table 26*.

Table 26. Special purpose registers

Mnemonic	Name	SPR Number	Access	Privileged	e200z specific
BUCSR	Branch Unit Control and Status Register	1013	R/W	Yes	Yes
CSRR0	Critical Save/Restore Register 0	58	R/W	Yes	No
CSRR1	Critical Save/Restore Register 1	59	R/W	Yes	No
CTR	Count Register	9	R/W	No	No
DAC1	Data Address Compare 1	316	R/W	Yes	No
DAC2	Data Address Compare 2	317	R/W	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
DBSR	Debug Status Register	304	Read/Clear(1)	Yes	No
DEAR	Data Exception Address Register	61	R/W	Yes	No
DSRR0	Debug save/restore register 0	574	R/W	Yes	Yes
DSRR1	Debug save/restore register 1	575	R/W	Yes	Yes
ESR	Exception Syndrome Register	62	R/W	Yes	No
HID0	Hardware implementation dependent reg 0	1008	R/W	Yes	Yes
HID1	Hardware implementation dependent reg 1	1009	R/W	Yes	Yes
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
IVPR	Interrupt Vector Prefix Register	63	R/W	Yes	No
LR	Link Register	8	R/W	No	No
L1CFG0	L1 cache configuration register 0	515	Read-only	No	Yes
MCSR	Machine Check Syndrome Register	572	R/W	Yes	Yes
MMUCFG	MMU/MPU configuration register	1015	Read-only	Yes	Yes
PID0	Process ID Register	48	R/W	Yes	No
PIR	Processor ID Register	286	Read-only	Yes	No
PVR	Processor Version Register	287	Read-only	Yes	No
SPRG0	SPR General 0	272	R/W	Yes	No

Table 26. Special purpose registers(Continued)

Mnemonic	Name	SPR Number	Access	Privileged	e200z specific
SPRG1	SPR General 1	273	R/W	Yes	No
SRR0	Save/Restore Register 0	26	R/W	Yes	No
SRR1	Save/Restore Register 1	27	R/W	Yes	No
SVR	System Version Register	1023	Read-only	Yes	Yes
XER	Integer Exception Register	1	R/W	No	No

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*.

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 13: System Integration Unit Lite2 (SIUL2)*.

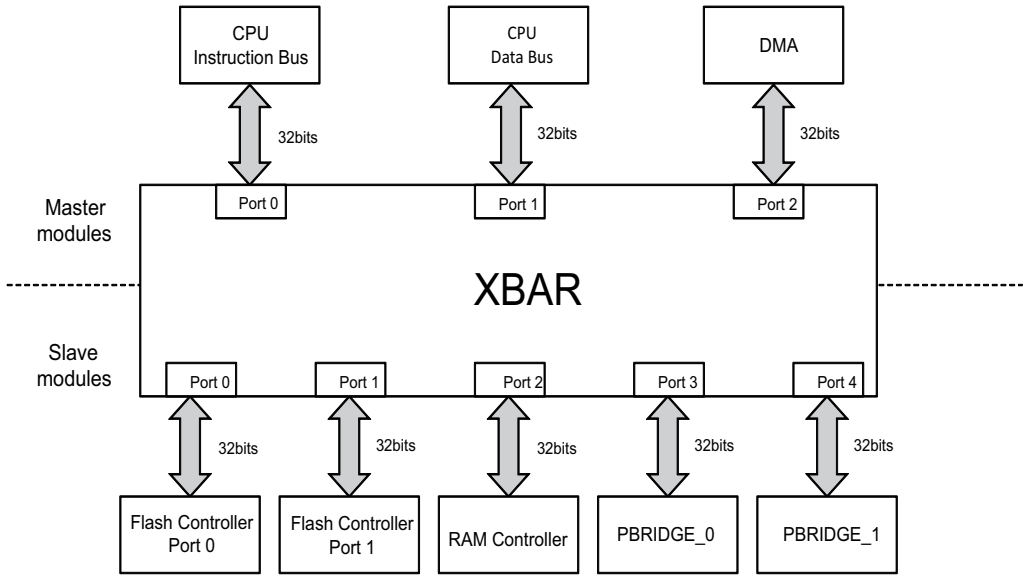
7.3.2 Crossbar switch

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 14: Crossbar switch (XBAR)*.

Figure 13. Crossbar switch integration



Note: For Nexus connections, refer to *Figure 675: Debug architecture*

Table 27. Reference links to related information

Topic	Related module	Reference
Full description	Crossbar switch	*Chapter 14: Crossbar switch (XBAR)*
Clocking	—	*Chapter 23: Clocking*
Crossbar switch master	Cores	*Section 7.2: Core modules*
	DMA	*Chapter 19: Enhanced direct memory access (eDMA)*
Crossbar switch slave	Flash memory controller	*Section 3.3.1: Flash memory controller*
	System RAM	*Section 3.2: 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 28 contains the master assignments.

Table 28. Crossbar switch master assignments

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

Table 29. Master ID number assignment

Master	Master ID number
CPU 0 (Instruction/Data)	0
DMA	1
CPU debug (Nexus)	2

7.3.2.1.2 Crossbar switch slave assignments

This device contains five slaves connected to the crossbar switch.

Table 30 contains the slave assignments.

Table 30. Crossbar switch slave assignments

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

Access to the flash memory via slave ports 0 and 1 is routed based on the master. The master-to-flash memory port routing have been assigned in a way to balance data traffic, with attention paid to facilitate core instruction fetches. *Table 31* shows the flash memory port assignments.

Table 31. Flash memory port assignments

Flash memory slave port 0	Flash memory slave port 1
e200z0 CPU DBUS (M1)	e200z0 CPU IBUS (M0)
DMA Master (M2)	—

7.3.2.1.3 XBAR_PRS*n* register reset values

The XBAR PRS*n* registers reset to 0000_0123h. Note that these registers are read-only.

Table 32. Reference links to related information

Topic	Related module	Reference
Full description	Crossbar switch	*Chapter 14: Crossbar switch (XBAR)*
System memory map	—	*Chapter 5: Memory map*
Clocking	—	*Chapter 23: Clocking*
Crossbar switch master	Cores	*Section 7.2: Core modules*
	DMA	*Section 7.3.6: DMA controller configuration*
Crossbar switch slave	Peripheral bridge	*Chapter 16: Peripheral bridge*

7.3.3 System Memory Protection Unit (SMPU) configuration

This chip contains one instance of the System Memory Protection Unit (SMPU):

- SMPU_0 for the Computational Shell

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 14. SMPU_0 configuration

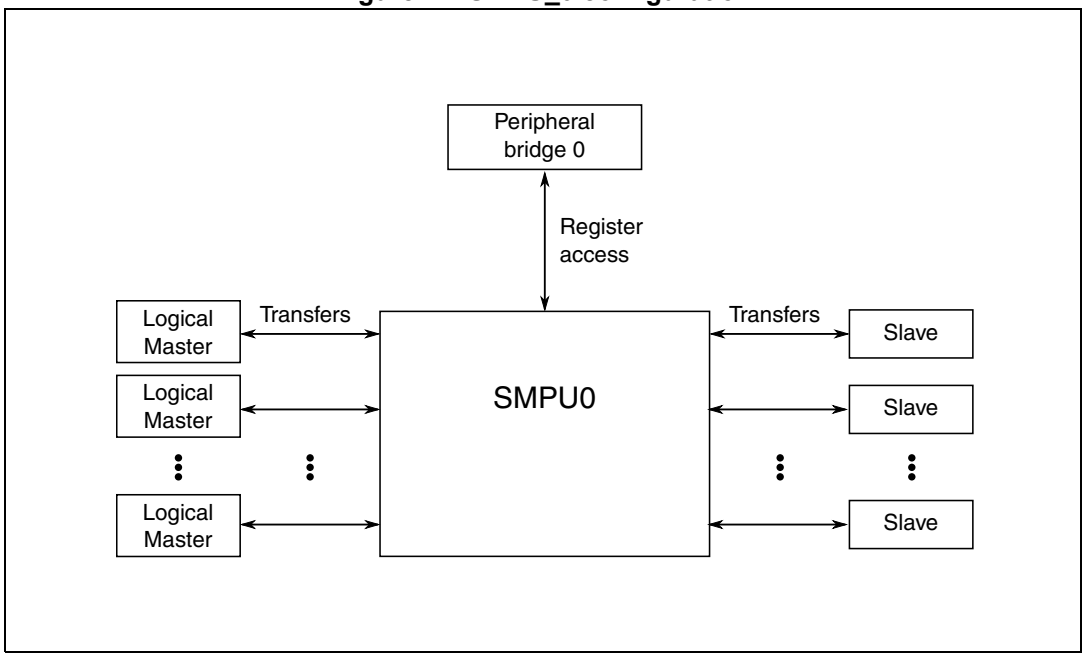


Table 33. Reference links to related information

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 23: Clocking*
Power management	—	*Chapter 10: Power management*
Logical masters	—	*Section 7.3.3.1.1*
Region descriptors	—	*Section 7.3.3.1.2*
RGD_WORD2 formats	—	*Section 7.3.3.1.3*

7.3.3.1.1 SMPU_0 logical bus master assignments

Table 34 contains the logical bus master assignments for the SMPU_0.

Table 34. SMPU_0 logical bus master assignments

SMPU logical bus master number	Bus master
0	CPU
1	DMA
2	CPU Debug

On this device, the SMPU (SMPU_0) is attached to the slave side of XBAR_0.

The SMPU port allocation is shown in *Table 35*.

Table 35. SMPU port allocation

Slave port number	IP
0	Flash controller port 0
1	Flash controller port 1
2	RAM controller
3	PBRIDGE_0
4	PBRIDGE_1

7.3.3.1.2 Number of region descriptors supported by SMPU_0

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

7.3.3.1.3 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.

7.3.3.1.4 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 15. Peripheral bridge configuration

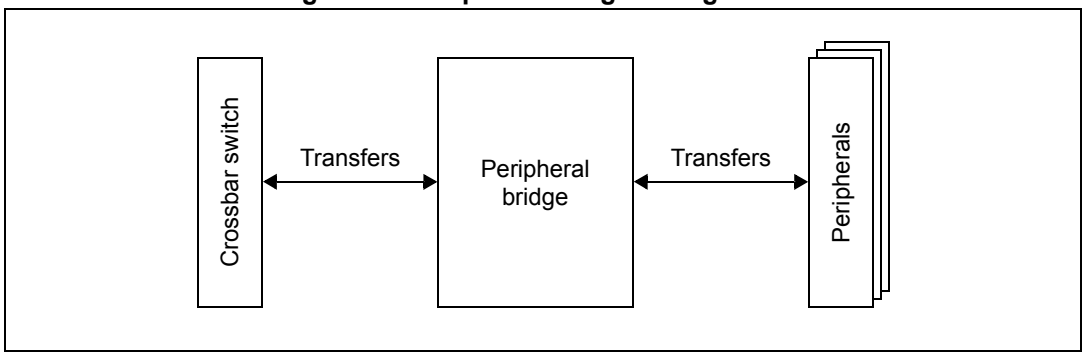


Table 36. Reference links to related information

Topic	Related module	Reference
Full description	Peripheral bridge	*Chapter 16: Peripheral bridge*
System memory map	—	*Chapter 5: Memory map*
Clocking	—	*Chapter 23: 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.

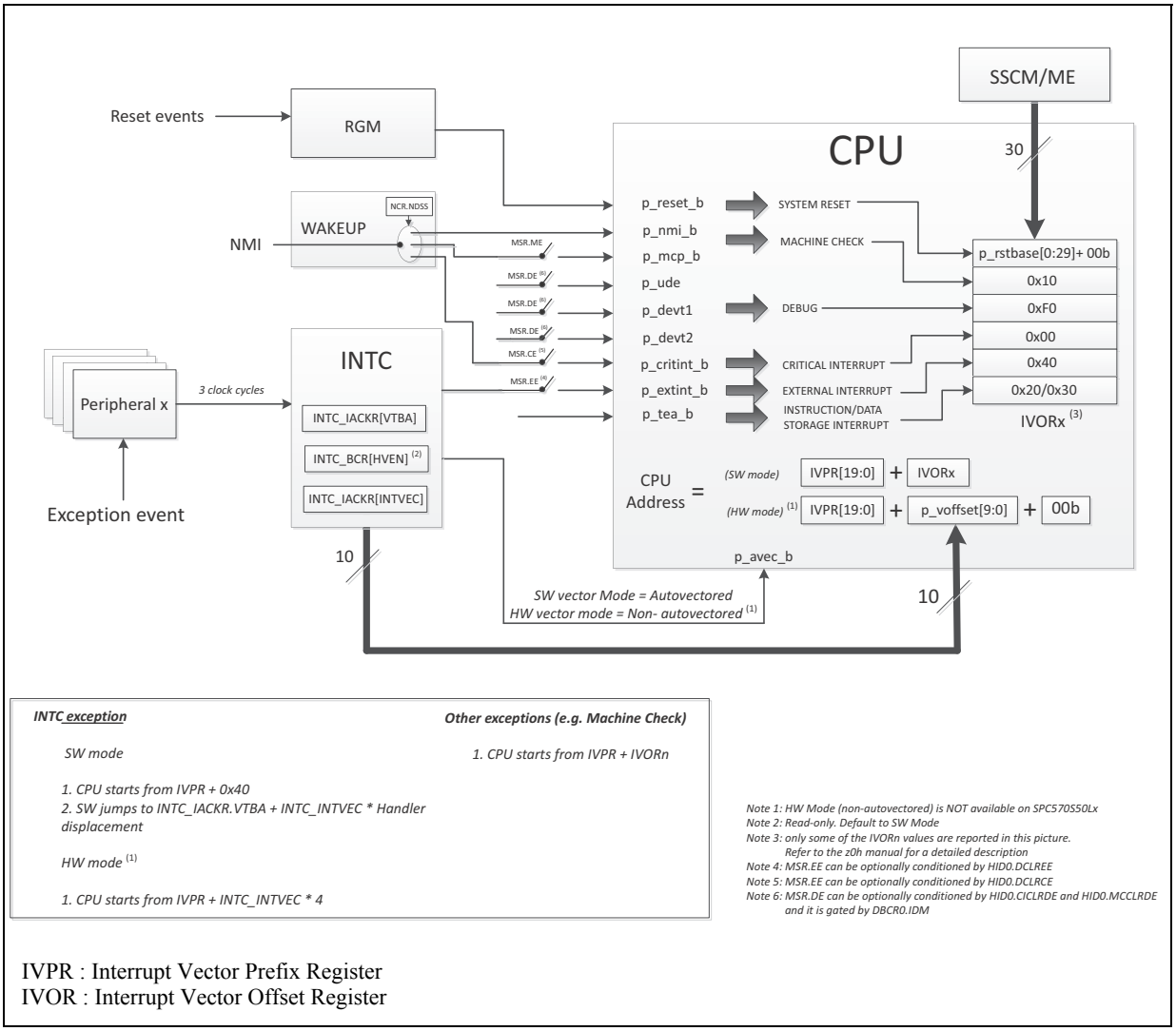
7.3.5 Interrupt controller (INTC) configuration

7.3.5.1 INTC instantiations

Two interrupt controllers, operating in lockstep with the main core and safety core, are present on this chip.

The INTC/NMI/CPU connectivity is illustrated in the following figure.

Figure 16. Interrupt scheme



IVPR : Interrupt Vector Prefix Register
IVOR : Interrupt Vector Offset Register

7.3.5.2 INTC implemented registers

Table 37 shows the registers implemented on the INTC on this chip.

Table 37. INTC implemented registers

Address offset	Register
0x0000	INTC Block Configuration Register (INTC_BCR)
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)

Table 37. INTC implemented registers(Continued)

Address offset	Register
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 sources

Table 38 defines the interrupt sources for the INTC on this chip.

Table 38. Interrupt sources

IRQ #	Source description	Source name
0	Software settable flag 0	SSCIR0_3[CLR0]
1	Software settable flag 1	SSCIR0_3[CLR1]
2	Software settable flag 2	SSCIR0_3[CLR2]
3	Software settable flag 3	SSCIR0_3[CLR3]
4	Software settable flag 4	SSCIR4_7[CLR4]
5	Software settable flag 5	SSCIR4_7[CLR5]
6	Software settable flag 6	SSCIR4_7[CLR6]
7	Software settable flag 7	SSCIR4_7[CLR7]
8	Software settable flag 8	SSCIR8_11[CLR8]
9	Software settable flag 9	SSCIR8_11[CLR9]
10	Software settable flag 10	SSCIR8_11[CLR10]
11	Software settable flag 11	SSCIR8_11[CLR11]
12	Software settable flag 12	SSCIR12_15[CLR12]
13	Software settable flag 13	SSCIR12_15[CLR13]
14	Software settable flag 14	SSCIR12_15[CLR14]
15	Software settable flag 15	SSCIR12_15[CLR15]
16	Software settable flag 16	SSCIR16_19[CLR16]
17	Software settable flag 17	SSCIR16_19[CLR17]
18	Software settable flag 18	SSCIR16_19[CLR18]
19	Software settable flag 19	SSCIR16_19[CLR19]
20	Software settable flag 20	SSCIR20_23[CLR20]
21	Software settable flag 21	SSCIR20_23[CLR21]
22	Software settable flag 22	SSCIR20_23[CLR22]
23	Software settable flag 23	SSCIR20_23[CLR23]
24	Software settable flag 24	SSCIR24_27[CLR24]
25	Software settable flag 25	SSCIR24_27[CLR25]

Table 38. Interrupt sources(Continued)

IRQ #	Source description	Source name
26	Software settable flag 26	SSCIR24_27[CLR26]
27	Software settable flag 27	SSCIR24_27[CLR27]
28	Software settable flag 28	SSCIR28_31[CLR28]
29	Software settable flag 29	SSCIR28_31[CLR29]
30	Software settable flag 30	SSCIR28_31[CLR30]
31	Software settable flag 31	SSCIR28_31[CLR31]
32–33	Reserved	
34	Watchdog timer	SWT_IR[TIF]
35–43	Reserved	
44	Platform periodic timer (STM) channel 0	STM_CIR0[CIF]
45	Platform periodic timer (STM) channel 1	STM_CIR1[CIF]
46	Platform periodic timer (STM) channel 2	STM_CIR2[CIF]
47	Platform periodic timer (STM) channel 3	STM_CIR3[CIF]
48–51	Reserved	
52	eDMA Combined Channel Error Flags 15–0	eDMA Channel Error Flags 15–0
53	eDMA Channel interrupt 0	DMA_INTL[INT0]
54	eDMA Channel interrupt 1	DMA_INTL[INT1]
55	eDMA Channel interrupt 2	DMA_INTL[INT2]
56	eDMA Channel interrupt 3	DMA_INTL[INT3]
57	eDMA Channel interrupt 4	DMA_INTL[INT4]
58	eDMA Channel interrupt 5	DMA_INTL[INT5]
59	eDMA Channel interrupt 6	DMA_INTL[INT6]
60	eDMA Channel interrupt 7	DMA_INTL[INT7]
61	eDMA Channel interrupt 8	DMA_INTL[INT8]
62	eDMA Channel interrupt 9	DMA_INTL[INT9]
63	eDMA Channel interrupt 10	DMA_INTL[INT10]
64	eDMA Channel interrupt 11	DMA_INTL[INT11]
65	eDMA Channel interrupt 12	DMA_INTL[INT12]
66	eDMA Channel interrupt 13	DMA_INTL[INT13]
67	eDMA Channel interrupt 14	DMA_INTL[INT14]
68	eDMA Channel interrupt 15	DMA_INTL[INT15]
69–184	Reserved	
185	Flash controller Program/Erase/Suspend IRQ_0	MCR[DONE]
186–225	Reserved	

Table 38. Interrupt sources(Continued)

IRQ #	Source description	Source name
226	PIT_0 Channel 0	PIT_0_TFLG0[TIF]
227	PIT_0 Channel 1	PIT_0_TFLG1[TIF]
228	PIT_0 Channel 2	PIT_0_TFLG2[TIF]
229	PIT_0 Channel 3	PIT_0_TFLG3[TIF]
230–241	Reserved	
242	XOSC counter	XOSC
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)
247–250	Reserved	
251	MC_ME Safe Mode interrupt	ME_IS[I_SAFE]
252	MC_ME Mode Transition interrupt	ME_IS[I_MTC]
253	MC_ME Invalid Mode interrupt	ME_IS[I_IMODE]
254	MC_ME Invalid Mode Configuration	ME_IS[I_ICONF]
255	MC_RGM 0	MC_RGM Functional and destructive reset alternate event interrupt
256–258	Reserved	
259	DSPI_0 combined overrun interrupt requests: – Transmit FIFO underflow – Receive FIFO overflow – Transmit FIFO invalid write flag	DSPI_0_SR[TFUF] | DSPI_0_SR[RFOF] | DSPI_0_SR[TFIWF]
260	DSPI_0 transmit FIFO end-of-queue flag	DSPI_0_SR[EOQF]
261	DSPI_0 Tx FIFO fill request flag	DSPI_0_SR[TFFF]
262	DSPI_0 transfer complete flag	DSPI_0_SR[TCF]
263	DSPI_0 Rx FIFO drain request flag	DSPI_0_SR[RFDF]
264	DSPI_0 Command Transfer Complete Flag	DSPI_0_SR[CMDTCF]
265	DSPI_0 Command FIFO Fill Flag	DSPI_0_SR[CMDFFF]
266	DSPI_0 SPI Parity Error Flags	DSPI_0_SR[SPEF]
267	Reserved	
268	DSPI_1 combined overrun interrupt requests: – Transmit FIFO underflow – Receive FIFO overflow – Transmit FIFO invalid write flag	DSPI_1_SR[TFUF] | DSPI_1_SR[RFOF] | DSPI_1_SR[TFIWF]
269	DSPI_1 transmit FIFO end-of-queue flag	DSPI_1_SR[EOQF]

Table 38. Interrupt sources(Continued)

IRQ #	Source description	Source name
270	DSPI_1 Tx FIFO fill request flag	DSPI_1_SR[TFFF]
271	DSPI_1 transfer complete flag	DSPI_1_SR[TCF]
272	DSPI_1 Rx FIFO drain request flag	DSPI_1_SR[RFDF]
273	DSPI_1 Command Transfer Complete Flag	DSPI_1_SR[CMDTCF]
274	DSPI_1 Command FIFO Fill Flag	DSPI_1_SR[CMDFFF]
275	DSPI_1 SPI Parity Error Flags	DSPI_1_SR[SPEF]
276	Reserved	
277	DSPI_2 combined overrun interrupt requests: – Transmit FIFO underflow – Receive FIFO overflow – Transmit FIFO invalid write flag	DSPI_2_SR[TFUF] | DSPI_2_SR[RFOF] | DSPI_2_SR[TFIWF]
278	DSPI_2 transmit FIFO end-of-queue flag	DSPI_2_SR[EOQF]
279	DSPI_2 Tx FIFO fill request flag	DSPI_2_SR[TFFF]
280	DSPI_2 transfer complete flag	DSPI_2_SR[TCF]
281	DSPI_2 Rx FIFO drain request flag	DSPI_2_SR[RFDF]
282	DSPI_2 Command Transfer Complete Flag	DSPI_2_SR[CMTCF]
283	DSPI_2 Command FIFO Fill Flag	DSPI_2_SR[CMDFFF]
284	DSPI_2 SPI Parity Error Flags	DSPI_2_SR[SPEF]
285–375	Reserved	-
376	LINFlexD_0 Reception combined interrupt: – Header Received – Data Reception complete – Data Buffer Full – Wakeup – LIN State (Sync Del, Sync Field, Identifier Field or Checksum Field)	LINFlexD_0_LINIER[HRIE] | LINFlexD_0_LINIER[DRIE] | LINFlexD_0_LINIER[DBFIE] | LINFlexD_0_LINIER[WUIE] | LINFlexD_0_LINIER[LSIE]
377	LINFlexD_0 Transmission combined interrupt: – Header Received – Data Transmitted – Data Buffer Empty	LINFlexD_0_LINIER[HRIE] | LINFlexD_0_LINIER[DTIE] | LINFlexD_0_LINIER[DBEIE]

Table 38. Interrupt sources(Continued)

IRQ #	Source description	Source name
378	LINFlexD_0 Error combined interrupt: – Stuck at zero – Output Compare – Bit Error – Checksum Error – Header Error – Frame Error – Buffer Overrun – UART Timeout Error	LINFlexD_0_LINIER[SZIE] | LINFlexD_0_LINIER[OCIE] | LINFlexD_0_LINIER[BEIE] | LINFlexD_0_LINIER[CEIE] | LINFlexD_0_LINIER[HEIE] | LINFlexD_0_LINIER[FEIE] | LINFlexD_0_LINIER[BOIE] | LINFlexD_0_LINIER[TOIE]
379–383	Reserved	
384	LINFlexD_1 Reception combined interrupt: – Header Received – Data Reception complete – Data Buffer Full – Wakeup – LIN State (Sync Del, Sync Field, Identifier Field or Checksum Field)	LINFlexD_1_LINIER[HRIE] | LINFlexD_1_LINIER[DRIE] | LINFlexD_1_LINIER[DBFIE] | LINFlexD_1_LINIER[WUIE] | LINFlexD_1_LINIER[LSIE]
385	LINFlexD_1 Transmission combined interrupt: – Header Received – Data Transmitted – Data Buffer Empty	LINFlexD_1_LINIER[HRIE] | LINFlexD_1_LINIER[DTIE] | LINFlexD_1_LINIER[DBEIE]
386	LINFlexD_1 Error combined interrupt: – Stuck at zero – Output Compare – Bit Error – Checksum Error – Header Error – Frame Error – Buffer Overrun – UART Timeout Error	LINFlexD_1_LINIER[SZIE] | LINFlexD_1_LINIER[OCIE] | LINFlexD_1_LINIER[BEIE] | LINFlexD_1_LINIER[CEIE] | LINFlexD_1_LINIER[HEIE] | LINFlexD_1_LINIER[FEIE] | LINFlexD_1_LINIER[BOIE] | LINFlexD_1_LINIER[TOIE]
387–476	Reserved	
477	Power Management Controller Supply Gauge Low Voltage Detect, High Voltage Detect Interrupt and USER BIST Interrupt	GR_S[VD11] | GR_S[VD14] | GR_S[VD13] | GR_S[VD6] | GR_S[VD3] | BIST_CTRL[IRQST]
478	Power Management Controller Temperature Sensor Interrupts	EPR_TD[TEMP_3] | EPR_TD[TEMP_2] | EPR_TD[TEMP_0]
479	Reserved	
480	PLL_0 Loss-of-lock interrupt request	PLL0SR[LOLF]
481		
482	PLL_1 Loss-of-lock interrupt request	PLL1SR[LOLF]
483	Reserved	

Table 38. Interrupt sources(Continued)

IRQ #	Source description	Source name
484	PLL_0 External Power Down Cycle Complete indication interrupt flag	PLL0SR[EXTPDF]
485	PLL_1 External Power Down Cycle Complete indication interrupt flag	PLL1SR[EXTPDF]
486–487(1)	Reserved	
487	PLL_1 PLL Clock Switch indication interrupt flag	PLL1SR[CKSWF]
488	FCCU_0	FCCU_IRQ_STAT[ALRM_STAT]
489	FCCU_1	FCCU_IRQ_STAT[CFG_TO_STAT]
490–491	Reserved	
492	FCCU_4	FCCU_OC_EOUT_TIMER
493–527	Reserved	
528	SARADC_0 combined interrupt: – 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) – WDGxL (WatchDog Low int) – WDGxH (WatchDog High int)	SARADC_0_ICIPR0[EOC_CH[15]] ..... SARADC_0_ICIPR0[EOC_CH[0]]
529–542	Reserved	
543	SARADC_B combined interrupt: – 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) – WDGxL (WatchDog Low int) – WDGxH (WatchDog High int)	SARADC_0_ICIPR0[EOC_CH[15] ..... SARADC_0_ICIPR0[EOC_CH[0]
544–673	Reserved	
674	JTAG Master combined interrupt: – Idle interrupt – Sequence Processing Unit interrupt	JTAGM_SR[SPU_INT] | JTAGM_SR[Idle] | EVTI0 | EVTI1
675	JTAG Data Communication combined interrupt: – JDC Input data received – JDC output data sent	JDC_MSR[JIN_INT] | JDC_MSR[JOUT_INT]
676	Reserved	
677	FlexCAN_0_MB0_3	FLEXCAN0_IFLAG1[0..3]

Table 38. Interrupt sources(Continued)

IRQ #	Source description	Source name
678	FlexCAN_0_MB4_7	FLEXCAN0_IFLAG1[4..7]
679	FlexCAN_0_MB8_11	FLEXCAN0_IFLAG1[8..11]
680	FlexCAN_0_MB12_15	FLEXCAN0_IFLAG1[12..15]
681	FlexCAN_0_MB16_31	FLEXCAN0_IFLAG1[16..31]
682–686	Reserved	
687	FlexCAN_0_ERR	FLEXCAN0_ESR[ERR_INT]
688	FlexCAN_0_BOFF	FLEXCAN0_ESR[BOFF_INT] | FLEXCAN0_ESR[TWRN_INT] | FLEXCAN0_ESR[RWRN_INT]
689	Reserved	
690	FlexCAN_1_MB0_3	FLEXCAN1_IFLAG1[0..3]
691	FlexCAN_1_MB4_7	FLEXCAN1_IFLAG1[4..7]
692	FlexCAN_1_MB8_11	FLEXCAN1_IFLAG1[8..11]
693	FlexCAN_1_MB12_15	FLEXCAN1_IFLAG1[12..15]
694	FlexCAN_1_MB16_31	FLEXCAN1_IFLAG1[16..31]
695–699	Reserved	
700	FlexCAN_1_ERR	FLEXCAN1_ESR[ERR_INT]
701	FlexCAN_1_BOFF	FLEXCAN1_ESR[BOFF_INT] | FLEXCAN1_ESR[TWRN_INT] | FLEXCAN1_ESR[RWRN_INT]
702–705	Reserved	
706	eTimer 0 Channel 0	eTimer0 STS 0
707	eTimer 0 Channel 1	eTimer0 STS 1
708	eTimer 0 Channel 2	eTimer0 STS 2
709	eTimer 0 Channel 3	eTimer0 STS 3
710	eTimer 0 Channel 4	eTimer0 STS 4
711	eTimer 0 Channel 5	eTimer0 STS 5
712–713	Reserved	
714	eTimer 0 Watchdog	eTimer0 WT
715	eTimer 0	eTimer0 RCF
716	Reserved	
717	eTimer 1 Channel 0	eTimer1 STS 0
718	eTimer 1 Channel 1	eTimer1 STS 1
719	eTimer 1 Channel 2	eTimer1 STS 2
720	eTimer 1 Channel 3	eTimer1 STS 3
721	eTimer 1 Channel 4	eTimer1 STS 4

Table 38. Interrupt sources(Continued)

IRQ #	Source description	Source name
722	eTimer 1 Channel 5	eTimer1 STS 5
723–724	Reserved	
725	eTimer 1 Watchdog	eTimer1 WT
726	eTimer 1	eTimer1 RCF
727	Reserved	
728	eTimer 2 Channel 0	eTimer2 STS 0
729	eTimer 2 Channel 1	eTimer2 STS 1
730	eTimer 2 Channel 2	eTimer2 STS 2
731	eTimer 2 Channel 3	eTimer2 STS 3
732	eTimer 2 Channel 4	eTimer2 STS 4
733	eTimer 2 Channel 5	eTimer2 STS 5
734–735	Reserved	
736	eTimer 2 Watchdog	eTimer2 WT
737	eTimer 2	eTimer2 RCF
738	Reserved	
739	eTimer 3 Channel 0	eTimer3 STS 0
740	eTimer 3 Channel 1	eTimer3 STS 1
741	eTimer 3 Channel 2	eTimer3 STS 2
742	eTimer 3 Channel 3	eTimer3 STS 3
743	eTimer 3 Channel 4	eTimer3 STS 4
744	eTimer 3 Channel 5	eTimer3 STS 5
745–746	Reserved	
747	eTimer 3 Watchdog	eTimer3 WT
748	eTimer 3	eTimer3 RCF
749–799	Reserved	
800	CTU 0	MRS event
801	CTU 0	ADC command
802	CTU 0	IEE
803	CTU 0 Timer 0	CTU0_T0
804	CTU 0 Timer 1	CTU0_T1
805	CTU 0 Timer 2	CTU0_T2
806	CTU 0 Timer 3	CTU0_T3
807	CTU 0 Timer 4	CTU0_T4
808	CTU 0 Timer 5	CTU0_T5
809	CTU 0 Timer 6	CTU0_T6

Table 38. Interrupt sources(Continued)

IRQ #	Source description	Source name
810	CTU 0 Timer 7	CTU0_T7
811	CTU 0 FIFO 0	CTU0_FIFO0
812	CTU 0 FIFO 1	CTU0_FIFO1
813	CTU 0 FIFO 2	CTU0_FIFO2
814	CTU 0 FIFO 3	CTU0_FIFO3
815–1023	Reserved	

1. Writes to these bits is possible but this will have no functional effect in the device.

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.

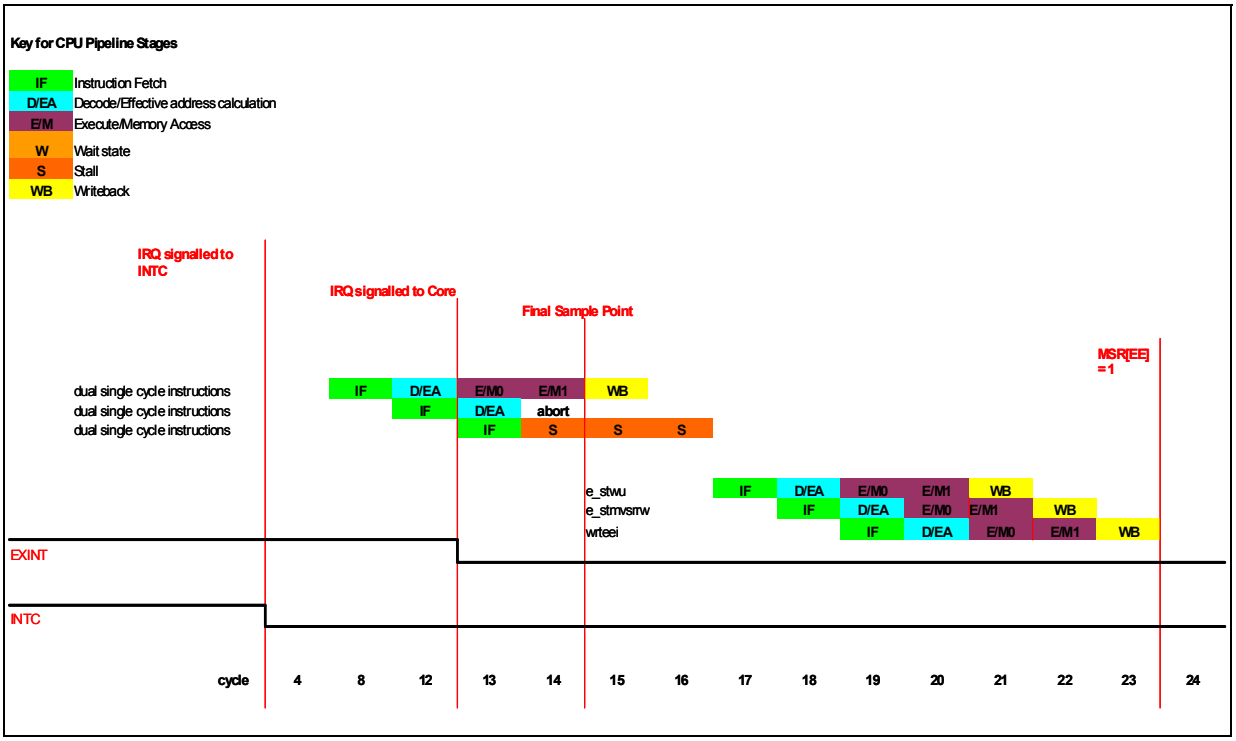
In general, the latency time is divided into three subsections:

- 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 SPC570Sx, the INTC operates at 40 MHz and the CPU operates at 80 MHz. Interrupt latency is:

- INTC Latency = 75 ns
- CPU Latency = 25 ns
- Software Latency = 43.75 ns
- Interrupt Latency Total = 143.75 ns

Figure 17. 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 18. DMA controller configuration

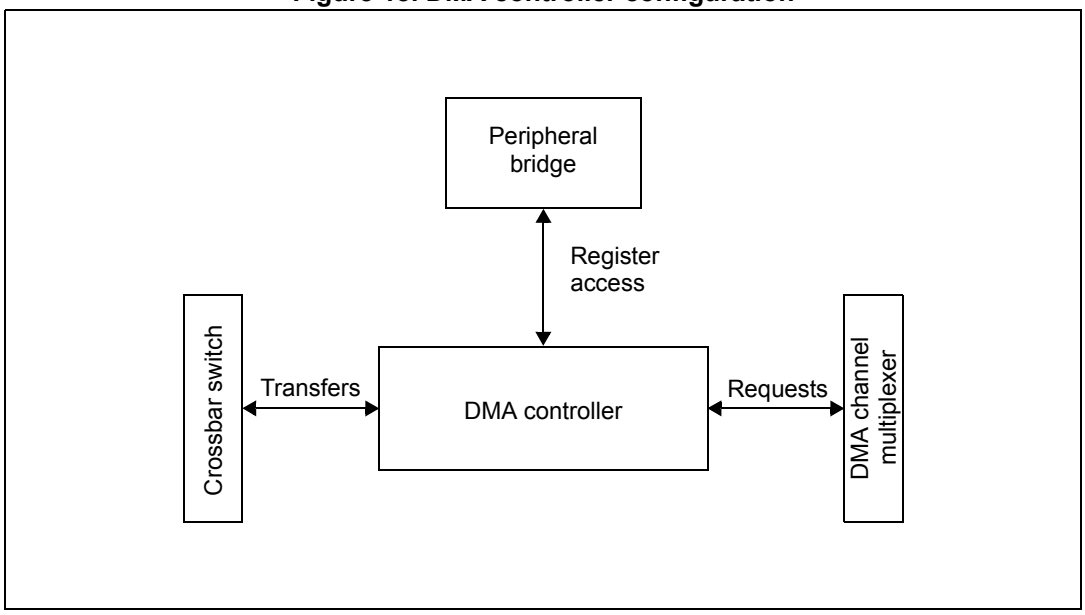


Table 39. Reference links to related information

Topic	Related module	Reference
Full description	DMA controller	*Chapter 19: Enhanced direct memory access (eDMA)*
System memory map	—	*Chapter 5: Memory map*
Register access	Peripheral bridge	*Chapter 16: Peripheral bridge*
Clocking	—	*Chapter 23: Clocking*
Transfers	Crossbar switch	*Chapter 14: 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.

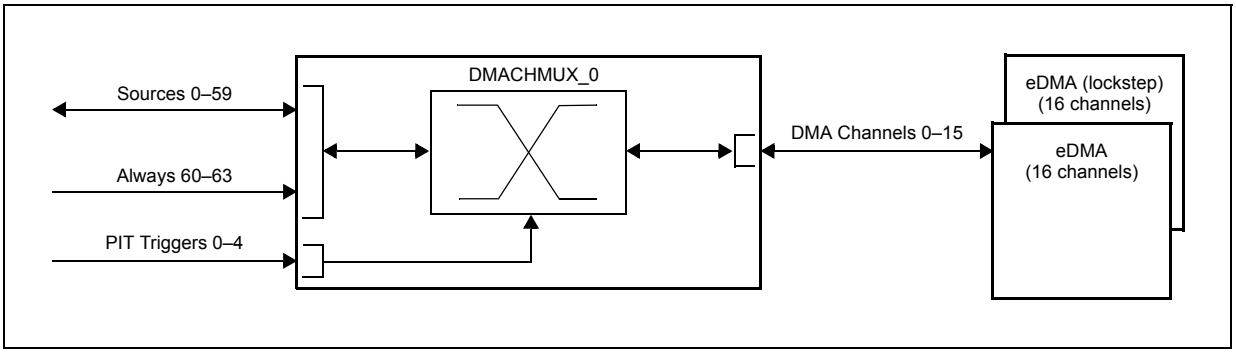
7.3.7.1 DMA channel assignment

SPC570Sx has one DMACHMUX. The DMACHMUX routes the DMA peripheral sources (called slots) to 16 DMA channels.

7.3.7.2 Block diagram

The connections of the DMA channel multiplexer to the eDMA are shown in *Figure 19*.

Figure 19. DMA channel multiplexing



7.3.7.3 DMA request mapping

The mapping of the peripheral DMA requests to the DMACHMUX input sources is shown in *Table 40*.

Table 40. DMAMUX peripheral DMA request to input source mapping

DMACHMUX source slot allocation (input trigger)	Peripheral	Event
0	Reserved	
1	DSPI 0	DSPI_SR[RFDF]
2	DSPI 0	DSPI_SR[TFFF]
3	DSPI 1	DSPI_SR[RFDF]

Table 40. DMAMUX peripheral DMA request to input source mapping(Continued)

DMACHMUX source slot allocation (input trigger)	Peripheral	Event
4	DSPI 1	DSPI_SR[TFFF]
5	DSPI 2	DSPI_SR[RFDF]
6	DSPI 2	DSPI_SR[TFFF]
7–8	Reserved	
9	LINFlexD 0	RX
10	LINFlexD 0	TX
11	LINFlexD 1	RX
12	LINFlexD 1	TX
13	SIUL2	REQ0
14	SIUL2	REQ1
15	SIUL2	REQ2
16	eTimer 0	REQ0
17	eTimer 0	REQ1
18–19	Reserved	
20	eTimer 1	REQ0
21	eTimer 1	REQ1
22–23	Reserved	
24	eTimer 2	REQ0
25	eTimer 2	REQ1
26–27	Reserved	
28	eTimer 3	REQ0
29	eTimer 3	REQ1
30–31	Reserved	
32	ADC	EOC
33	ADC (Supervisor)	EOC
34	CTU	MRS
35	CTU	FIFO0
36	CTU	FIFO1
37	CTU	FIFO2
38	CTU	FIFO3
39	FCCU	DMA trigger
40	DSPI 0	CMD
41	DSPI 1	CMD
42	DSPI 2	CMD

Table 40. DMAMUX peripheral DMA request to input source mapping(Continued)

DMACHMUX source slot allocation (input trigger)	Peripheral	Event
43	Reserved	
44	SIUL2	REQ3
45	SIUL2	REQ4
46	SIUL2	REQ5
47–59	Reserved	
60–63	—	Always ON

Note: Just as multiple CHCONFIG registers within the same instance must not select the same source value, CHCONFIG registers 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 41*.

Table 41. PIT periodic trigger to DMA channel mapping

PIT periodic trigger	DMA channel
0	0
1	1
2	2
3	3

7.3.7.4 DMACHMUX memory map

Chapter 20: DMA channel multiplexer (DMACHMUX) provides the DMACHMUX memory map.

7.3.8 Wakeup unit (WKPU) configuration

The WKPU supports 4 NMI sources. Please refer to *Chapter 21: Wakeup unit (WKPU)* for details.

NMI[0] and NMI[3] sources are tied together while NMI[1] and NMI[2] sources are tied to ‘1’ (refer to *Section 21.2.2.1: NMI Status Flag Register (NSR)*).

Although access to all the bits in the *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 23: 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: Embedded memories*
System memory map	—	*Chapter 5: Memory map*
Platform Flash	—	*Chapter 30: Flash memory controller (PFLASH Controller)*
Platform RAM controller	—	*Chapter 29: Platform RAM controller (PRAM)*
Embedded flash memory	—	*Chapter 31: 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, PFCR2, 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	Reserved
4	Reserved
5	Reserved
6	Reserved
7	Reserved

Table 43. Flash controller bus master assignments(Continued)

Bus Master Number	Bus Master
8	Reserved
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 30: Flash memory controller (PFLASH Controller)*
Flash memory architecture overview	Flash memory, Flash controller	*Section 3.3: Flash memory*
Flash memory	Flash memory	*Chapter 31: Embedded Flash Memory*
System memory map	—	*Chapter 5: Memory map*

7.6 Analog modules

This section highlights the following analog modules that are implemented on SPC570Sx:

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 22: Temperature sensor (TSENS)*
ADC subsystem	*Section 33.1: Overview*	—
Successive Approximation Register Analog-to-Digital Converters (SARADCs)	*Section 33.2: Configuration*	*Chapter 34: 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 SPC570Sx devices. In general, the temperature sensor generates an analog voltage that is linearly proportional to temperature. Refer to *Section 22.5: Obtaining device temperature using TSENS* in *Chapter 22: Temperature sensor (TSENS)*

7.6.2 SARADC configuration

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.

Chapter 33: Analog-to-digital converter (ADC) configuration provides configuration details for the SARADCs as well as an overview of the ADC subsystem, including a block diagram and a summary of analog input pin multiplexing.

7.6.3 CTU

The connections of the 13 input triggers of the CTU to the peripheral output modules are configured in the SIUL registers (refer to the Peripheral Muxing Description table in *Chapter 4: Signal description*).

The CTU contains four FIFOs to store ADC results. Each FIFO has sixteen 32-bit entries.

7.7 Timers

Four types of timers and a complex timer subsystem are implemented on SPC570Sx.

Table 46. Timers

Timer	Device-specific information	Block details
System Timer Module (STM)	*Section 7.7.1: System Timer Module (STM) configuration*	*Chapter 37: System Timer Module (STM)*
Software Watchdog Timer (SWT)	*Section 7.7.2: Software Watchdog Timer (SWT) configuration*	*Chapter 38: Software Watchdog Timer (SWT)*
Periodic Interrupt Timer (PIT)	*Section 7.7.3: PIT configuration*	*Chapter 39: Periodic Interrupt Timer (PIT)*
Enhanced Motor Control Timer (eTimer)	*Section 7.7.4: eTimers*	*Chapter 36: Enhanced Motor Control Timer (eTimer)*

The main core has a dedicated System Timer Module (STM) and a dedicated Software Watchdog Timer (SWT). The checker core does not have an STM or SWT.

One general-purpose Periodic Interrupt Timer (PIT) module is implemented. The PIT module comprises four 16-bit timer channels.

7.7.1 System Timer Module (STM) configuration

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 system clock divided by an 8-bit prescale value (1 to 256).

SPC570Sx includes one STM.

Table 47. 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

7.7.2 Software Watchdog Timer (SWT) configuration

SPC570Sx includes one SWT instance.

Table 48. Software Watchdog Timer (SWT) instances

Instance	Description
SWT_0	Intended for Main Core_0 in the high-speed computational clock domain.

The SWT has the following features:

- 32-bit time-out register to set the time-out period
- Internal oscillator clock for timer operation (IRCOSC)
- 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

7.7.2.1 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.

7.7.2.2 Default configuration

On SPC570Sx, the SWT comes out of reset with the default reset values summarized in *Table 49*. Minimum timeout value is 256 cycles.

Table 49. Register reset values

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

7.7.3 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 20. PIT configuration

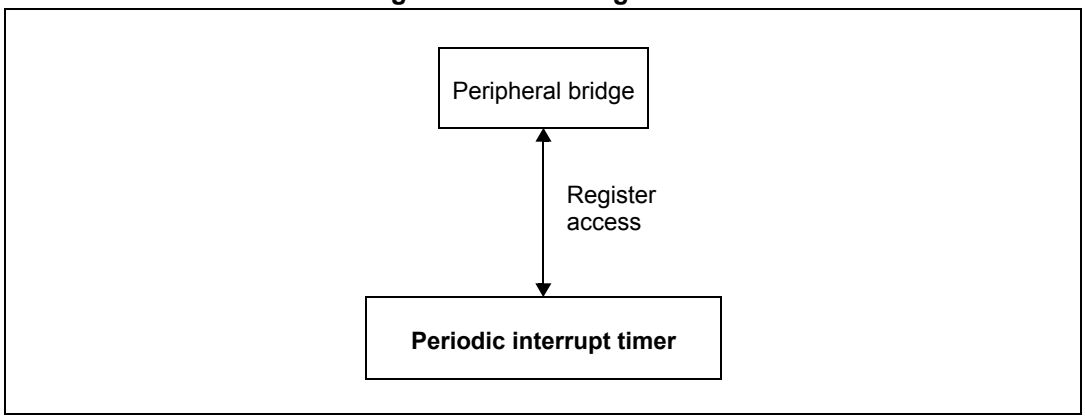


Table 50. Reference links to related information

Topic	Related module	Reference
Full description	PIT	*Chapter 39: Periodic Interrupt Timer (PIT)*
System memory map	—	*Chapter 5: Memory map*
Clocking	—	*Chapter 23: Clocking*
Power management	—	*Chapter 10: Power management*
DMA channel assignments	Direct memory access channel multiplexer	*Chapter 20: DMA channel multiplexer (DMACHMUX)*

7.7.3.1 PIT instantiation

Table 51. PIT information

PIT module instances	PIT channels	Description
1	4	The PIT has four standard channels. All PIT registers are cleared by any reset.

7.7.3.2 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: DMAMUX peripheral DMA request to input source mapping*.

7.7.3.3 PIT clocking

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

7.7.4 eTimers

7.7.4.1 eTimer configuration

SPC570Sx includes four 6-channel eTimer units. The following table provides the instantiation information for each module.

Table 52. eTimer instantiation

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

7.7.4.2 Fault inputs

The Fault inputs are not connected to external pins but to ETC outputs. See *Table 7: Peripheral multiplexing description* in *Chapter 4: Signal description* for more details.

7.7.4.3 eTimer registers

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

Table 53. eTimer registers memory map

Absolute address (hex)	Module
FFC2_3FFF	eTimer_0
FFC2_7FFF	eTimer_1
FBC2_0000	eTimer_2
FBC2_4000	eTimer_3

7.8 Communication interfaces

7.8.1 FlexCAN

7.8.1.1 FlexCAN instantiations

This chip implements two FlexCAN modules: FlexCAN_0 and FlexCAN_1.

7.8.1.2 FlexCAN configuration

FlexCAN data transmission and reception is handled through a mailbox system, which is composed of a set of up to 32 Message Buffers (MB) that store configuration and control data, time stamp, message ID and data.

The FlexCAN modules are compliant with the CAN protocol version 2.0 B. The bit rate can be programmed up to 1 Mbit/s.

7.8.2 DSPI configuration

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

DSPI module features:

- Transmission speed: 14 MHz
- Frame size: 8-bit and 16-bit
- 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 21. DSPI configuration

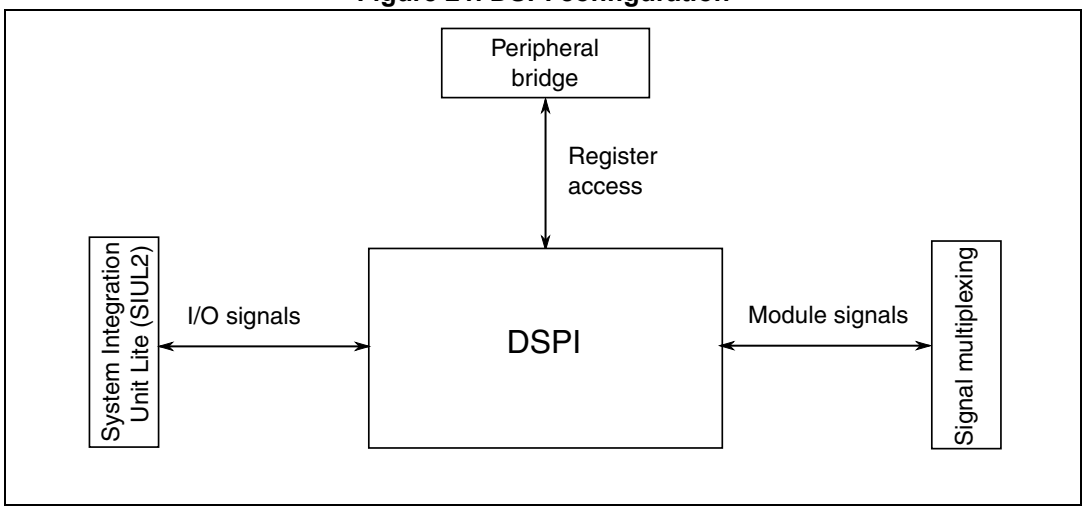


Table 54. Reference links to related information

Topic	Related module	Reference
Full description	DSPI	*Chapter 41: Deserial Serial Peripheral Interface (DSPI)*
System memory map	—	*Chapter 5: Memory map*
Clocking	—	*Chapter 23: Clocking*
Signal multiplexing	Port control	*Chapter 4: Signal description*
System module	System Integration Unit Lite	*Chapter 13: System Integration Unit Lite2 (SIUL2)*
DMA channel assignments	Direct memory access channel multiplexer	*Chapter 20: DMA channel multiplexer (DMACHMUX)*

7.8.2.1 DSPI instantiation

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

Table 55. DSPI instantiation

DSPI module	CTARs	TX FIFO depth	RX FIFO depth	PCS signals
DSPI_0	8	4	4	PCS[7:0]
DSPI_1	8	4	4	PCS[7:0]
DSPI_2	8	4	4	PCS[7:0]

7.8.2.2 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.

7.8.2.3 DSPI registers

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

Table 56. DSPI registers memory map

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

7.8.2.3.1 DSPI*x*_MCR implementation

Figure 22 shows the implementation of the DSPI*x*_MCR in this MCU.

Address: Base + 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	PCSIS[7:0]							
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	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	0

Figure 22. DSPI Module Configuration Register (DSPI*x*_MCR)

In this device, field DSPI_MCR[XSPI] **cannot** be configured to enable Extended SPI mode. DSPI_MCR[XSPI] value must remain at '0', which enables Normal SPI mode. In this mode, 8 and 16-bit SPI frames can be sent.

7.8.3 LINFlexD

7.8.3.1 LINFlexD – configurations

SPC570Sx has two instances of LINFlexD. *Table 57* lists the configuration for each instance.

Table 57. LINFlexD configurations

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

1. TX_CH_NUM = 0, Number of TX DMA channels = 2**TX_CH_NUM where ** stands for exponentiation.
2. RX_CH_NUM = 0, Number of RX DMA channels = 2**RX_CH_NUM where ** stands for exponentiation.

7.8.3.2 LINFlexD – implemented registers

Table 58 shows the registers implemented for each LINFlexD instance on the SPC570Sx chip.

Table 58. LINFlexD register comparison map

Register description	linflexd_0 0xFFE8C000 Address offset (hex)	linflexd_1 0xFFE90000 Address offset (hex)
LIN Control register 1 (LINCR1)	00	00
LIN Interrupt Enable register (LINIER)	04	04
LIN Status register (LINSR)	08	08
LIN Error Status register (LINESR)	0C	0C
UART Mode Control register (UARTCR)	10	10
UART Mode Status register (UARTSR)	14	14
LIN Timeout Control Status register (LINTCSR)	18	18
LIN Output Compare register (LINOCR)	1C	1C
LIN Timeout Control register (LINTOCR)	20	20
LIN Fractional Baud Rate register (LINFBRR)	24	24
LIN Integer Baud Rate register (LINIBRR)	28	28
LIN Checksum Field register (LINCFR)	2C	2C

Table 58. LINFlexD register comparison map(Continued)

Register description	linflexd_0 0xFFE8C000 Address offset (hex)	linflexd_1 0xFFE90000 Address offset (hex)
LIN Control register 2 (LINCR2)	30	30
Buffer Identifier register (BIDR)	34	34
Buffer Data Register Least Significant (BDRL)	38	38
Buffer Data Register Most Significant (BDRM)	3C	3C
Identifier Filter Enable register (IFER)	40	40
Identifier Filter Match index (IFMI)	44	44
Identifier Filter Mode register (IFMR)	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	—
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
UART Preset Timeout register (UARTPTO)	90	50
UART Current Timeout register (UARTCTO)	94	54
DMA Tx Enable register (DMATXE)	98	58
DMA Rx Enable register (DMARXE)	9C	5C

7.9 Reset and Boot modules

7.9.1 BAF configuration

7.9.1.1 Search boot header and boot options

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

7.9.1.2 LINFlexD and FlexCAN pins configuration

Table 59 shows the device pin configurations for LINFlexD_0 and FlexCAN_1 modules.

Table 59. LinFlexD and FlexCAN pin configuration

Pads	Initial Serial Boot Mode		Serial Boot Mode after a valid CAN message received	Serial Boot Mode after a valid LINFlexD message received
	Function	Pad Configuration	Function	Function
PA[12]	LINFlexD_0 Rx and FlexCAN_1 Rx	Input:— pull-down hysteresis TTL voltage level	FlexCAN_1 Rx	LINFlexD_0 Rx
PA[13]	LINFlexD_0 Tx	Output:— Push/Pull medium drive no pullup/pulldown	FlexCAN_1 Tx	LINFlexD_0 Tx

7.9.2 System Status and Configuration Module (SSCM) configuration

This section summarizes the System Status and Configuration Module (SSCM) configuration in the SPC570Sx device. For a comprehensive description of the SSCM, please refer to the SSCM's dedicated chapter.

7.9.2.1 SSCM instantiation

There is one SSCM instance in the SPC570Sx device.

7.9.2.2 Device-specific features

The device-specific reset values for the SSCM_MEMCONFIG register are shown in *Table 60*.

Table 60. SSCM_MEMCONFIG reset values

Field	Field definition	Value
MREV	Minor Revision	Minor silicon version
JPIN	JTAG Part ID	0A2h

Table 61 shows the address locations that the SSCM uses to search for startup information.

Table 61. 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

7.9.2.3 Protected registers

The SSCM registers are protected by a register protection module. After software has activated the protection for a register, writes can only be performed in supervisor mode until the protection is either deactivated by software or the device undergoes reset.

See the Register Protection chapter (*Chapter 62: Register Protection (REG_PROT)*) for a detailed description of the register protection mechanism.

7.9.3 Self-Test Control Unit (STCU2) configuration

Offline selftest is supported to work in the following configurations:

- no BIST (default)
- STCU bypass
- offline sequential LBIST
- offline parallel MBIST
- offline sequential LBIST + offline parallel MBIST

For LBIST run, RC clock (16 MHz) is to be used and for MBIST run, PLL clock (80MHz) is to be used.

Other BIST configurations are not supported.

7.10 Safety modules

7.10.1 CRC – Number of contexts

Table 62 lists the number of contexts available for the current CRC computation of multiple data streams of CRC.

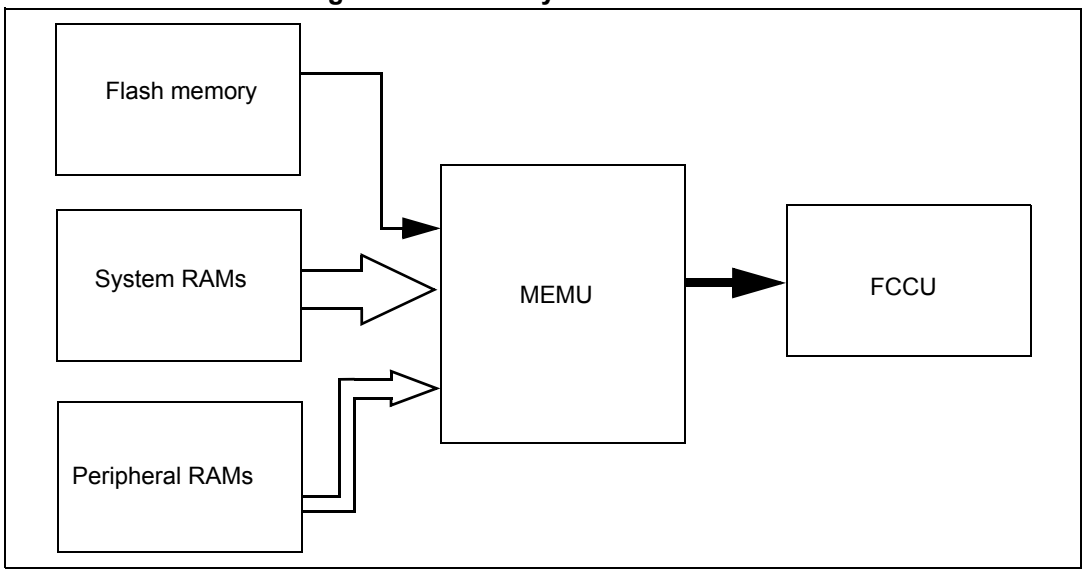
Table 62. CRC contexts

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

7.10.2 MEMU configuration

7.10.2.1 MEMU system connections

Figure 23. MEMU system connections



7.10.2.2 MEMU error sources

Table 63 shows the MEMU error sources for each instance configured on this chip.

Table 63. MEMU error sources

Instance	Value
Number of system RAM unique error sources	9
Number of peripheral RAM unique error sources	1
Number of flash memory unique error sources	2

7.10.2.3 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 64 describes the number of entries in each reporting table for the different error sources.

Table 64. 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

7.10.2.4 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 65. Corresponding error sources for system RAM OFLW0 register

OFLW0 bit-field position (bit)(1)	Error sources
0	SRAM
1	MBIST DMA
2	MBIST CAN0 MB
3	MBIST CAN1 MB
4	MBIST CAN0 RIXM
5	MBIST CAN1 RIXM
6	MBIST SRAM0
7	MBIST SRAM1
8	MBIST Flash RAM
9	Reserved
10	Reserved
11	Reserved
12	Reserved
13	Reserved
14	Reserved
15	Reserved
16	Reserved
17	Reserved
18	Reserved
19	Reserved
20	Reserved
21	Reserved
22	Reserved

Table 65. Corresponding error sources for system RAM OFLW0 register(Continued)

OFLW0 bit-field position (bit)[1]	Error sources
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 66. Corresponding error sources for peripheral RAM OFLW0 register

OFLW0 bit-field position (bit)[1]	Error sources
0	DMA
1	Reserved
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
19	Reserved
20	Reserved
21	Reserved

Table 66. Corresponding error sources for peripheral RAM OFLW0

OFLW0 bit-field position (bit)[1]	Error sources
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 67. 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
19	Reserved
20	Reserved

Table 67. Corresponding error sources for flash memory 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

7.10.2.5 MEMU registers

Table 68 shows the address offsets and implementation of the MEMU registers in this chip.

Table 68. 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)
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)

Table 68. MEMU registers(Continued)

Offset	Register
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)
0x0C2C	Flash memory correctable error reporting table address register (FLASH_CERR_ADDR1)
0x0C30	Flash memory correctable error reporting table status register (FLASH_CERR_STS2)

Table 68. MEMU registers(Continued)

Offset	Register
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)
0x0C94	Flash memory correctable error reporting table address register (FLASH_CERR_ADDR14)
0x0C98	Flash memory correctable error reporting table status register (FLASH_CERR_STS15)

Table 68. MEMU registers(Continued)

Offset	Register
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)

7.10.3 FCCU configuration

7.10.3.1 Available FCCU output signals

Table 69 shows the available FCCU output signals implemented on this chip.

Table 69. FCCU output signals

FCCU output signal name	Chip-top signal name
EOUT[0], FI[0], ERROR0	FCCU_F0
EOUT[1], FI[1], ERROR1	FCCU_F1

7.10.3.2 FCCU register reset values

Table 70 shows the reset values of selected FCCU registers.

Table 70. Reset values of selected FCCU registers

Register	Reset value
FCCU_RF_CFG0	0xFFFF_FFFF
FCCU_RF_CFG1	0xFFFF_FFFF
FCCU_RF_CFG2	0x0000_0000

Table 70. Reset values of selected FCCU registers(Continued)

Register	Reset value
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

7.10.3.3 FOSU parameters

The FOSU_COUNT, or FOSU timeout response time, is 1000 safe clock cycles.

7.10.4 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 62: Register Protection (REG_PROT)*.

Table 71. Reference links to related information

Topic	Related module	Reference
System memory map	—	*Chapter 5: Memory map*
CMU registers	CMU	*Chapter 25: Clock Monitor Unit (CMU)*
Clock Generation Module (ME_CGM) registers	MC_CGM	*Chapter 26: Clock Generation Module (MC_CGM)*
Mode Entry (MC_ME) registers	MC_ME	*Chapter 48: Mode Entry Module (MC_ME)*
Reset Generation Module (MC_RGM) registers	MC_RGM	*Chapter 43: Reset Generation Module (MC_RGM)*
Memory Error Management Unit (MEMU) registers	MEMU	*Chapter 59: Memory Error Management Unit (MEMU)*
Oscillator Digital Interface (OSC_DIG) registers	OSC_DIG	*Chapter 27: OSC Digital Interface (XOSC)*
PLL Digital Interface (PLL_DIG) registers	PLL_DIG	*Chapter 24: Dual PLL Digital Interface (PLLDIG)*
PMC Digital Interface (PMC_DIG) registers	PMC_DIG	*Chapter 46: Power management controller digital interface (PMC_Dig)*
SIUL2 registers	SIUL2	*Chapter 13: System Integration Unit Lite2 (SIUL2)*
System Status and Control Module (SSCM) registers	SSCM	*Chapter 45: System Status and Configuration Module (SSCM)*

7.10.4.1 SIUL2 protected registers

Table 72 lists the SIUL2 registers that can be protected.

Table 72. 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 1–79	32	—(1)	
MSCR 480—494	32	—(1)	32
MSCR 512–513	32	—(1)	32
MSCR 560–561	32	—(1)	32
MSCR 624–632	32	—(1)	32
MSCR 688–711	32	—(1)	32

Table 72. Protected SIUL2 registers(Continued)

Register	Register size (bits)	Offset from module base address	Protected size (bits)
MSCR 752–767	32	—(1)	32
MSCR 816–821	32	—(1)	32
MSCR 944	32	0x1100	32
SIUL2_GPDO0–79	8	—(1)	8
SIUL2_PGPDO0–4	16	—(1)	16
SIUL2_MPGPDO0–4	32	—(1)	32

1. See the module memory map in *Chapter 13: System Integration Unit Lite2 (SIUL2)* for address offset.

7.10.4.2 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 73* lists the CMU registers that can be protected.

Table 73. Protected CMU registers

Module	Register(1)	Register size (bits)	Offset from module base address	Protected size (bits)
CMU_0	CMU_CSR	32	0x0203	8
	CMU_HFREFR	32	0x0208	32
	CMU_LFREFR	32	0x020C	32
	CMU_MDR	32	0x0218	32
CMU_1	CMU_CSR	32	0x0203	8
	CMU_HFREFR	32	0x0208	32
	CMU_LFREFR	32	0x020C	32
	CMU_MDR	32	0x0218	32
CMU_2	CMU_CSR	32	0x0243	8
	CMU_HFREFR	32	0x0248	32
	CMU_LFREFR	32	0x024C	32
	CMU_MDR	32	0x0258	32
CMU_3	CMU_CSR	32	0x0283	8
	CMU_HFREFR	32	0x0288	32
	CMU_LFREFR	32	0x028C	32
	CMU_MDR	32	0x0298	32

1. See Register Protection chapter for bit field details.

7.10.4.3 PMC Digital Interface (PMC_DIG) protected registers

Table 74 lists the PMC_DIG registers that can be protected.

Table 74. 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
FEE_HV1	32	0x0470	32
MREG_CTL	32	0x0643	8
REE_TD	32	0x0704	32
RES_TD	32	0x0708	32
CTL_TD	32	0x070C	32
FEE_TD	32	0x0718	32

7.10.4.4 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 75* lists the PLL_DIG registers that can be write-protected.

Table 75. Protected PLL_DIG registers

Register[1]	Register size (bits)	Offset from module base address	Protected size (bits)
PLL0_CR	32	0x0100	32
PLL0_DV	32	0x0108	32
PLL1_CR	32	0x0120	32
PLL1_DV	32	0x0128	32
PLL1_FM	32	0x012C	32
PLL1_FD	32	0x0130	32

1. See Register Protection chapter for bit field details.

7.10.4.5 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 76* lists the Oscillator Digital Interface registers that can be write-protected.

Table 76. Protected XOSC registers

Register(1)	Register size (bits)	Offset from module base address	Protected size (bits)
XOSC_CTL	32	0x0080	32

1. See Register Protection chapter for bit field details.

7.10.4.6 Memory Error Management Unit (MEMU) protected registers

Table 77 lists the MEMU registers that can be protected.

Table 77. Protected MEMU registers

Register	Register size (bits)	Offset from module base address	Protected size (bits)
CTRL	8	0x000	8
DEBUG	32	0x00C	32

7.10.4.7 Mode Entry (MC_ME) protected registers

Table 78 lists the MC_ME registers that can be protected.

Table 78. 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

Table 78. Protected MC_ME registers(Continued)

Register	Register size (bits)	Offset from module base address	Protected size (bits)
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
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_PCTL30	8	0x0DE	8
ME_PCTL36	8	0x0E4	8
ME_PCTL38	8	0x0E6	8
ME_PCTL79	8	0x10F	8
ME_PCTL92	8	0x11C	8
ME_PCTL98	8	0x122	8
ME_PCTL99	8	0x123	8
ME_PCTL112	8	0x130	8
ME_PCTL113	8	0x131	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_PCTL207	8	0x18F	8
ME_PCTL220	8	0x19C	8
ME_PCTL227	8	0x1A3	8
ME_PCTL244	8	0x1B4	8
ME_CADDR0	32	0x1E0	32

7.10.4.8 Clock Generation Module (MC_CGM) protected registers

Table 79 lists the MC_CGM registers that can be protected.

Table 79. Protected MC_CGM registers

Register	Register size (bits)	Offset from module base address	Protected size (bits)
CGM_SC_DIV_RC	32	0x00D0	32
CGM_DIV_UPD_TYPE	32	0x00D4	32
CGM_DIV_UPD_TRIG	32	0x00D8	32
CGM_SC_DC0	32	0x00E8	32
CGM_SC_DC1	32	0x00EC	32
CGM_SC_DC2	32	0x00F0	32
CGM_AC0_SC	32	0x0100	8
CGM_AC0_DC0	32	0x0108	32
CGM_AC0_DC1	32	0x010C	32
CGM_AC0_DC2	32	0x0110	32
CGM_AC0_DC3	32	0x0114	32
CGM_AC0_DC4	32	0x0118	32
CGM_AC0_DC5	32	0x011C	32
CGM_AC1_SC	32	0x0120	8
CGM_AC1_DC0	32	0x0128	32
CGM_AC2_SC	32	0x0140	8
CGM_AC3_SC	32	0x0160	8

7.10.4.9 Reset Generation Module (MC_RGM) protected registers

Table 80 lists the MC_RGM registers that can be protected.

Table 80. 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_DBRE	32	0x030	32
RGM_FERD	32	0x310	32
RGM_FEAR	32	0x320	32
RGM_FBRE	32	0x330	32
RGM_FESS	32	0x340	32
RGM_PRST0	32	0x610	32

Table 80. Protected MC_RGM registers(Continued)

Register	Register size (bits)	Offset from module base address	Protected size (bits)
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
RGM_PRST6	32	0x628	32
RGM_PRST7	32	0x62C	32

7.10.4.10 System Status and Control Module (SSCM) protected registers

Table 81 lists the SSCM registers that can be protected.

Table 81. Protected SSCM registers

Register	Register size (bits)	Offset from module base address	Protected size (bits)
SSCM_ERROR	16	0x006	16

7.11 Safety modules

7.11.1 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 SPC570Sx 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 *Chapter 63: Password and Device Security Module (PASS)* for further details.

7.11.1.1 PASS_LOCK*n*_*PGn* 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 SPC570Sx 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.

Figure 24. PASS_LOCK0_PGn Register

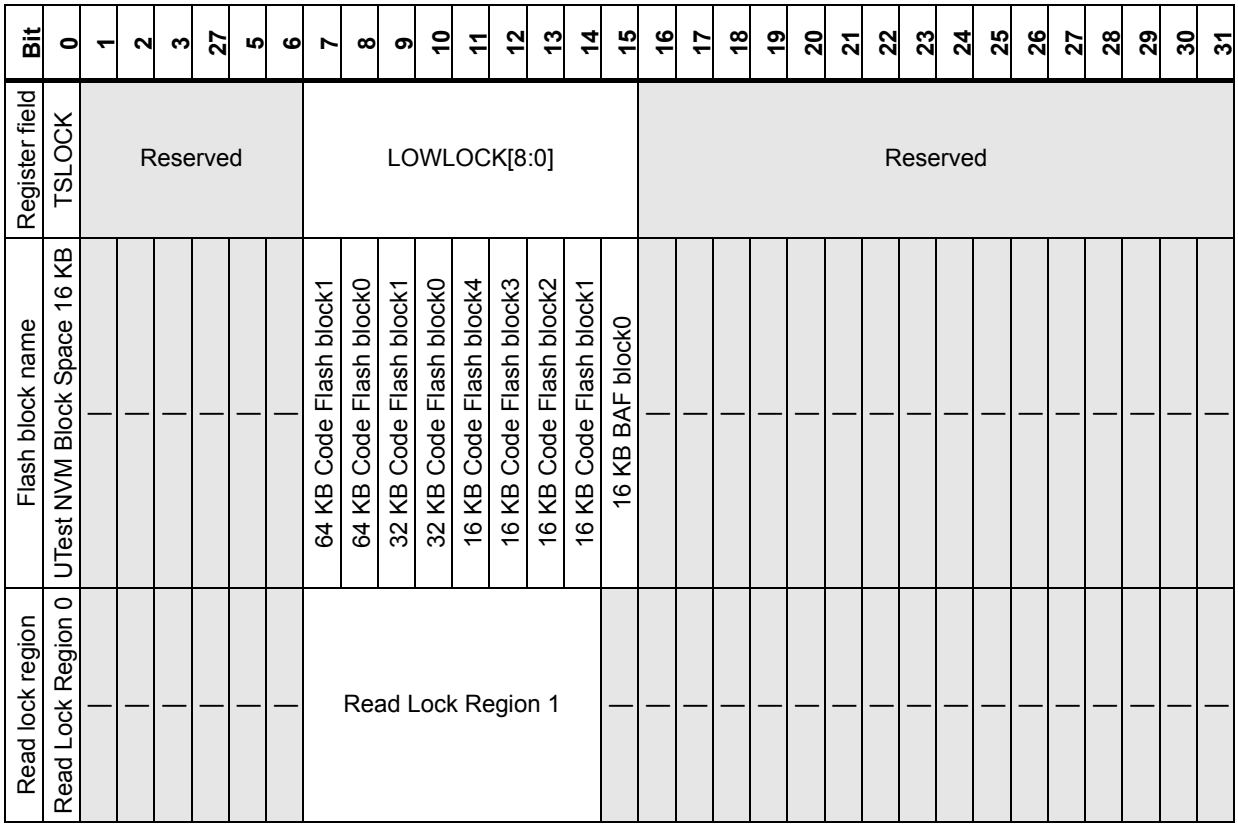


Figure 25. PASS_LOCK1_PGn Register

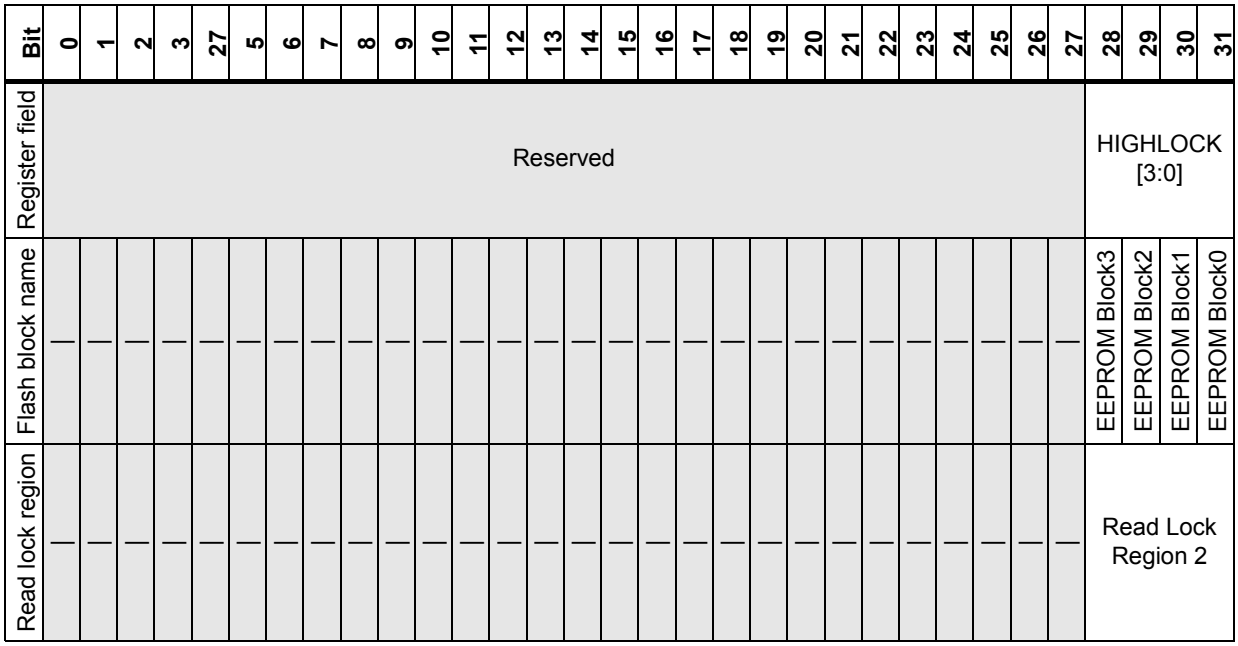
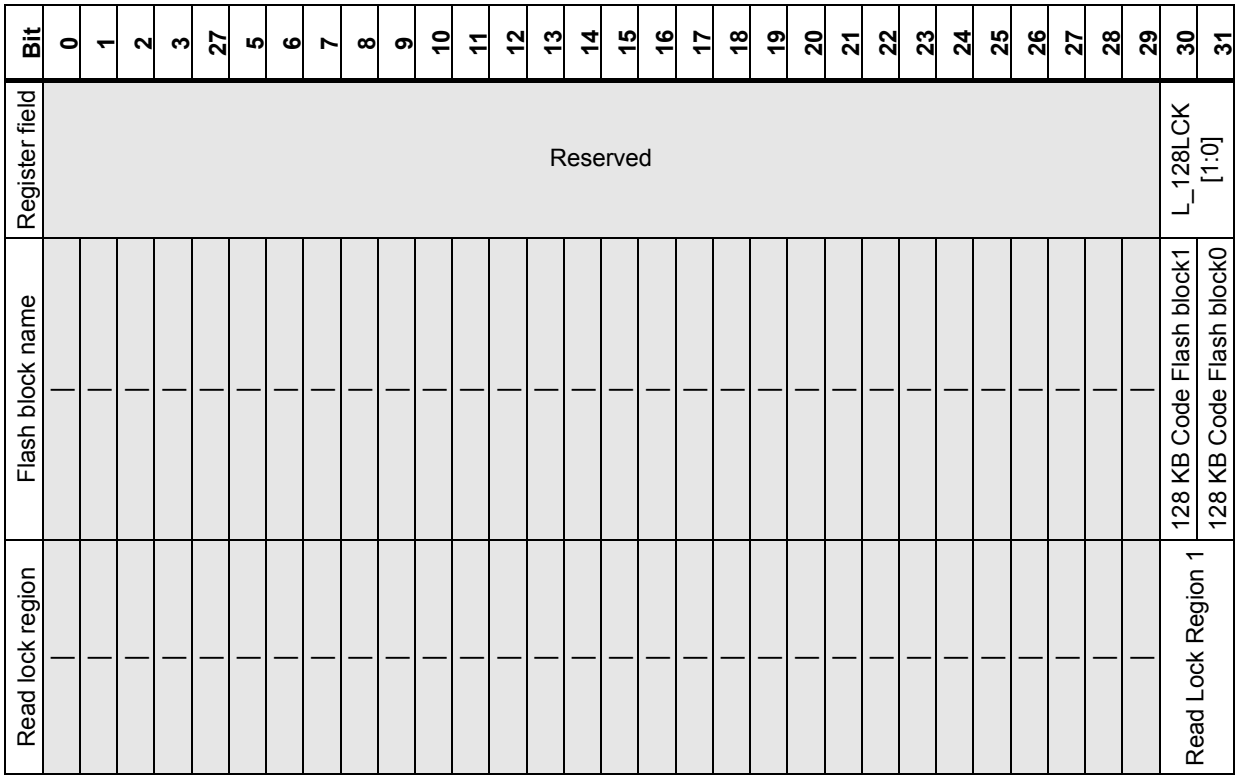


Figure 26. PASS_LOCK2_PGn Register



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																														L_128LCK [1:0]	
Flash block name	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	128 KB Code Flash block1	128 KB Code Flash block0
Read lock region	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	Read Lock Region 1	

Figure 27. PASS_LOCK3_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	PGL	DBL	MO	Reserved	MSTR[3:0]				Reserved					RL2	RL1	RL0	Reserved															
Flash block name	Password Group Lock	Debug Interface Lock	Master Only	—	Master Access				—	—	—	—	—	EEPROM Data Flash Read Lock	Code Flash Read Lock	UTEST Read Lock	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—

7.11.1.2 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 82*. These values provide the following capabilities:

- Set the current lifecycle state
- 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 82. 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
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

1. See *Section 63.5.3: Censoring and uncensoring the device* for values.
2. See *Section 7.11.1.1: PASS_LOCKn_PGn register bit mapping* for values.

7.11.2 Debug and trace

SPC570Sx microcontrollers have a comprehensive feature set to assist debug and trace. *Chapter 50: Debug and Trace* contains full details as well as the top-level block diagram (*Figure 675*) of the SPC570Sx debug architecture. The figure shows the general interconnection of functional debug modules.

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 46: Power management controller digital interface (PMC_Dig)*), the Reset generation Module (see *Chapter 43: Reset Generation Module (MC_RGM)*), the Mode Entry (see *Chapter 48: Mode Entry Module (MC_ME)*), the System Status and Configuration Module (see *Chapter 45: System Status and Configuration Module (SSCM)*) and the Self Test Configuration (see *Chapter 61: 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 83 shows the pin used for the reset system functionality.

Table 83. 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 43: 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 84 shows the various destructive reset events and *Table 85* 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 84 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 43: Reset Generation Module (MC_RGM)* for further details).

Table 84. 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 84. 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 85 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 43: Reset Generation Module (MC_RGM)* for further details).

Table 85. 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	Software device reset(5)	F[3]	P[E]	P[E]	P[D]	X	X
Reserved		F[4]	—	—	—	—	—
F_FCCU_HARD	Hard reaction request	F[5]	E	O	X	X	X

Table 85. 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
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

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(c).

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.

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 (d). Once 'destructive' reset escalation has been enabled, the RGM increases a counter on

c. The default value at reset for this register is 4

d. The default value at reset for this register is 8

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.

Figure 28. Reset sequence

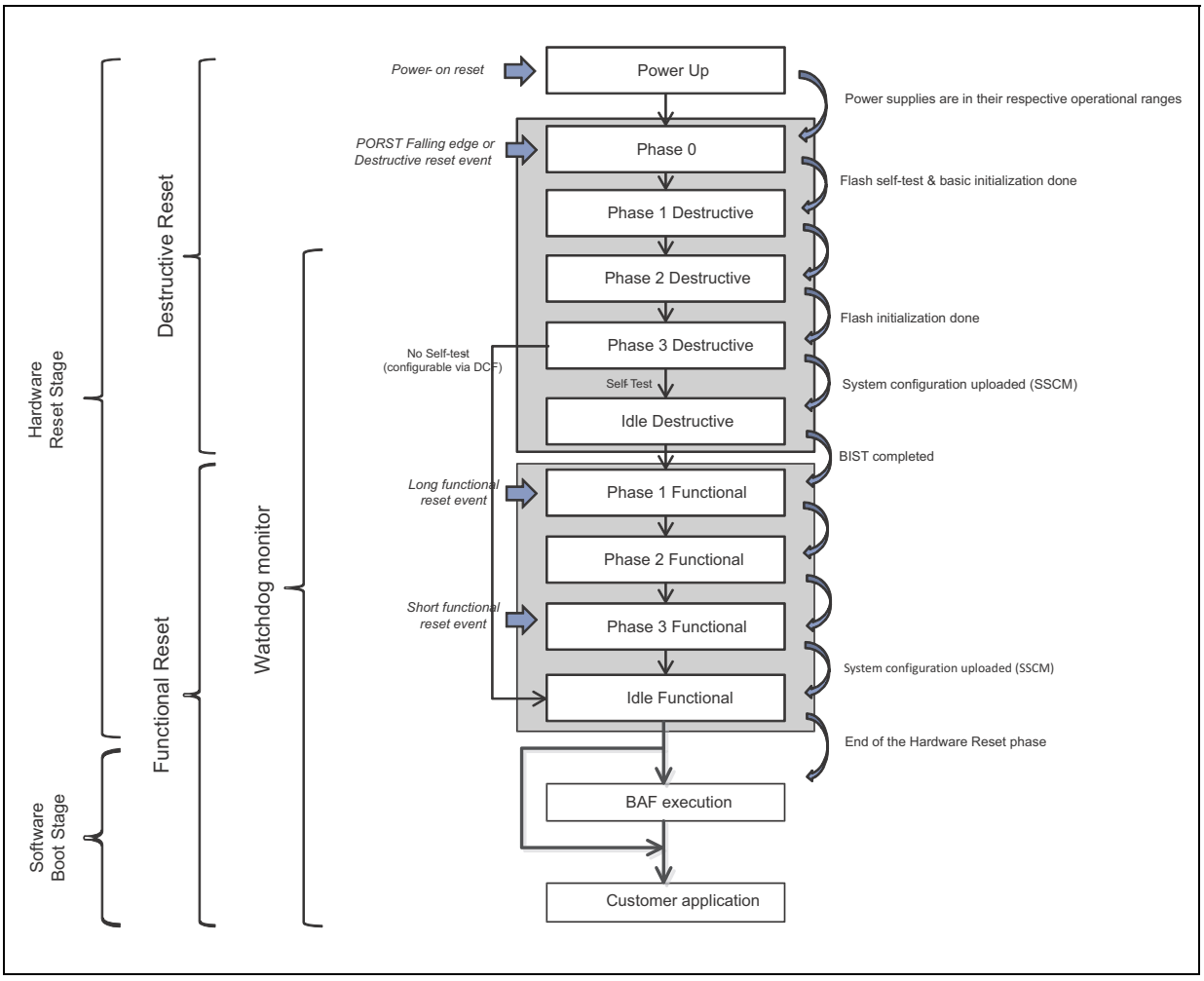


Table 86 summarizes the reset sequence phases.

Table 86. 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.
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 45: 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 87 shows the status of the various system modules during the hardware reset phase.

Table 87. 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
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 29* 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 29. PORS Reset pin vs. Phase 3 Destructive

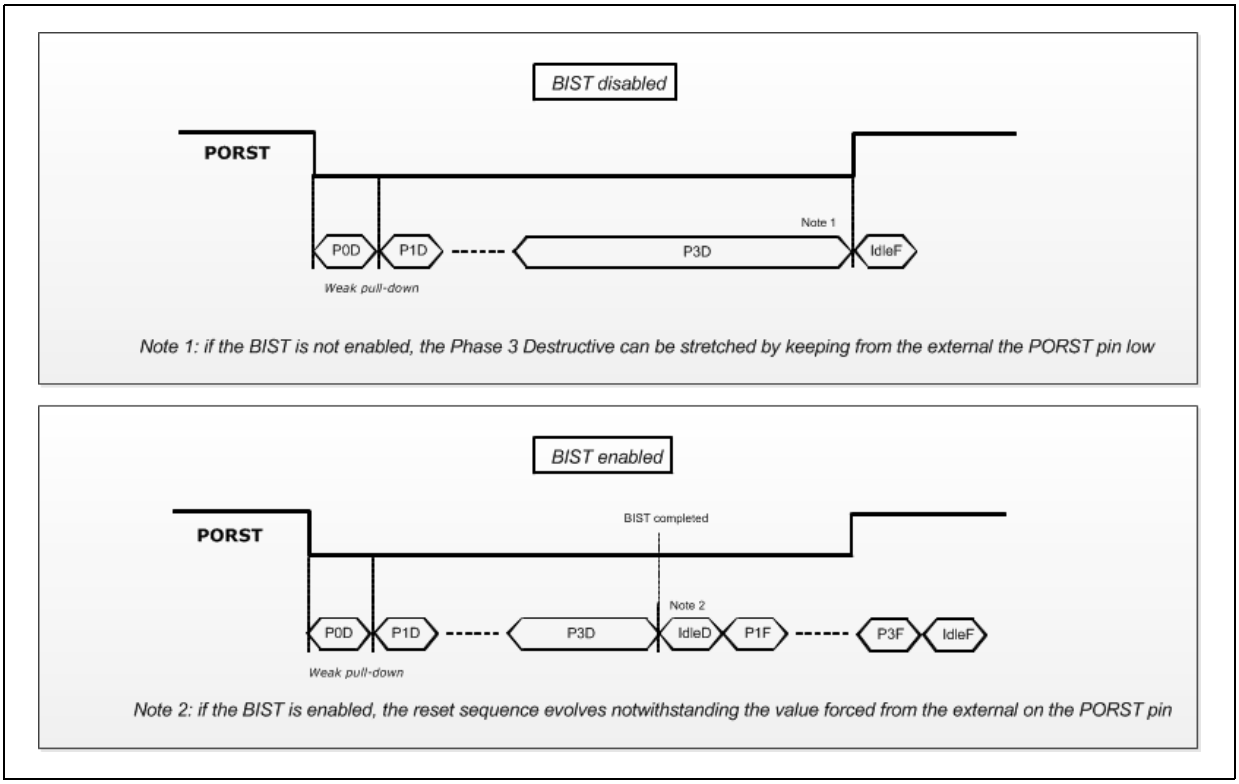


Figure 30 shows the effect of the PORST pin during the functional reset phase. It is possible to stretch the Phase 3 Functional by keeping PORST low. Note that this case is possible only when the BIST is enabled, otherwise the reset sequence would have been formerly stopped at the Phase 3 Destructive, as explained above.

Figure 30. PORS Reset pin vs. Phase 3 Functional

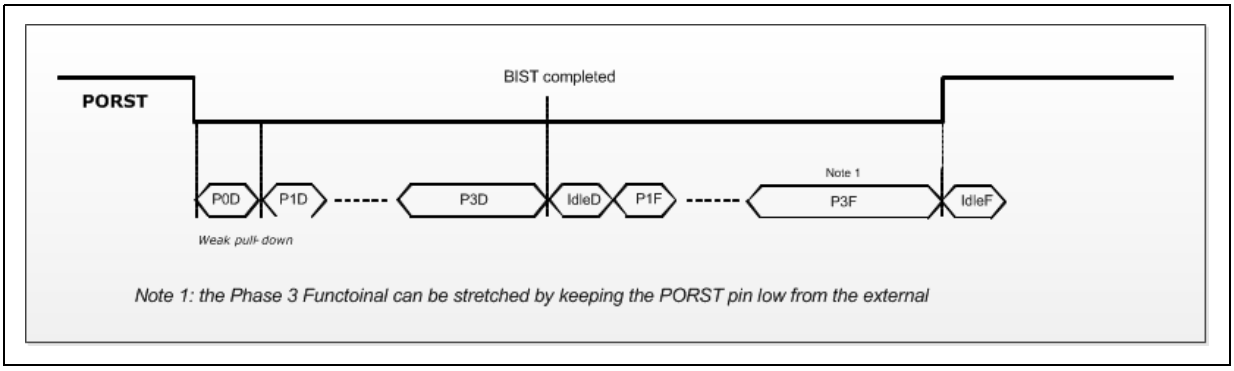


Table 88, *Table 89* and *Table 90* show PORST pin characteristics.

Table 88. 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 89. PORST filter characteristics

Description	Min	Max	Unit
Non filtered pulses	—	500	ns
Filtered pulses	2000	—	ns

Table 90. 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.

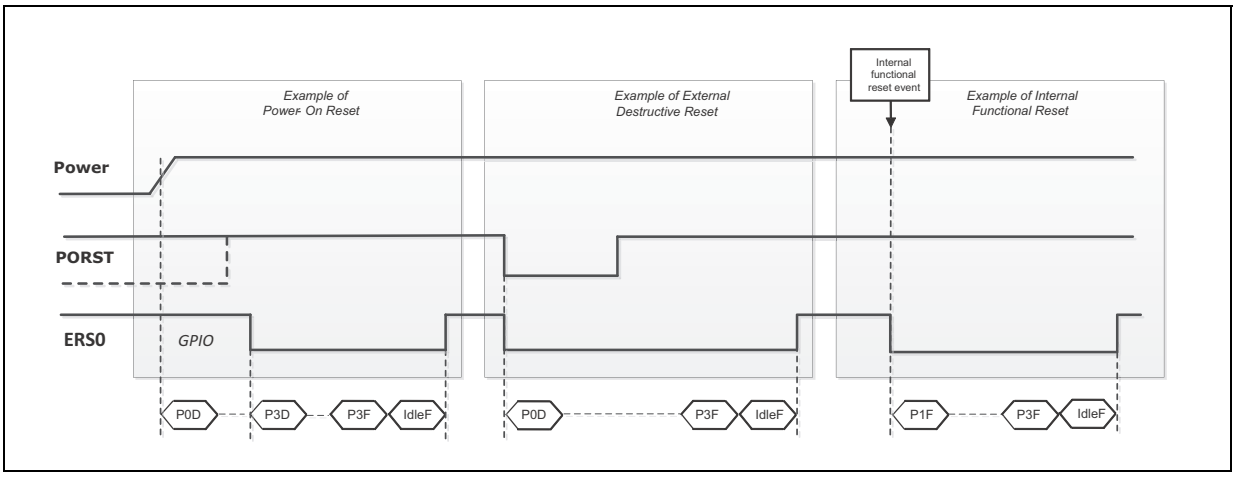
ESR0 pin characteristics: The ESR0 pin has the same electrical characteristic as a GPIO.

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

Figure 31 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 31. PORST reset pin state during the reset sequence



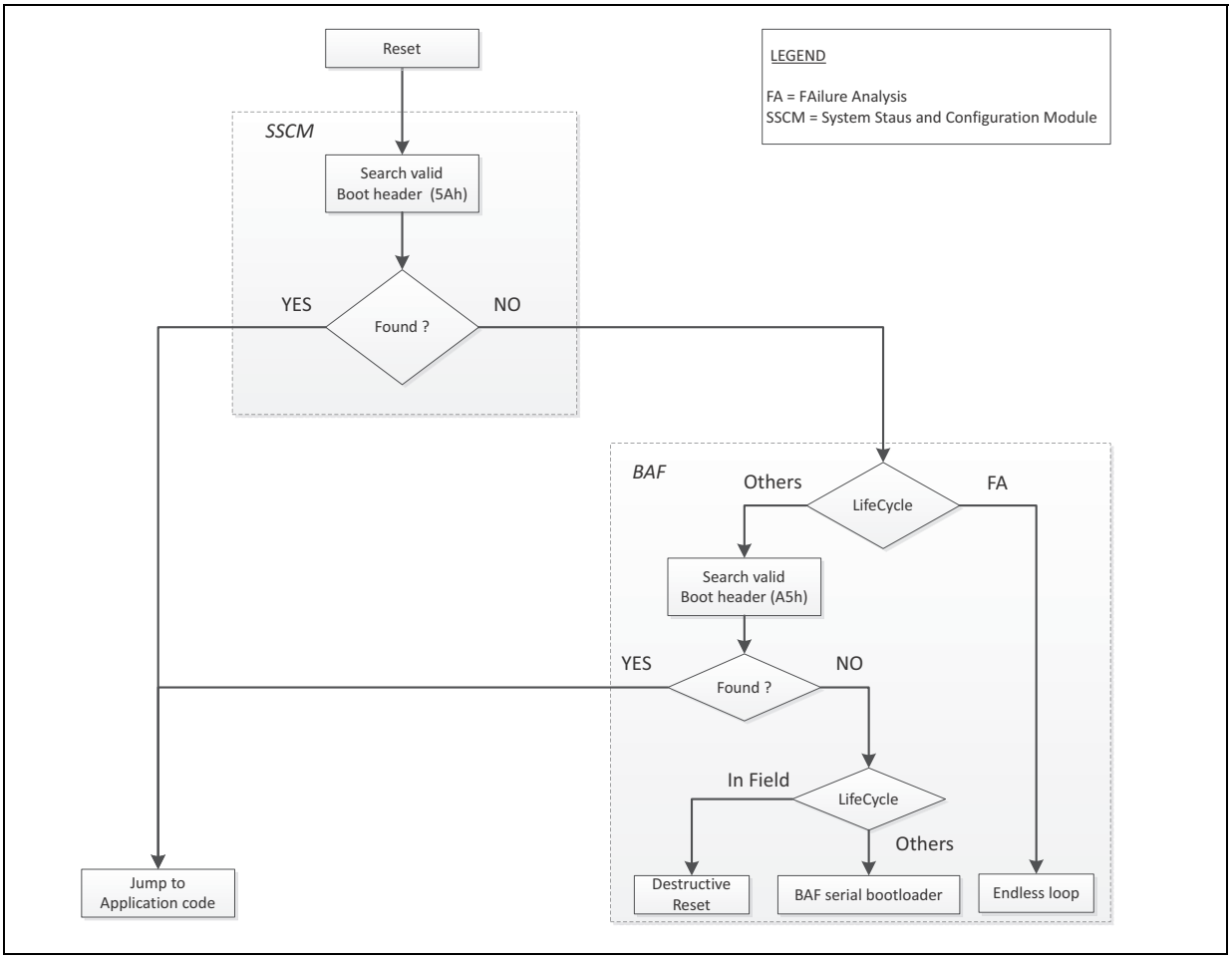
8.7 System boot

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 32 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 32. 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 91*. 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 91. 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 92*. 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 92. 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 128 KB Code Block 0	RCHW_LOC_4
6	0x0102_0000	Flash 128 KB Code Block 1	RCHW_LOC_5
7 (Last)	0x0040_4000	Flash 8 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.

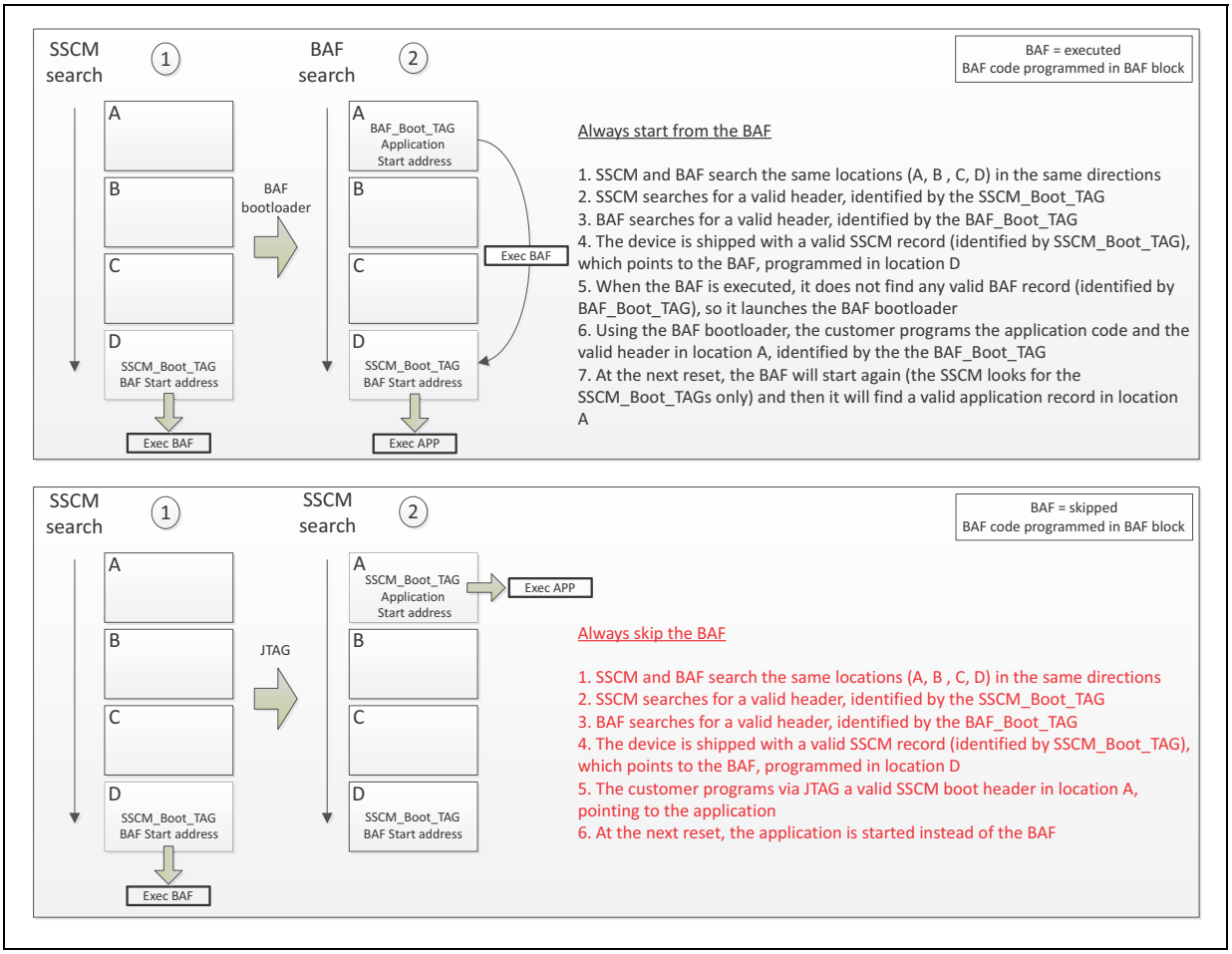
Depending upon the fact that a valid SSCM boot record is found and the content of the SSCM boot record, three scenarios can happens:

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 33 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 33. 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 93* shows the BAF boot record structure.*Table 94* shows the addresses of the location searched by the BAF

code. *Figure 32* shows a flow chart of the whole reset sequence, including the BAF search for a valid boot header. See *Chapter 44: Boot Assist Flash (BAF)* for additional information about the BAF code.

Table 93. BAF boot record structure

Offset	Field description	
0x0	BAF Boot Record Tag (A5h)	Reserved
0x4	Application start address	
0x8	Reserved	
0xC	Reserved	

Table 94. 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 128 KB Code Block 0
6 (Last)	0x0102_0000	Flash 128 KB Code Block 1

8.10 Performances

Table 95 shows boot timings.

Table 95. 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 98*. (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 34*.

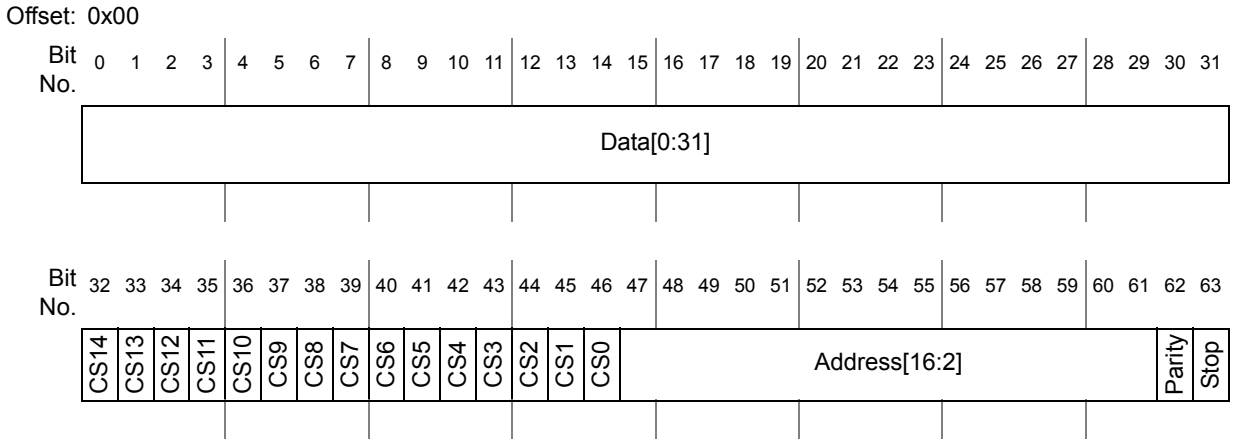


Figure 34. DCF record structure

Table 96. 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 35* 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 35. 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 36*. 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 36. 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 97*.

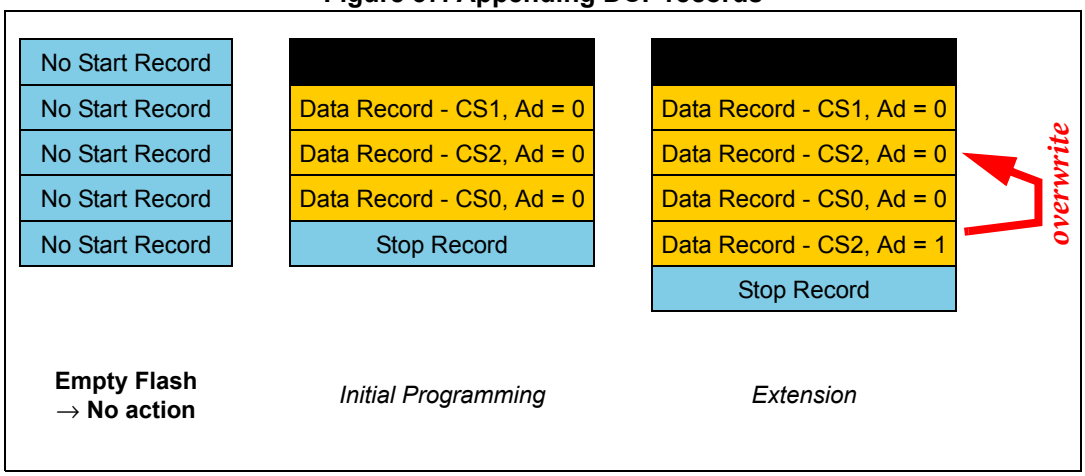
Table 97. Series of DCF records in UTEST flash memory

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 37*.

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 37*, 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 37. 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 98*. 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 98. DCF client list(1)

DCF CS [14:0]	DCF address [16:2] (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 3 LBIST partitions; 9 Memory Cuts; 32-bit PRPG; 64-bit MISR									
000_0000_0000_0100	000000000000010	STCU_SKC	—	—	yes	None	None	First record among STCU, Key1/Key2 DCF record sequence	16

Table 98. DCF client list[1](Continued)

DCF CS [14:0]	DCF address [16:2] (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	000000000000000	STCU_RUN	—	—	yes	None	None	None	8
000_0000_0000_0100	000000000000011	STCU_CFG	—	—	yes	None	None	None	8
000_0000_0000_0100	000000000000100	STCU_PLL_CFG	—	—	yes	None	None	None	8
000_0000_0000_0100	000000000000101	STCU_WDG	—	—	yes	None	None	None	8
000_0000_0000_0100	000000000000111	STCU_CRCE	—	—	yes	None	None	None	8
000_0000_0000_0100	000000000001010	STCU_ERR_FM	—	—	yes	None	None	None	8
000_0000_0000_0100	000000000010000	STCU_LBUFM	—	—	yes	None	None	None	8
000_0000_0000_0100	000000000011101	STCU_MBUFML [31:0]	—	—	yes	None	None	None	8
000_0000_0000_0100	000000001000000	STCU_LB_CTRL_0	—	—	yes	None	None	None	8
000_0000_0000_0100	000000001000001	STCU_LB_PCS_0	—	—	yes	None	None	None	8
000_0000_0000_0100	000000001000010	STCU_LB_PRPGL_0	—	—	yes	None	None	None	8
000_0000_0000_0100	000000001000100	STCU_LB_MISREL_0	—	—	yes	None	None	None	8
000_0000_0000_0100	000000001000101	STCU_LB_MISREH_0	—	—	yes	None	None	None	8
000_0000_0000_0100	000000001010000	STCU_LB_CTRL_1	—	—	yes	None	None	None	8
000_0000_0000_0100	000000001010001	STCU_LB_PCS_1	—	—	yes	None	None	None	8
000_0000_0000_0100	000000001010010	STCU_LB_PRPGL_1	—	—	yes	None	None	None	8
000_0000_0000_0100	000000001010100	STCU_LB_MISREL_1	—	—	yes	None	None	None	8
000_0000_0000_0100	000000001010101	STCU_LB_MISREH_1	—	—	yes	None	None	None	8
000_0000_0000_0100	000000001100000	STCU_LB_CTRL_2	—	—	yes	None	None	None	8

Table 98. DCF client list[1](Continued)

DCF CS [14:0]	DCF address [16:2] (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	000000001100001	STCU_LB_PCS_2	—	—	yes	None	None	None	8
000_0000_0000_0100	000000001100010	STCU_LB_PRPGL_2	—	—	yes	None	None	None	8
000_0000_0000_0100	000000001100100	STCU_LB_MISREL_2	—	—	yes	None	None	None	8
000_0000_0000_0100	000000001100101	STCU_LB_MISREH_2	—	—	yes	None	None	None	8
000_0000_0000_0100	000000110000000	STCU_MB_CTRL_0	—	—	yes	None	None	None	8
000_0000_0000_0100	000000110000001	STCU_MB_CTRL_1	—	—	yes	None	None	None	8
000_0000_0000_0100	000000110000010	STCU_MB_CTRL_2	—	—	yes	None	None	None	8
000_0000_0000_0100	000000110000011	STCU_MB_CTRL_3	—	—	yes	None	None	None	8
000_0000_0000_0100	000000110000100	STCU_MB_CTRL_4	—	—	yes	None	None	None	8
000_0000_0000_0100	000000110000101	STCU_MB_CTRL_5	—	—	yes	None	None	None	8
000_0000_0000_0100	000000110000110	STCU_MB_CTRL_6	—	—	yes	None	None	None	8
000_0000_0000_0100	000000110000111	STCU_MB_CTRL_7	—	—	yes	None	None	None	8
000_0000_0000_0100	000000110001000	STCU_MB_CTRL_8	—	—	yes	None	None	None	8
000_0000_0000_0100	000000110001001	Reserved							
PASS									
000_0000_0000_1000	000000000101000	Reserved							
000_0000_0000_1000	000000000101001 – 000000000101011	Reserved							
000_0000_0000_1000	000000000101100	Censorship	0x0000	0x55AA	no	None	None	None	8

Table 98. DCF client list[1](Continued)

DCF CS [14:0]	DCF address [16:2] (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	000000000101101 – 000000000101111	Reserved							
000_0000_0000_1000	000000000110000	Reserved							
000_0000_0000_1000	000000000110001 – 000000000111111	Reserved							
000_0000_0000_1000	000000001000000	LOCK0_PG0	0xFFF_FFFF	—	yes	None	None	None	8
000_0000_0000_1000	000000001000001	LOCK1_PG0	0xFFF_FFFF	—	yes	None	None	None	8
000_0000_0000_1000	000000001000010	LOCK2_PG0	0xFFF_FFFF	—	yes	None	None	None	8
000_0000_0000_1000	000000001000011	LOCK3_PG0	0xFFF_FFFF	—	yes	None	None	None	8
000_0000_0000_1000	000000001000100	LOCK0_PG1	0xFFF_FFFF	—	yes	None	None	None	8
000_0000_0000_1000	000000001000101	LOCK1_PG1	0xFFF_FFFF	—	yes	None	None	None	8
000_0000_0000_1000	000000001000110	LOCK2_PG1	0xFFF_FFFF	—	yes	None	None	None	8
000_0000_0000_1000	000000001000111	LOCK3_PG1	0xFFF_FFFF	—	yes	None	None	None	8
000_0000_0000_1000	000000001001000	LOCK0_PG2	0xFFF_FFFF	—	yes	None	None	None	8
000_0000_0000_1000	000000001001001	LOCK1_PG2	0xFFF_FFFF	—	yes	None	None	None	8
000_0000_0000_1000	000000001001010	LOCK2_PG2	0xFFF_FFFF	—	yes	None	None	None	8
000_0000_0000_1000	000000001001011	LOCK3_PG2	0xFFF_FFFF	—	yes	None	None	None	8
000_0000_0000_1000	000000001001100	LOCK0_PG3	0xFFF_FFFF	—	yes	None	None	None	8
000_0000_0000_1000	000000001001101	LOCK1_PG3	0xFFF_FFFF	—	yes	None	None	None	8
000_0000_0000_1000	000000001001110	LOCK2_PG3	0xFFF_FFFF	—	yes	None	None	None	8

Table 98. DCF client list[1](Continued)

DCF CS [14:0]	DCF address [16:2] (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	000000001001111	LOCK3_PG3	0xFFF_FFFF	—	yes	None	None	None	8
000_0000_0000_1000	000000001001100 – 000000011111111	Reserved							
Miscellaneous									
000_0000_1000_0000	000000000000000	UTEST Miscellaneous (See *Figure 38* and *Table 99* for bit position details)	0x0000 0140	—	no	Triple voted	None	None	8
000_0000_1000_0000	000000000000001 – 000000000011110	Reserved							
000_0000_1000_0000	000000000011111	PMC_REE bits	0x0000 0000[2]	—	no	Triple voted	Non-Differ-ential	None	8
000_0000_1000_0000	000000000100000 – 000000000111111	Reserved							
000_0000_1000_0000	000000001111110	ESR0_CFG	0x0000 0000	—	yes	Triple voted	None	None	8
000_0000_1000_0000	000000001111111 – 000000011111111	Reserved							
BAF 'Soft' Clients									
100_0000_0000_0000	0x0000	Reserved							
100_0000_0000_0000	0x0001	Reserved							
100_0000_0000_0000	0x0002	Reserved							
100_0000_0000_0000	0x0003	Reserved							
100_0000_0000_0000	0x0004 – 0x01FF	Reserved							

Table 98. DCF client list[1](Continued)

DCF CS [14:0]	DCF address [16:2] (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					
100_0000_0000_0000	0x0200	Address	—	—	no	None	None	Must be written first as a pair with a data register	8
100_0000_0000_0000	0x0201	32-bit Data	—	—	no	None	None	Must be written second as a pair with the Address register	8
100_0000_0000_0000	0x0202	16-bit Data	—	—	no	None	None	Must be written second as a pair with the Address register	8
100_0000_0000_0000	0x0203	8-bit Data	—	—	no	None	None	Must be written second as a pair with the Address register	8
100_0000_0000_0000	0x0204 – 0x020F	Reserved							
100_0000_0000_0000	0x0210	Callback address	—	—	no	None	None	None	8
100_0000_0000_0000	0x0211	Serial boot callback address	—	—	no	None	None	None	8
100_0000_0000_0000	0x0212 – 0x7FFF	Reserved							
OTP Glue									
000_0000_0001_0000	000000000000000 – 000000000000111	Reserved							
000_0000_0001_0000	000000000001000	OTP_EN0	0x00000000	—	no	Write once	None	None	8
000_0000_0001_0000	000000000001001	OTP_EN1	0x00000000	—	no	Write once	None	None	8
000_0000_0001_0000	000000000001010	OTP_EN2	0x00000000	—	no	Write once	None	None	8

Table 98. DCF client list(1)(Continued)

DCF CS [14:0]	DCF address [16:2] (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_0001_0000	000000000001011	OTP_EN3	0x0000 0000	—	no	Write once	None	None	8
000_0000_0001_0000	000000000001100 – 000000011111111	Reserved							

1. DCF programming should start from address 0x00400208.
2. Reset value will depend on the TEST DCF programmed value.

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 98: DCF client list*.

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0			0			0									0
W		XOSC ALC_DIS	XOSC LF_EN		3p3V_5V	OSC EN_40MHZ		LOCKSTEP EN	XOSC EN	XOSC EXT CLOAD	XOSC LOAD CAP SEL					

Reset The reset value is described in *Table 98: DCF client list*.

Figure 38. UTEST Miscellaneous DCF Client

Table 99. 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)
20 3p3V_5V	Main segment 3.3V vs 5V indicator 0 selected power supply is 3.3V 1 selected power supply is 5V

Table 99. UTEST Miscellaneous DCF Client field descriptions(Continued)

Field	Description
21 OSC EN_40MHZ	XOSC EN_40MHz **Note:** This field must be configured according to the external cristal used. 0 20 MHz 1 40 MHz XOSC
23 LOCKSTEP EN	Enable checker core to be in lockstep with main core 0 Lockstep disabled 1 Lockstep enabled
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 Device 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	0	0	0	0	0	0							
W										LVD14_AD	HVD13_C	LVD11_AD	LVD11_FL	LVD11_C	HVD6_C	LVD3_C
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 39. PMC_REE bits DCF Client

Note: LVD11_C, LVD11_FL & LVD3_C monitors once enabled in DCF cannot be cleared by DCF record.

Table 100. PMC_REE bits DCF client field descriptions

Field	Description
25 LVD14_AD	LVD14_AD reset enable This bit defines whether an LVD assertion on the first supply of the ADC generates a system reset. The RES_HV1[RES14_AD] bit determines whether the reset is functional or destructive. 0 Disabled. LVD assertion on the first supply of the ADC does not cause system reset 1 Enabled. LVD assertion on the first supply of the ADC causes system reset
26 HVD13_C	HVD13_C reset enable This bit defines whether an HVD assertion on the core 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 core supply does not cause system reset 1 Enabled. HVD assertion on the core supply causes system reset
27 LVD11_AD	LVD11_AD reset enable This bit defines whether an LVD assertion on the first supply of the ADC generates a system reset. The RES_HV0[RES11_AD] bit determines whether the reset is functional or destructive. 0 Disabled. LVD assertion on the first supply of the ADC does not cause system reset 1 Enabled. LVD assertion on the first supply of the ADC causes system reset
28 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
29 LVD11_C	LVD11_C reset enable This bit defines whether an LVD assertion on the core 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 core supply does not cause system reset 1 Enabled. LVD assertion on the core supply causes system reset
30 HVD6_C	HVD6_C 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 LVD3_C	LVD3_C 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

SPC570Sx microcontrollers include a robust power management infrastructure that enables applications to select among various operational and low-power modes and to monitor internal voltages for high- and low-voltage conditions. The monitoring capability is also used to ensure supply voltages and internal voltages are within the required ranges before the microcontroller can leave reset. This chapter gives an overview of the built-in power management features.

The power management infrastructure comprises the following modules:

- Power Control Unit (MC_PCU)
- Clock Generation Module (MC_CGM)
- Mode Entry Module (MC_ME)

This chapter describes these modules in brief. For complete details, see the respective module chapters.

The MC_PCU includes a status register which provides power domain status information. An MC_PCU submodule, the Power Management Controller Digital Interface (or “PMC_DIG”), provides an interface to configure and control internal voltages.

The MC_CGM generates reference clocks for all the chip blocks. It works in conjunction with internal clock gating logic to implement low power modes.

The MC_ME module controls chip operational modes and mode transition sequences. It also contains configuration, control and status registers accessible for the application.

The SPC570Sx has two types of modes: SYSTEM modes and USER modes.

SYSTEM modes:

- SYSTEM modes are entered automatically on system events or on application request
- The SYSTEM modes are: RESET, SAFE, TEST and DRUN

Table 101. SYSTEM modes

Mode	Description	Entry	Exit
RESET	System remains in this mode until all resources are available for software to take device control. RESET manages hardware initialization of chip configuration, voltage regulators, oscillators, PLLs, and flash.	System reset assertion from RGM	System reset deassertion from RGM
SAFE	May be entered on detection of a recoverable error. Forces system into predefined safe configuration from which the system may try to recover.	Hardware failure, software request from DRUN, TEST and RUNx	System reset assertion, DRUN through software
TEST	Control environment for device self-test. Allows application to run its own self-test like flash checksum and RAM BIST.	Software request from DRUN	System reset asserted, DRUN through software

Table 101. SYSTEM modes(Continued)

Mode	Description	Entry	Exit
DRUN	Entry mode for the embedded software. Provides full system accessibility and enables system configuration. Provides gate to enter USER modes. BAF, when present, is executed in DRUN mode.	Reset removed, from SAFE, TEST and RUNx through software	Reset applied, RUNx, TEST through software, SAFE through software or hardware failure

USER modes:

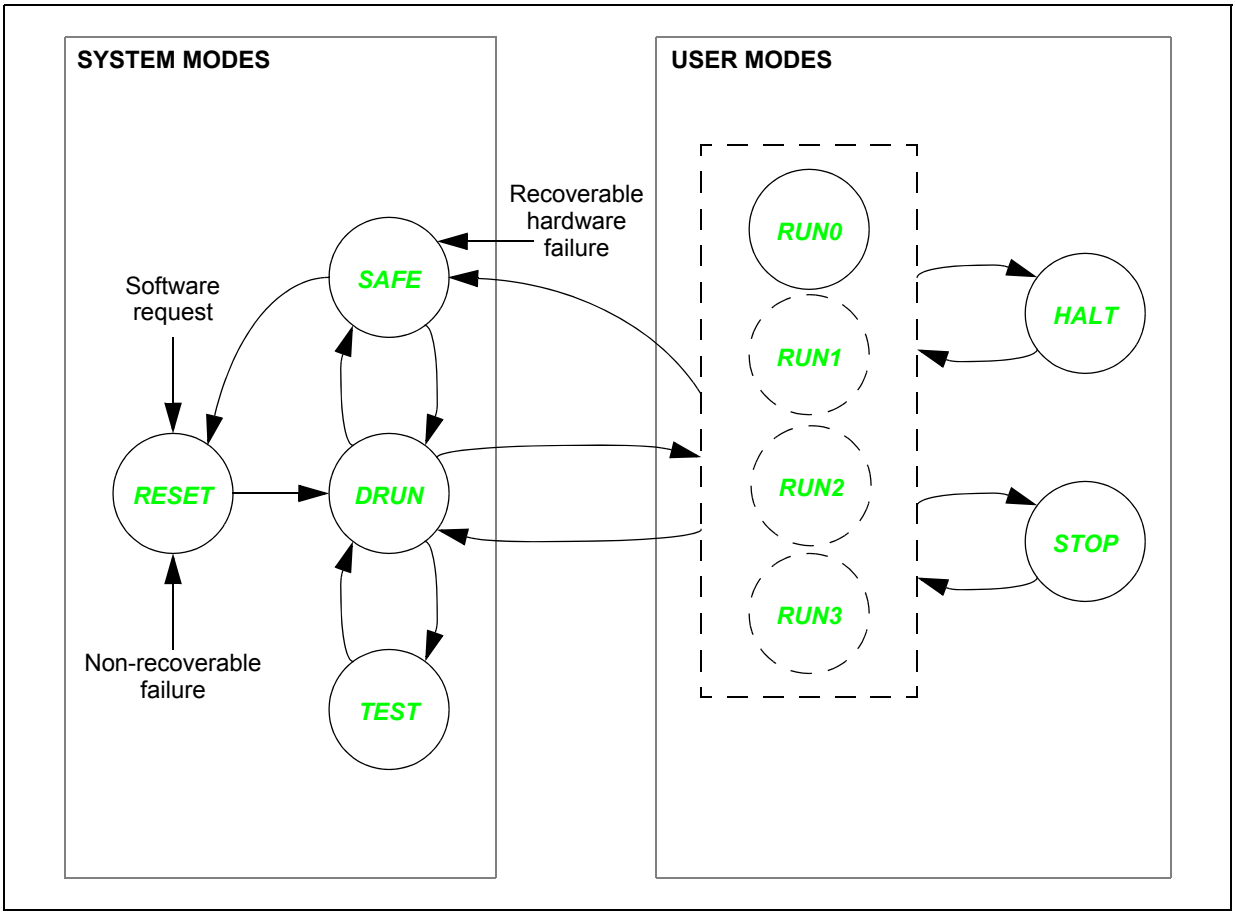
- USER modes control performance and consumption during normal application operation. The modes are organized according to the performance provided.
- The USER modes are RUNx, HALT, and STOP.
- Each mode is entered by writing a dedicated register protected by a key, to avoid unwanted transition.

Figure 40 shows the mode relationships.

Table 102. USER modes

Mode	Description	Entry	Exit
RUNx	These are software running modes where most of the processing activity is done. These run modes allow the user to enable different clock and power configurations of the system with respect to each other.	Software request from DRUN, interrupt event from HALT, wakeup request from STOP	Reset asserted, SAFE through software or hardware failure, RUNx, HALT, STOP
HALT	Reduced-activity low-power mode during which the core clock is disabled. It can be configured to switch off analog peripherals like PLL, flash, main voltage regulator, etc., for efficient power management at the cost of higher wakeup latency.	Software request from RUNx	Reset asserted, SAFE on hardware failure, RUNx on interrupt event
STOP	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 the oscillator for efficient power management at the cost of higher wakeup latency.	Software request from RUNx	Reset asserted, SAFE on hardware failure, RUNx on wakeup event

Figure 40. Mode relationships



10.2 Architecture

10.2.1 Power management modules

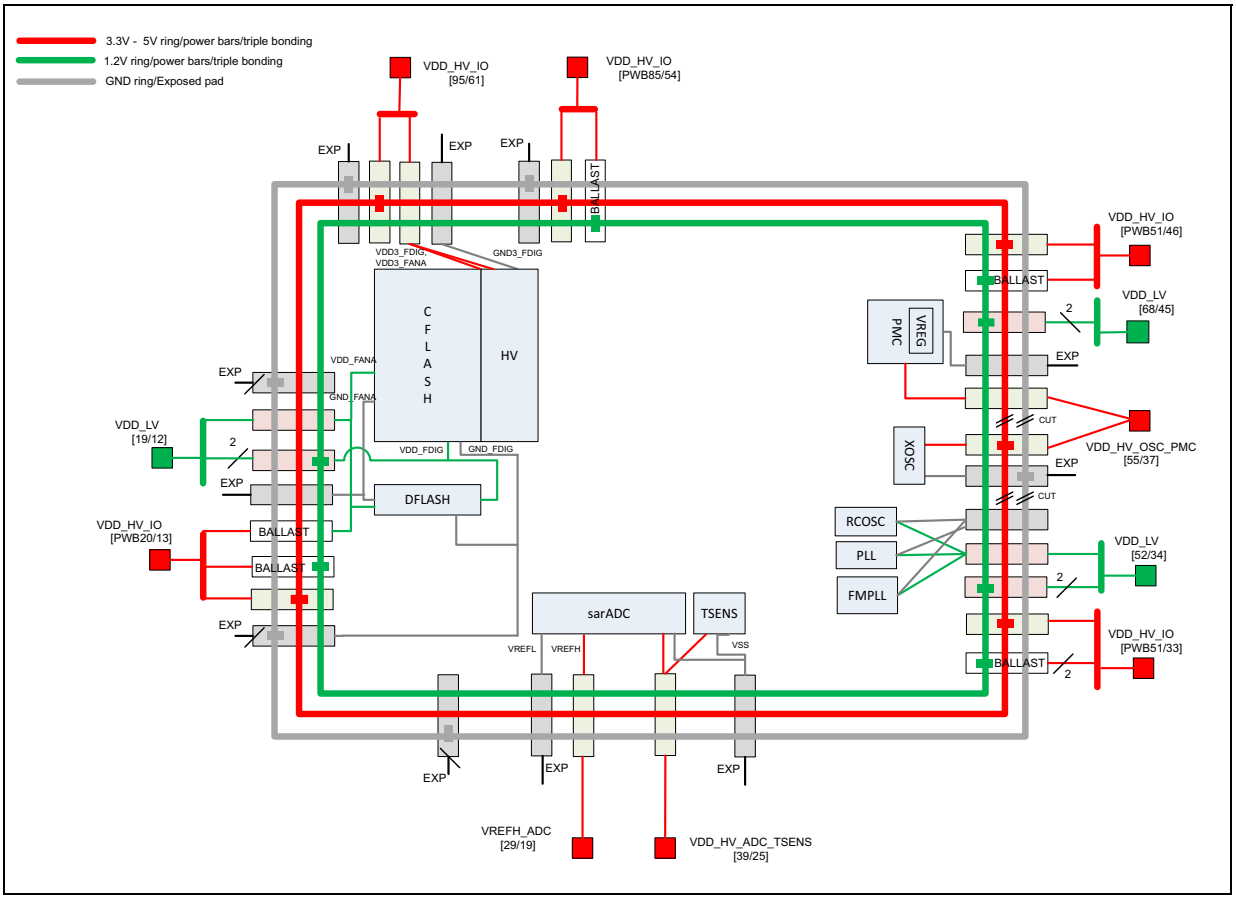
The SPC570Sx provides two dedicated supplies at the package pin level (refer to *Chapter 4: Signal description* for the pinout description):

- VDD_HV: high voltage external power supply for I/Os, JTAG, and most analog modules
- VDD_HV_ADC: high voltage external power supply for ADC module

The internal LV supply, which is used to power all the digital macrocells, is generated by an internal voltage regulator, called HPREG.

Figure 41 shows the general power supply scheme.

Figure 41. Power supply scheme



10.2.2 Power management supply description

Table 103 provides an overview of the PMC supply signals.

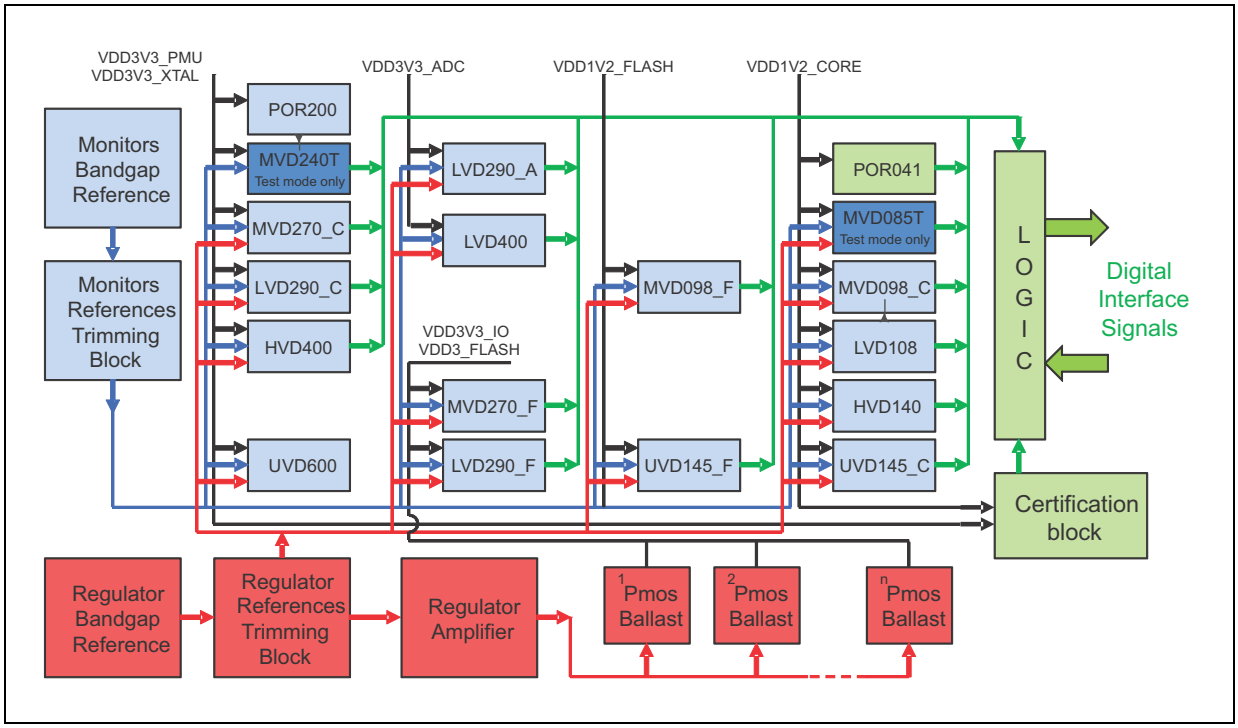
Table 103. Device power management controller external signals

Name	Type	Description
VSS	Ground	Ground supply for the device / I/O. This is covering both VSS_LV and VSS_HV in case of exposed pad device.
VSS_HV_ADC_TSENS	Ground	SARADC ground supply connected to exposed pad
VREFH_ADC	Reference	SARADC voltage reference
VREFL_ADC	Reference	SARADC ground reference
VDD_HV_ADC_TSENS	Supply	SARADC high voltage power supply
VDD_HV_IO	Supply	High voltage power supply for the I/Os
VDD_HV_OSC_PMC	Supply	High voltage power supply for internal power management unit and oscillator
VDD_LV	Supply	Low voltage power supply for the core area generated by internal regulator

Warning: **There are constraints on some input voltages relative to other input voltages. Please refer to the device data sheet for detailed information.**

Figure 42 shows the different power segments generated from the HV supply and the voltage monitor architecture.

Figure 42. Power segment and voltage monitor architecture



10.3 Functionality

10.3.1 Regulators capabilities

The PMU provides the following regulation capability:

- Fully internal regulator capability
- Up to 150mA current capability of internal regulator
- Separate reference and trimming block to avoid common cause of failure with the monitors unit

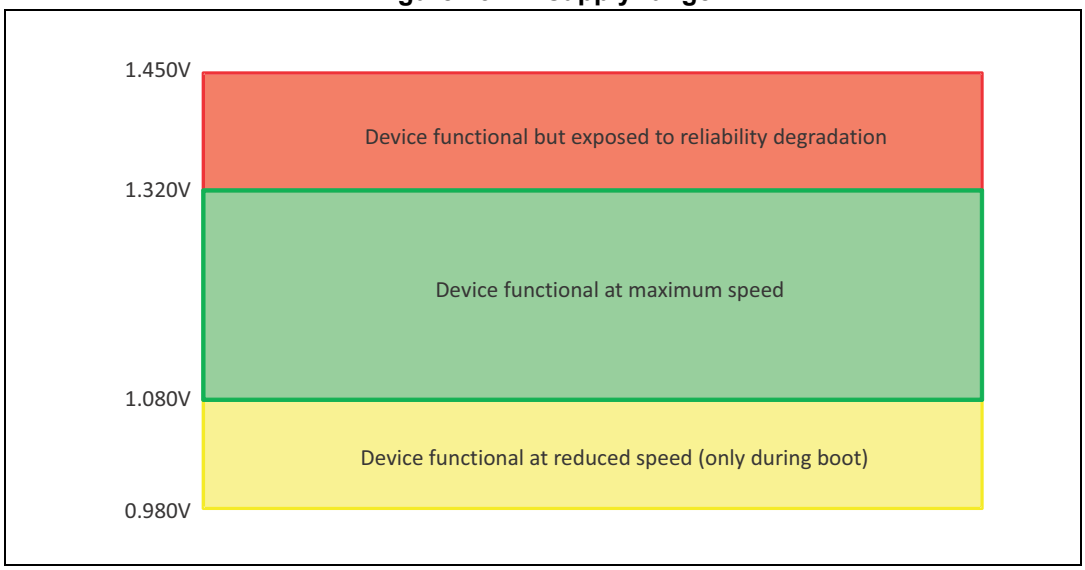
The BIST shall verify the capability of all the monitors to trigger at the correct voltage with a tolerance of roughly 5 %.

10.3.2 1.2 V supply

The LV supply is generated from VDD_HV input supply by an internal voltage regulator.

Figure 43 shows the LV supply operating range.

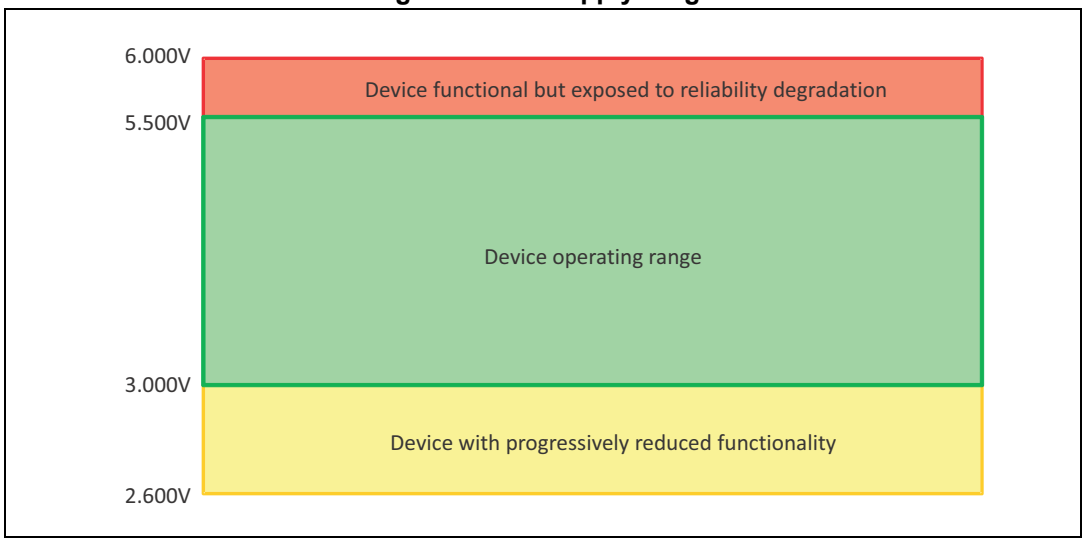
Figure 43. LV supply range



10.3.3 High voltage supply

The device can work for both 3 V and 5 V ranges. *Figure 44* shows the HV supply operating range.

Figure 44. HV supply range



10.4 Voltage monitors

10.4.1 Nomenclature

- **POR** stands for Power On Reset. The POR circuit 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 generate 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 generate a destructive reset.

10.4.2 Voltage monitor summary

1.2 V domain monitors:

- POR voltage
  - POR041_C
- Low voltage
  - MVD098_C/F, LVD108_C, MVD085T_C(e)
- High voltage
  - HVD140_C, UVD145_C/F

3.3 V domain monitors:

- POR voltage
  - POR200_C
- Low voltage
  - MVD270_C/F, LVD290_A/C/F, MVD240T_C(e)
- High voltage
  - HVD400_C

5 V domain monitors:

- Low voltage
  - LVD400_A
- High voltage
  - UVD600_C

Table 104, *Table 105* and *Table 106* summarize the voltage monitor ranges.

Table 104. 1.2 V voltage monitors

Name	Segment	Target value	Lower limit	Upper limit
POR041_C	Core	0.7 V	0.39 V	0.95 V
MVD085T_C	Core	0.85 V	0.817 V	0.883 V

e. Used in test mode only

Table 104. 1.2 V voltage monitors(Continued)

Name	Segment	Target value	Lower limit	Upper limit
MVD098_C	Core	1.030 V	1.005 V	1.055 V
MVD098_F	Flash	1.030 V	1.005 V	1.055 V
LVD108_C	Core	1.110 V	1.085 V	1.137 V
HVD140_C	Core	1.369 V	1.340 V	1.400 V
UVD145_C	Core	1.410 V	1.379 V	1.441 V
UVD145_F	Flash	1.410 V	1.379 V	1.441 V

Table 105. 3.3 V voltage monitors

Name	Segment	Target value	Lower limit	Upper limit
POR200_C	Core	2.180 V	1.960 V	2.400 V
MVD240T_C	Core	2.400 V	2.340 V	2.460 V
MVD270_C	Core	2.752 V	2.694 V	2.826 V
MVD270_F	Flash	2.752 V	2.694 V	2.826 V
LVD290_C	Core	2.940 V	2.881 V	2.999 V
LVD290_F	Flash	2.940 V	2.881 V	2.999 V
LVD290_A	ADC	2.940 V	2.881 V	2.999 V
HVD400_C	Core	3.750 V	3.660 V	3.840 V

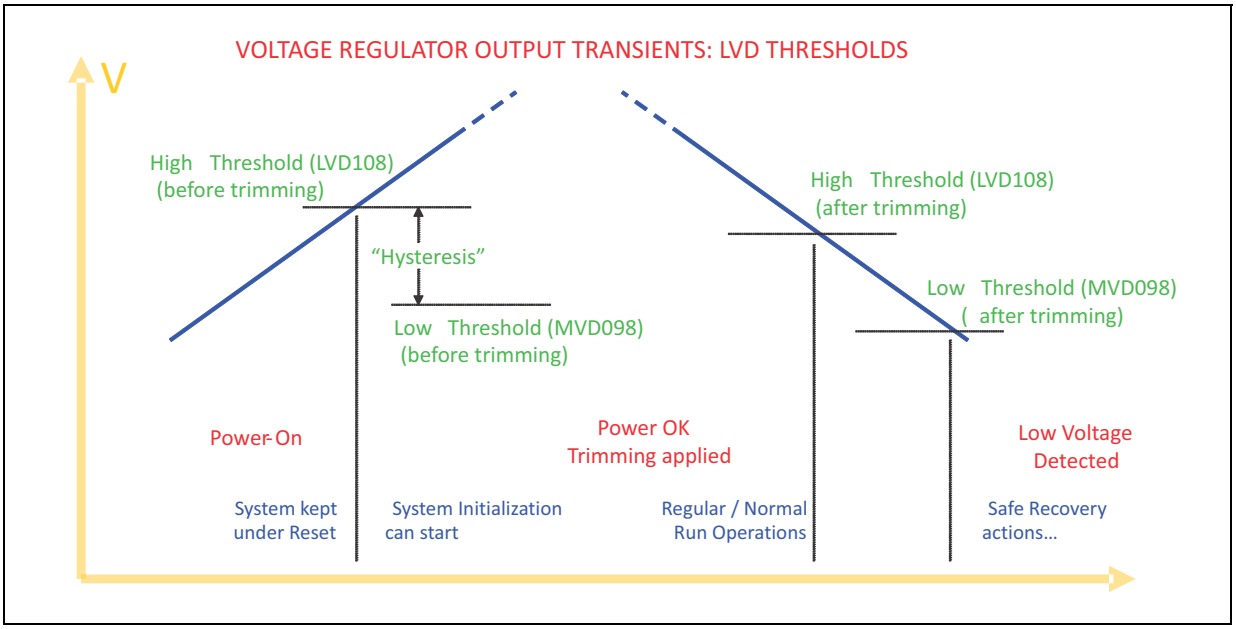
Table 106. 5 V voltage monitors

Name	Segment	Target value	Lower limit	Upper limit
LVD400_A	ADC	4.230 V	4.128 V	4.332 V
UVD600_C	Core	5.820 V	5.684 V	5.920 V

10.4.3 LV monitors

Figure 45 shows what LV monitors are applied from the power-on till the application is running in order to protect from 1.2V voltage undershoot events.

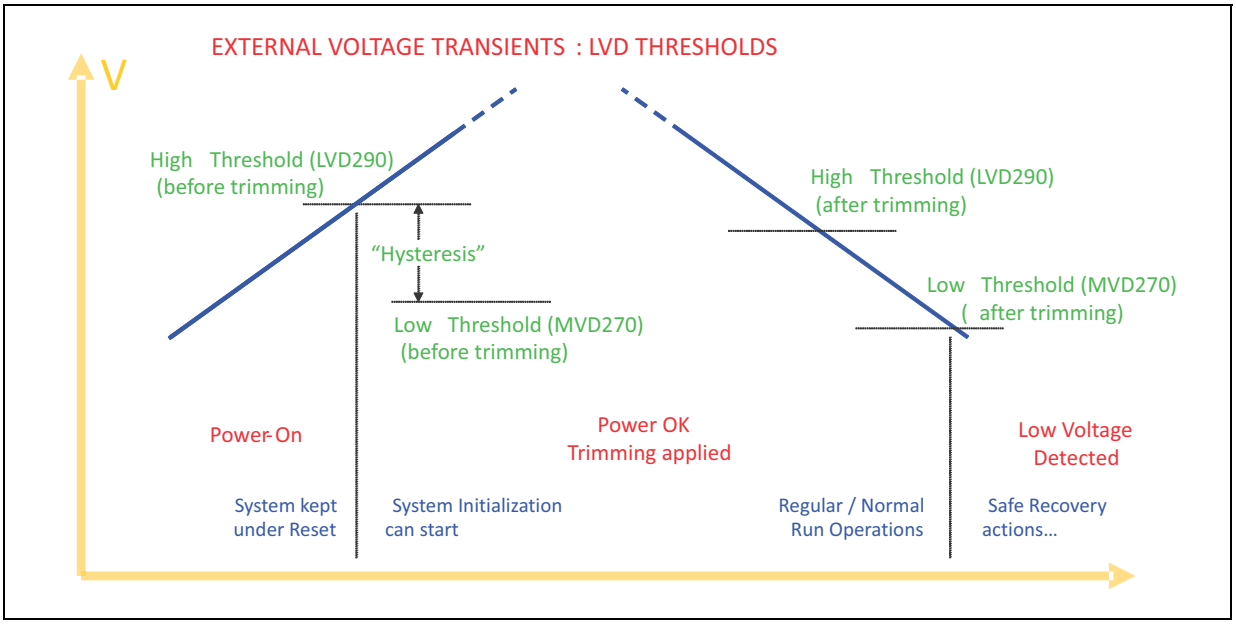
Figure 45. LV supply monitors vs. voltage undershoot events



10.4.4 HV monitors

Figure 46 shows what HV monitors are applied from the power-on till the application is running in order to protect from 3.3V/5V voltage undershoot events.

Figure 46. HV supply monitors vs. voltage undershoot events



10.4.5 Voltage monitor digital interface

The voltage monitors are controlled by the MC_PMC block. *Table 107* summarizes the various features associated to each voltage monitor.

The table uses the following nomenclature:

- N = Voltage monitor cannot be disabled
- TM = Voltage monitor can be disabled in Test Mode
- TCU = Voltage monitor can be disabled by the TCU
- DCF = Voltage monitor can be disabled via DCF[f]
- U = Voltage monitor can be disabled via Test Mode (TCU) only
- D = Always triggers a destructive reset
- POR = Always triggers a power on reset (phase 0)
- X = Can never generate the corresponding trigger
- P[DR/FR] = Programmable to generate a destructive or functional reset trigger
- P = Programmable to generate the corresponding trigger

Table 107. Voltage monitor features

Voltage monitor	Disable	Trigger generation		
		Reset	Interrupt	FCCU
POR041_C	N	POR	X	X
MVD085T_C	TCU	POR	X	X
MVD098_C	TM	POR	X	X
MVD098_F	TM	POR	X	X
LVD108_C	TM/DCF	P[DR/FR]	P	P
HVD140_C	TM/DCF	P[POR/FR](1)	P	P
UVD145_C	TM	POR	X	X
UVD145_F	TM	POR	X	X
POR200_C	N	POR	X	X
MVD240T_C	TM	POR	X	X
MVD270_C	TM	POR	X	X
MVD270_F	TM	POR	X	X
LVD290_C	TM/DCF	P[DR/FR]	P	P
LVD290_F	TM/DCF	P[DR/FR](2)	P	P
LVD290_A	TM/DCF	P[DR/FR]	P	P
HVD400_C	TM/DCF	P[DR/FR]	P	P
LVD400_A	TM/DCF	P[DR/FR]	P	P
UVD600_C	TM	POR	X	X

1. If HVD140_C is programmed as FR and if this reset occurs multiple times then it will generate POR (after crossing the functional reset escalation threshold).
2. If the user needs a functional reset for LVD290_F, do not program RES_HV0 register, but instead program a functional reset reaction by programming an Interrupt/FCCU event in IE_HV0/FEE_HV0 respectively.

f. If the voltage monitor is enabled via DCF, it can be disabled in Test Mode only

Table 108. POR and voltage monitors description

Voltage Detector		Power ON exit	PH0D exit	PH1D-PH2D	Trimming	Post PH3D	Destr RST	Funct RST	IRQ	FCCU fault	VD disabling	Trimming	SW BIST
No.	Name	PH1F-PH2F-PH33F-exit											
VD15	UVD600_C	Active (HW)						—			Test mode	Y	Y
VD14	LVD400_AD	NA				Active (SW)	user[1]	user[1]	user[2]	user[2]	Test mode or SW	Y	Y
							disabled	disabled	disabled	disabled			
VD13	HVD400_C					Active (SW)	user[1]	user[1]	user[2]	user[2]	Test mode or SW	Y	Y
							disabled	disabled	disabled	disabled			
VD11	LVD290_AD					Active (SW)	user[1]	user[1]	user[2]	user[2]	Test mode or SW	Y	Y
							disabled	disabled	disabled	disabled			
VD11	LVD290_F	—	Active (HW)		NA	Active (SW)	active[3]	user[1]	user[2]	user[2]	Test mode or SW	Y	Y
							disabled	disabled	disabled	disabled			
VD11	LVD290_C		Active (HW)		NA	Active (SW)	active[3]	user[1]	user[2]	user[2]	Test mode or SW	Y	Y
							disabled	disabled	disabled	disabled			
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	NA
VD8	POR200_C	Active (HW)									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	user	user	user	Test mode or SW	Y	Y
							masked	masked	masked	masked			
VD3	LVD108_C		Active (HW)	NA		Active (SW)	active[3]	user	masked	masked	Test mode or SW	Y	Y
							masked	masked	masked	masked			
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	NA
VD0	POR041_C	Active (HW)									NA		
Key to colors		SW setting via DCF/PBRIDGE regs init						SW setting via PBRIDGE registers init					
		Default setting w/o DCF init (user/test FLASH)						Default setting (PBRIDGE regs)					
Key to abbreviations:		HVD = High VD LVD = low VD		MVD = minVD NA = not available				SW = software (option bit) UVD = ultra VD			VD = voltage detector Y = available		

1. Destructive reset reaction can be set by UTEST or PBRIDGE. Functional reset reaction can be set by PBRIDGE. PBRIDGE can only modify the type of reaction (RES register -> destructive or functional) without to disable the REE setting by UTEST.
2. IRQ/FCCU reaction type is USER defined (PBRIDGE) according to the related application. It is typically enabled if a destructive or functional reset reaction is disabled.
3. mandatory setting to be programmed in the TEST FLASH that cannot be overwritten (VD disabling) by the UTEST FLASH.

10.4.6 Safety requirements

LVDs and HVDs shall be testable after power-up.

If voltage regulation is implemented internally, it shall use a different bandgap than its supervision.

10.4.7 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 25 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 110*).

The timing diagram in *Figure 47* explains the sequence for checking three different monitors: LVD_290C,LVD_290_F_LVD_290A. All inputs and outputs are controlled inside the PMC_Dig module.

Figure 47. Monitor checking timing diagram

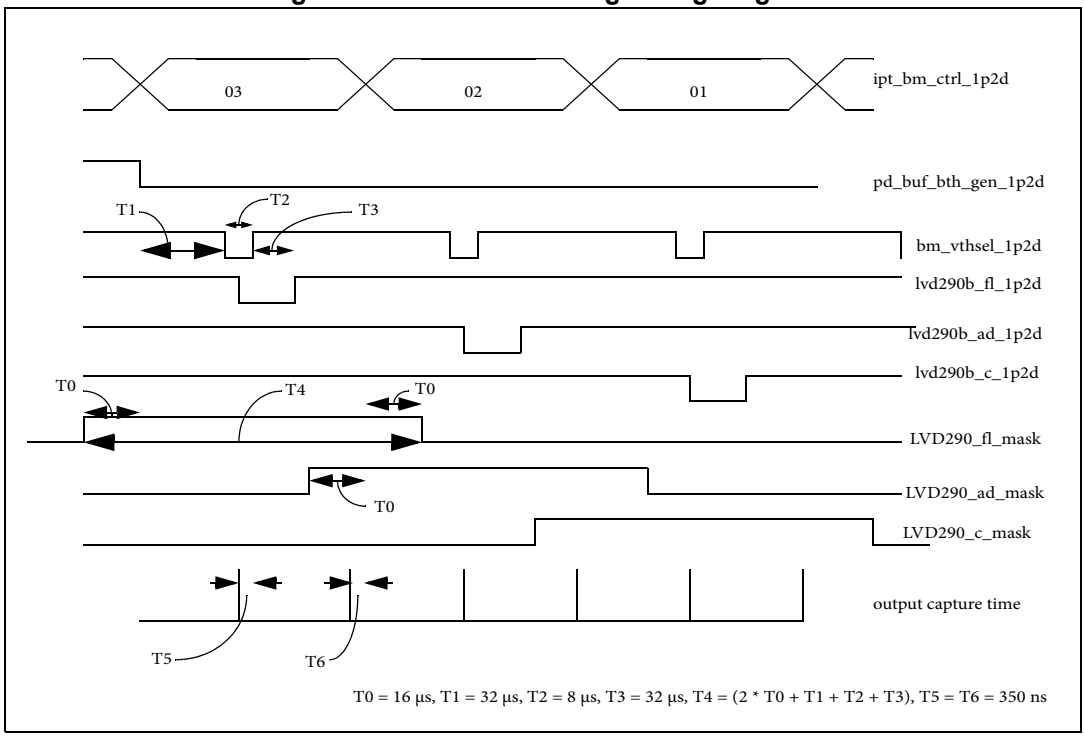


Table 109. Timing descriptions

Label	Description	Value
T0	the time that the monitor output is masked from the SoC before and after the monitor selection.	16 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.	32 µs
T2	the time for which "bm_vthsel_1p2d" is kept logic 0.	8 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	104 µs
T5	the time before "bm_vthsel_1p2d" becomes logic 1 such that the comparator logic 0 output can be checked	350 ns
T6	the time before the end of T3 phase to check logic 1 output of the comparator	350 ns

The BIST_TIME registers need to be programmed according to the following formula:

Equation 1 TIME_*x* = *Tx* * pbridge_clk_freq - 1 where *x* is 0 to 3, 5 and 6.
See *Table 109* for the values of T*x*.

Table 110. 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)
25 MHz	dec	399	799	199	799	8	8	0x031F018F	0x031F00C7	0x00080008
	hex	0x018F	0x031F	0x00C7	0x031F	0x0008	0x0008			
40 MHz	dec	639	1279	319	1279	13	13	0x04FF027F	0x04FF013F	0x000D000D
	hex	0x027F	0x04FF	0x013F	0x04FF	0x000D	0x000D			

1. See *Section 46.5.33: User BIST Time1 and Time0 Register (BIST_TIME10)*
2. See *Section 46.5.34: User BIST Time3 and Time2 Register (BIST_TIME32)*
3. See *Section 46.5.35: 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 46.5.16: Event Pending Register (EPR_HV1)).

Table 111 shows the individual monitors to check.

Table 111. 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 46.5.31: 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 SPC570Sx 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 SPC570Sx 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 SPC570Sx security features are published in a separate SPC570Sx *Microcontroller Security Reference Manual*, which is only available to qualified customers.

12 e200z0h Core description

12.1 Overview

The e200z0h processor family is a set of CPU cores that implement cost-efficient versions of the Power Architecture®. e200z0h processors are designed for deeply embedded control applications which require low cost solutions rather than maximum performance.

The e200z0h processors integrate an integer execution unit, branch control unit, instruction fetch and load/store units, and a multi-ported register file capable of sustaining three read and two write operations per clock. Most integer instructions execute in a single clock cycle. Branch target prefetching is performed by the branch unit to allow single-cycle branches in some cases.

The e200z0h core is a single-issue, 32-bit Power Architecture technology VLE-only (Variable Length Encoding) design with 32-bit general purpose registers (GPRs). All arithmetic instructions that execute in the core operate on data in the GPRs.

Instead of the base Power Architecture technology support, the e200z0h core implements only the VLE (variable-length encoding) APU, providing improved code density.

12.2 Microarchitecture summary

The e200z0h processor utilizes a 4-stage pipeline for instruction execution:

- Instruction Fetch (stage 1)
- Instruction Decode/Register file Read/Effective Address Calculation (stage 2)
- Execute/Memory Access (stage 3)
- Register Writeback (stage 4)

The four stages operate in an overlapped fashion, allowing single clock instruction execution for most instructions.

The integer execution unit consists of a 32-bit Arithmetic Unit (AU), a Logic Unit (LU), a 32-bit Barrel shifter (Shifter), a Mask-Insertion Unit (MIU), a Condition Register manipulation Unit (CRU), a Count-Leading-Zeros unit (CLZ), an 8×32 Hardware Multiplier array, result feed-forward hardware, and a hardware divider.

Arithmetic and logical operations are executed in a single cycle with the exception of the divide and multiply instructions. A Count-Leading-Zeros unit operates in a single clock cycle.

The Instruction Unit contains a PC incrementer and a dedicated Branch Address adder to minimize delays during change of flow operations. Sequential prefetching is performed to ensure a supply of instructions into the execution pipeline. Branch target prefetching from the BTB is performed to accelerate certain taken branches in the e200z0h. Prefetched instructions are placed into an instruction buffer with four entries in the e200z0h, each capable of holding a single 32-bit instruction or a pair of 16-bit instructions.

Conditional branches which are not taken execute in a single clock. Branches with successful target prefetching have an effective execution time of one clock on the e200z0h. All other taken branches have an execution time of two clocks.

Memory load and store operations are provided for byte, halfword, and word (32-bit) data with automatic zero or sign extension of byte and halfword load data as well as optional byte reversal of data. These instructions can be pipelined to allow effective single cycle

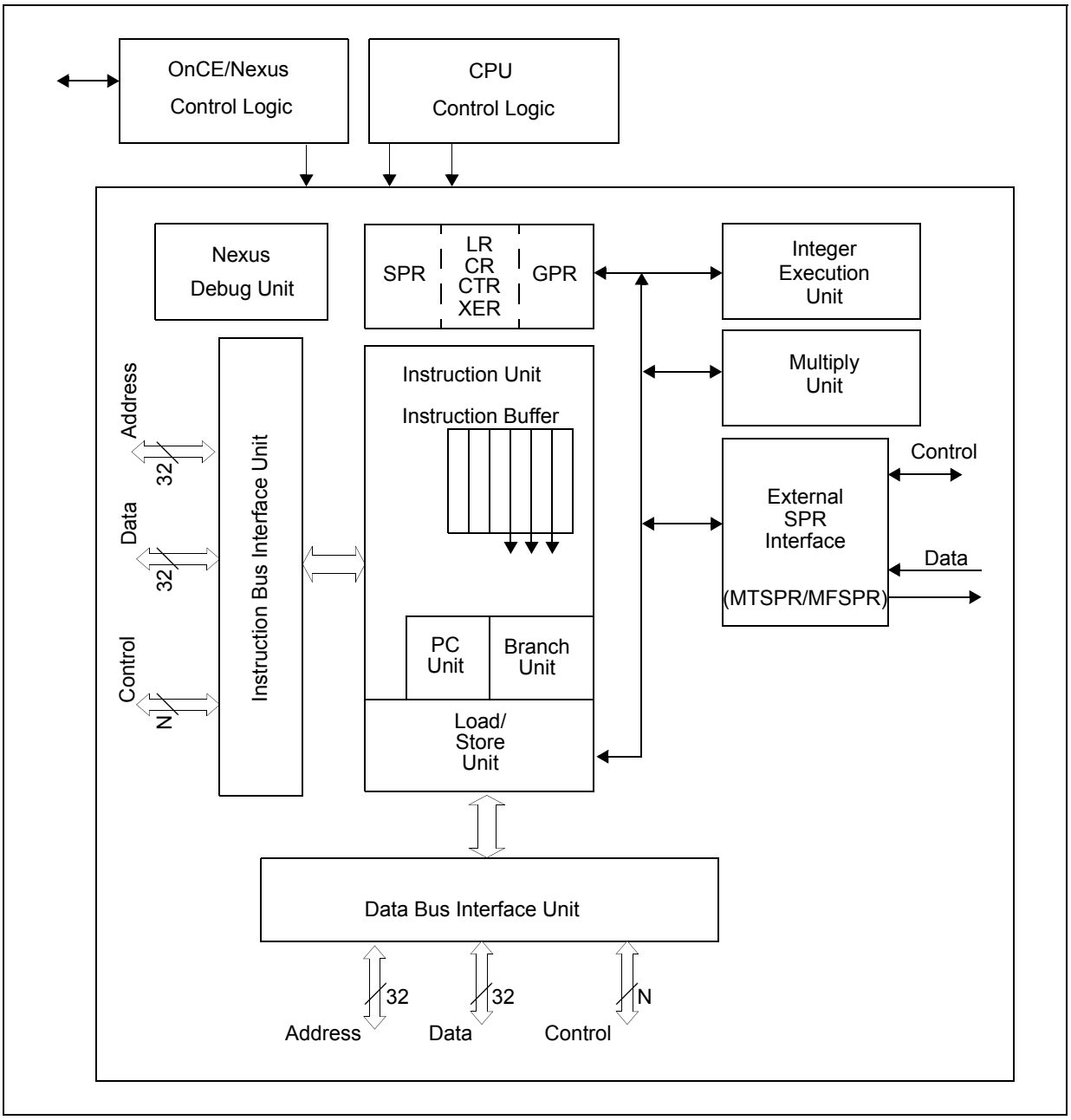
throughput. Load and store multiple word instructions allow low overhead context save and restore operations. The load/store unit contains a dedicated effective address adder to allow effective address generation to be optimized. Also, a load-to-use dependency does not incur any pipeline bubbles for most cases.

The Condition Register unit supports the condition register (CR) and condition register operations defined by the Power Architecture platform. The condition register consists of eight 4-bit fields that reflect the results of certain operations, such as move, integer and floating-point compare, arithmetic, and logical instructions, and provide a mechanism for testing and branching.

Vectored interrupts are supported by the CPU to allow multiple interrupt sources to have unique interrupt handlers invoked with no software overhead.

12.3 Block diagram

Figure 48. e200z0h block diagram



12.4 Features

The following is a list of some of the key features of the e200z0h core:

- 32-bit Power Architecture VLE-only programmer's model
- Single issue, 32-bit CPU
- Implements the VLE APU for reduced code footprint
- In-order execution and retirement
- Precise exception handling
- Branch processing unit
  - Dedicated branch address calculation adder
  - Branch acceleration using Branch Target Buffer
- Supports independent instruction and data accesses to different memory subsystems, such as SRAM and Flash memory via independent Instruction and Data bus interface units (BIUs)
- Load/store unit
  - 1 cycle load latency
  - Fully pipelined
  - Big-endian support only
  - Misaligned access support
  - Zero load-to-use pipeline bubbles for aligned transfers
- Power management
  - Low-power design
  - Power saving modes: Doze, Nap, Sleep, and Wait
  - Dynamic power management of execution units
- Testability
  - Synthesizeable, full MuxD scan design

12.4.1 Instruction unit features

The features of the e200z0h Instruction unit are:

- 32-bit instruction fetch path supports fetching of one 32-bit instruction per clock, or up to two 16-bit VLE instructions per clock
- Instruction buffer with 4 entries in e200z0h, each holding a single 32-bit instruction, or a pair of 16-bit instructions
- Dedicated PC incrementer supporting instruction prefetches
- Branch unit with dedicated branch address adder supporting single cycle of execution of certain branches, two cycles for all others

12.4.2 Integer unit features

The e200z0h integer unit supports single cycle execution of most integer instructions:

- 32-bit Arithmetic Unit for arithmetic and comparison operations
- 32-bit Logic Unit for logical operations
- 32-bit priority encoder for count leading zero's function
- 32-bit single cycle barrel shifter for shifts and rotates
- 32-bit mask unit for data masking and insertion
- Divider logic for signed and unsigned divide in 5 to 34 clocks with minimized execution timing
- 8×32 hardware multiplier array supports 1 to 4 cycle 32×32 -> 32 multiply (early out)

12.4.3 Load/Store unit features

The e200z0h load/store unit supports load, store, and the load multiple/store multiple instructions:

- 32-bit effective address adder for data memory address calculations
- Pipelined operation supports throughput of one load or store operation per cycle
- 32-bit interface to memory

12.4.4 e200z0h system bus features

The features of the e200z0h system bus interface are as follows:

- Independent instruction and data buses
- AMBA[g] AHB[h] Lite Rev 2.0 specification with support for ARM v6 AMBA extensions
  - Exclusive access monitor
  - Byte lane strobes
- 32-bit address bus plus attributes and control on each bus
- 32-bit read data bus for instruction interface
- Separate uni-directional 32-bit read data bus and 32-bit write data bus for data interface
- Overlapped, in-order accesses

12.4.5 Nexus 2+ features

The Nexus 2+ module is compliant with Class 2 of the IEEE-ISTO 5001-2003 standard, with additional Class 3 and Class 4 features available. The following features are implemented:

- Program Trace via Branch Trace Messaging (BTM). Branch trace messaging displays program flow discontinuities (such as direct and indirect branches, exceptions), allowing the development tool to interpolate what transpires between the discontinuities. Thus, static code may be traced.
- Ownership Trace via Ownership Trace Messaging (OTM). OTM facilitates ownership trace by providing visibility of which process ID or operating system task is activated.

g. Advanced Microcontroller Bus Architecture

h. Advanced High Performance Bus

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 the processor memory map via the JTAG port. This allows for enhanced download/upload capabilities.
- Watchpoint Messaging via the auxiliary interface
- Watchpoint Trigger enable of Program Trace Messaging
- Auxiliary interface for higher data input/output
  - Configurable (min/max) Message Data Out pins (**nex_mdo[n:0]**)
  - One or two Message Start/End Out pins (**nex_mseo_b[1:0]**)
  - One Read/Write Ready pin (**nex_rdy_b**)
  - One Watchpoint Event pin (**nex_evto_b**)
  - One Event In pin (**nex_evti_b**)
  - One MCKO (Message Clock Out) pin
- Registers for Program Trace, Ownership Trace and Watchpoint Trigger control
- All features controllable and configurable via the JTAG port

12.5 Core registers and programmer's model

This section describes the registers implemented in the e200z0h cores. It includes an overview of registers defined by the Power Architecture platform and provides a detailed description of e200z0h-specific registers. Full descriptions of the architecture-defined register set are provided in the Power Architecture specification.

The Power Architecture defines register-to-register operations for all computational instructions. Source data for these instructions are accessed from the on-chip registers or are provided as immediate values embedded in the opcode. The three-register instruction format allows specification of a target register distinct from the two source registers, thus preserving the original data for use by other instructions. Data is transferred between memory and registers with explicit load and store instructions only.

Figure 49 and *Figure 50* show the e200z0h register set including the registers which are accessible while in Supervisor mode, and the registers which are 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).

Note: e200z0h is a 32-bit implementation of the Power Architecture specification. In this document, register bits are sometimes numbered from bit 0 (Most Significant Bit) to 31 (Least Significant Bit), rather than the Book E numbering scheme of 32:63, thus register bit numbers for some registers in Book E are 32 higher.

Where appropriate, the Book E defined bit numbers are shown in parentheses.

Figure 49. e200z0h SUPERVISOR mode program model SPRs

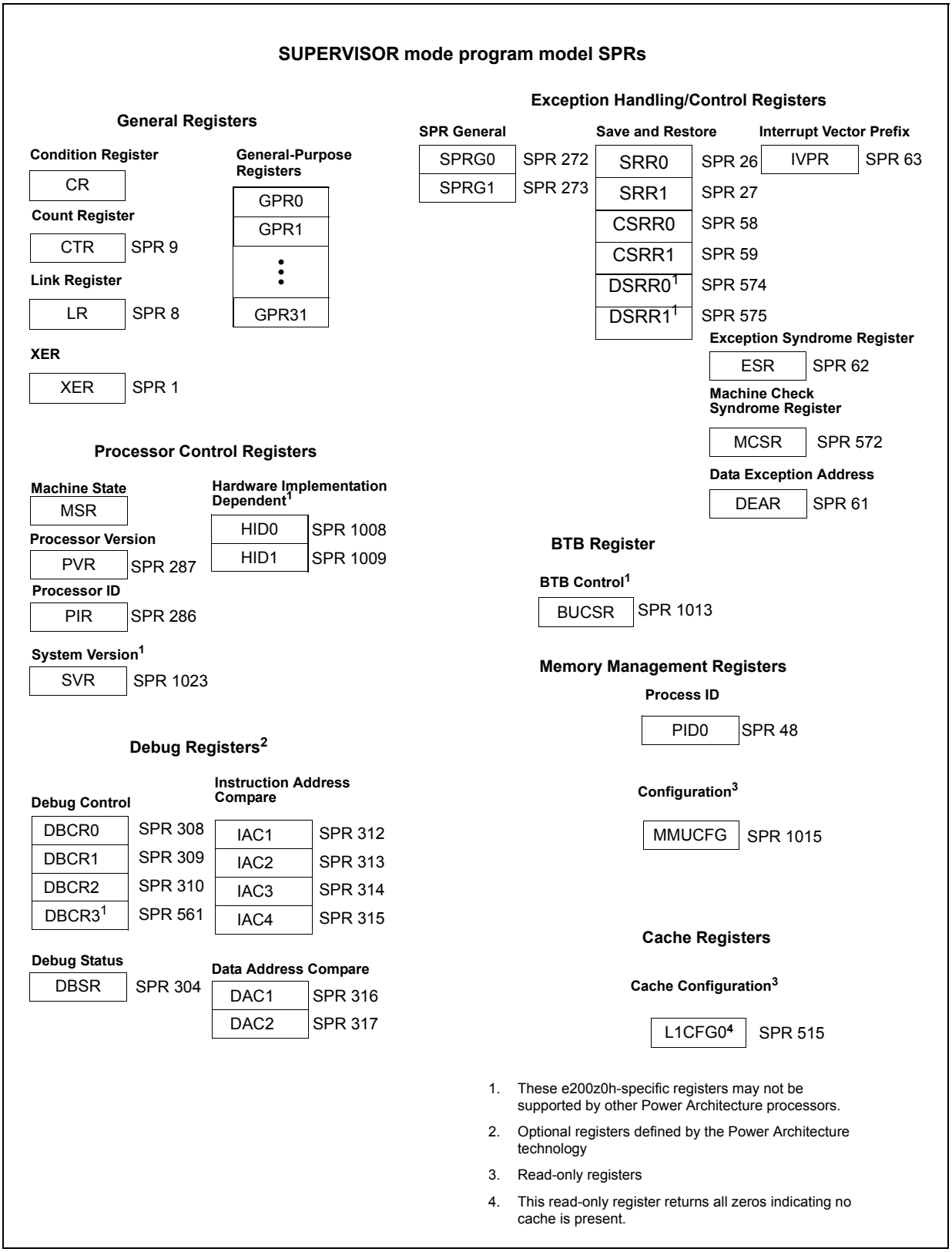
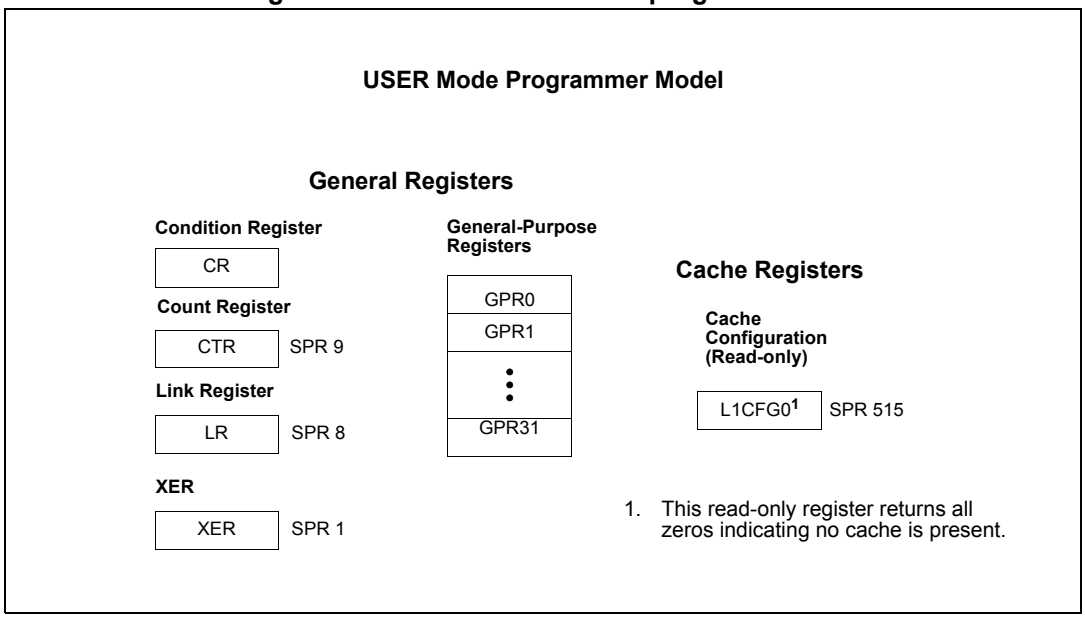


Figure 50. e200z0h USER mode program model



13 System Integration Unit Lite2 (SIUL2)

13.1 Introduction

13.1.1 Overview

The System Integration Unit Lite2 (SIUL2) provides control over all the I/O ports on this device and supports 5 ports with 16 bits of bidirectional, general-purpose input and output signals. It supports 6 external interrupts with independent trigger event configuration, that are combined to generate four internal interrupts versus the interrupt controller. For more details see *Section 13.3.4*. *Figure 51* 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.

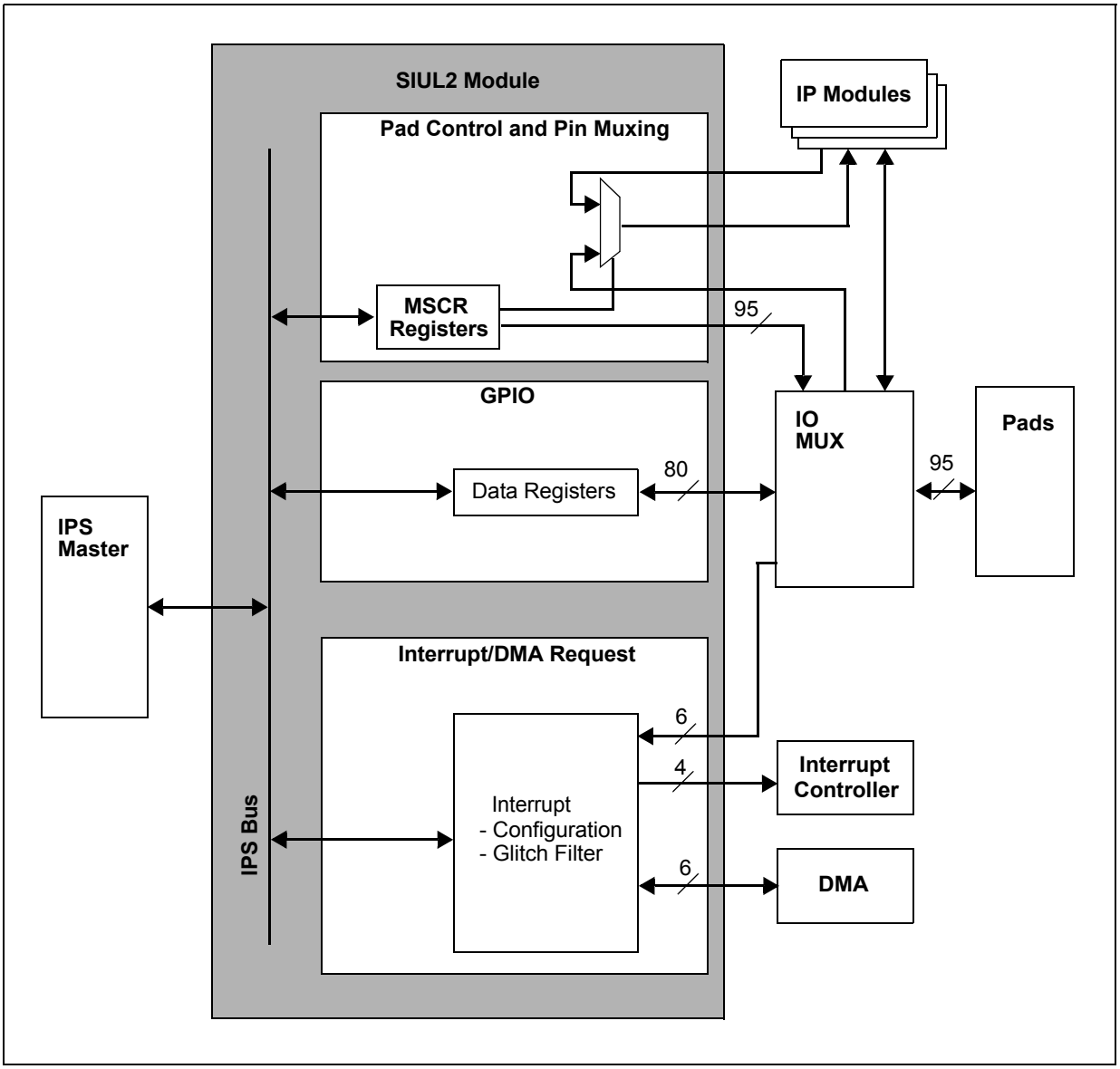
In order to assist in software development, there are three different access mechanisms to the GPIO data registers:

- Access to two 16-bit ports in one access
- Read/Write access to a single bit
- A 16-bit port write with a bit mask, using single 32-bit 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 6 EIRQ pins in the device packages.

Figure 51. SIUL2 block diagram



13.1.2 Features

The SIUL2 supports these distinctive features:

- 5 GPIO ports with data control
  - Drive data to up to 16 independent IO channels
  - Sample data from up to 16 independent IO channels
- Two 16-bit registers can be read/written with one access to a 32-bit port if needed.
- External interrupt/DMA request support with:
  - 4 system interrupt vectors for 6 interrupt sources (6 connected to EIRQ pins) with independent interrupt mask
  - 6 programmable digital glitch filters (one for each EIRQ)
  - Independent DMA channel for each EIRQ pin
  - Edge detection

Additionally the SIUL2 contains the multiplexed signal configuration registers that configure the electrical parameters and settings of all 95 functional pads (80 GPIO + 15 dedicated pads)[(i)]. 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

13.1.3 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 62: Register Protection (REG_PROT)* and *Chapter 7: Device configuration*.

13.2 Memory map and register description

13.2.1 Memory map

Table 112 gives an overview of the SIUL2 registers implemented.

i. Refer to Device Configuration chapter for details on package-specific pin and register implementation.

Table 112. 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 13.2.2.1*
0x0008	SIUL2 MCU ID Register #2 (SIUL2_MIDR2)	R	32	32	0xA800_530F	Saf-Relv	*Section 13.2.2.2*
0x000C–0x000F	Reserved						
0x0010	SIUL2 DMA/Interrupt Status Flag Register 0 (SIUL2_DISR0)	R/W	32	32/ 16/8	0x0000_0000	Saf-Relv	*Section 13.2.2.3*
0x0014–0x0017	Reserved						
0x0018	SIUL2 DMA/Interrupt Request Enable Register 0 (SIUL2_DIRER0)	R/W	32	32/ 16/8	0x0000_0000	Saf-Relv	*Section 13.2.2.4*
0x001C–0x001F	Reserved						
0x0020	SIUL2 DMA/Interrupt Request Select Register 0 (SIUL2_DIRSR0)	R/W	32	32/ 16/8	0x0000_0000	Saf-Relv	*Section 13.2.2.5*
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 13.2.2.6*
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 13.2.2.7*
0x0034-0x0037	Reserved						
0x0038	SIUL2 Interrupt Filter Enable Register 0 (SIUL2_IFER0)	R/W	32	32 /16/8	0x0000_0000	Saf-Lat	*Section 13.2.2.8*
0x003C-0x003F	Reserved						
0x0040–0x00BC	SIUL2 Interrupt Filter Maximum Counter Registers (SIUL2_IFMCR0–SIUL2_IFMCR31)	R/W	32	32	0x0000_0000	Saf-Lat	*Section 13.2.2.9*

Table 112. SIUL2 memory map(Continued)

Address offset	Register name	Access	Size (bits)	Supported access (bit)	Reset value	Register classification (1)	Location
0x00C0	SIUL2 Interrupt Filter Clock Prescaler (SIUL2_IFCPR)	R/W	32	32	0x0000_0000	Saf-Lat	*Section 13.2.2.10*
0x00C4–0x023F	Reserved(7)						
0x0240–0x0A3C	I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR0–SIUL2_MSCR511)(2)	R/W	32	32/16/8(3)	0x0000_0000(4)	Saf-Relv	*Section 13.2.2.11*
0x0A40–0x123C	Multiplexed Signal Configuration Registers for Multiplexed Input Selection (SIUL2_MSCR512–SIUL2_MSCR1023)(2)	R/W	32	32/16/8(3)	0x0000_0000(5)	Saf-Relv	*Section 13.2.2.11*
0x1240–0x12FF	Reserved						
0x1300–0x134F	SIUL2 GPIO Pad Data Out Register (SIUL2_GPDO0 – SIUL2_GPDO79	R/W	8	8	0x00	Saf-Relv	*Section 13.2.2.16*
0x1350–0x14FF	Reserved						
0x1500–0x154F	SIUL2 GPIO Pad Data In Registers (SIUL2_GPDI0 – SIUL2_GPDI79)	R	8	8	0x00	Saf-Relv	*Section 13.2.2.17*
0x1550–0x16FF	Reserved						
0x1700–0x1708	SIUL2 Parallel GPIO Pad Data Out Registers (SIUL2_PGPDO0 – SIUL2_PGPDO4)	R/W	16	16/8	0x0000	Saf-Relv	*Section 13.2.2.18*
0x170A–0x173F	Reserved						
0x1740–0x1748	SIUL2 Parallel GPIO Pad Data In Register (SIUL2_PGPDI0 – SIUL2_PGPDI4)	R	16	16/8	0x0000	Saf-Relv	*Section 13.2.2.19*
0x174A–0x177F	Reserved						
0x1780–0x1790	SIUL2 Masked Parallel GPIO Pad Data Out Register (SIUL2_MPGPDO0–SIUL2_MPGPDO4)	R/W	32	32(6)	0x0000_0000	Saf-Relv	*Section 13.2.2.20*
0x1794–0x17FF	Reserved(7)						

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. Unimplemented MSCRs do not give transfer error for 8 bit access.
4. For some registers refer to *Table 125: SIUL2_MSCRn Reset value*.
5. For some registers refer to *Table 127: SIUL2_MSCRn – Reset value*.
6. 32-bit writes, 8/16/32-bit reads are allowed (8-bit and 16-bit writes are not supported and generate a transfer error)
7. 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.

13.2.2 Register descriptions

This section describes in address order all the SIUL2 registers.

13.2.2.1 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	0	0	0	0	0	0	1

	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(2)	0(2)	0(2)	0(2)	1(2)	0	0	X(3)	X(3)	X(3)	X(3)	X(3)	X(3)	X(3)	X(3)

2. Values are set at factory and cannot be modified
3. Value is set at factory and will vary each revision of device

Figure 52. SIUL2 MCU ID Register #1 (SIUL2_MIDR1)

Table 113. 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: 0x1 SPC570Sx
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: 00000 48-pin QFP 00001 64-pin QFP 01001 100-pin QFP 01101 144-pin QFP
24:27 MAJOR_MASK	Major Mask revision
28:31 MINOR_MASK	Minor Mask revision

13.2.2.2 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)	0(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)

	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)	1(1)	1(1)	1(1)	1(1)

1. Only possible to change value by modifying the side band signals (in the Test Flash). Software can not modify

Figure 53. SIUL2 MCU ID Register #2 (SIUL2_MIDR2)

Table 114. 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 114. 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” (SPC570Sxx) family

13.2.2.3 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 54. SIUL2 DMA/Interrupt Status Flag Register (SIUL2_DISR0)

Table 115. 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

13.2.2.4 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 55. SIUL2 DMA/Interrupt Request Enable Register 0 (SIUL2_DIRER0)

Table 116. 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

13.2.2.5 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 56. SIUL2 DMA/Interrupt Request Select Register 0 (SIUL2_DIRSR0)

Table 117. 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

13.2.2.6 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 57. SIUL2 Interrupt Rising-Edge Event Enable Register 0 (SIUL2_IREER0)

Table 118. 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

13.2.2.7 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 58. SIUL2 Interrupt Falling-Edge Event Enable Register 0 (SIUL2_IFEER0)

Table 119. 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.

13.2.2.8 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 59. SIUL2 Interrupt Filter Enable Register 0 (SIUL2_IFER0)

Table 120. 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

13.2.2.9 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 60. SIUL2 Interrupt Filter Maximum Counter Register (SIUL2_IFMCR0–SIUL2_IFMCR31)

Table 121. SIUL2_IFMCR0–SIUL2_IFMCR31 field descriptions

Field	Description
28:31 MAXCNT	Maximum Interrupt Filter Counter setting 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 = -1 to 3) 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, that is T(DELAY) = (1 to 5) * TCK + MAXCOUNT * TCK (The SW expects the worst filter delay value). 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.

13.2.2.10 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 61. SIUL2 Interrupt Filter Clock Prescaler Register (SIUL2_IFCPR)

Table 122. 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.

13.2.2.11 SIUL2 Multiplexed Signal Configuration Registers (SIUL2_MSCR0–SIUL2_MSCR1023)

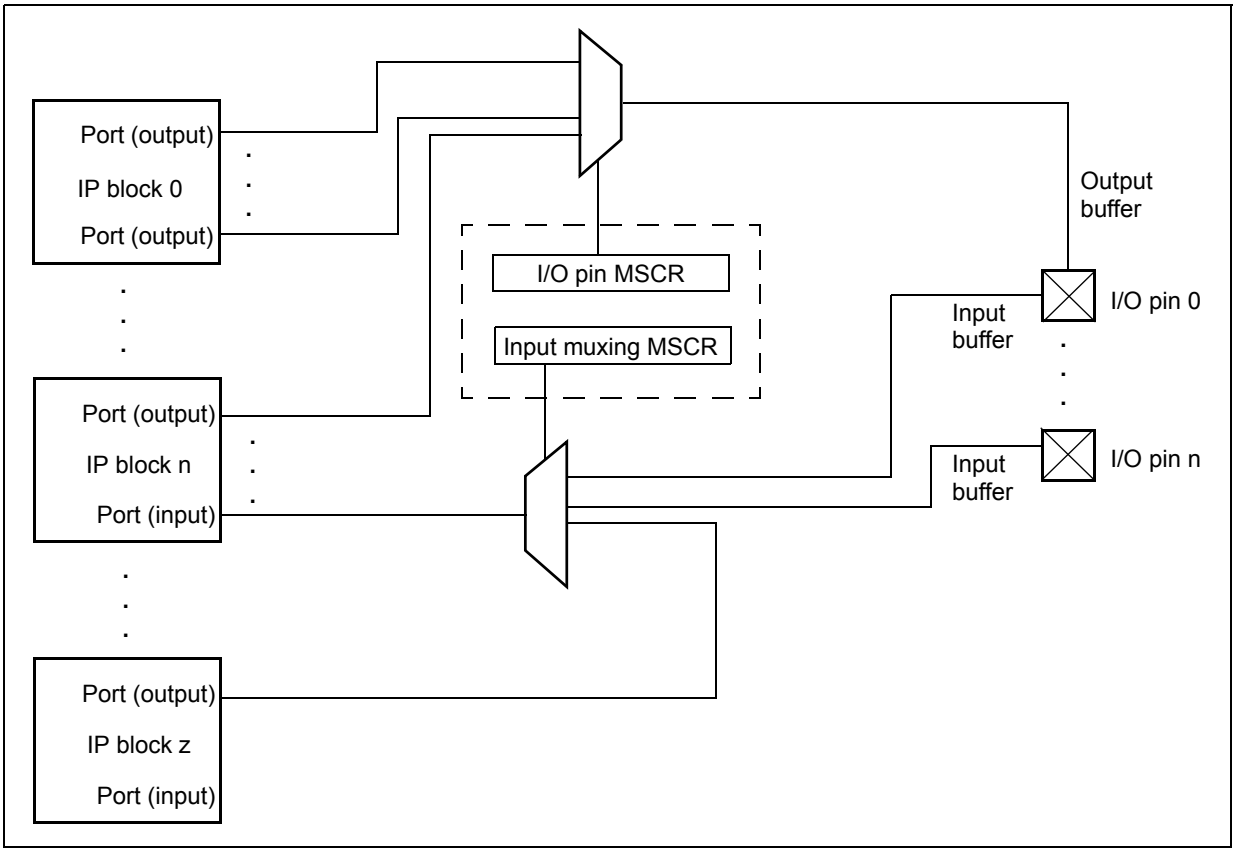
The MSCR s control the I/O pin function and electrical properties for each pin on the device, and are used to select the input signal for IP blocks on the device that have multiple input sources. There is a dedicated MSCR for each I/O pin on the device. Each MSCR is independent of the others, except 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 62*.

Figure 62. MSCR I/O pin and IP block port connectivity



13.2.2.12 I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR0–SIUL2_MSCR511)

The I/O pin MSCR definition is given in *Figure 63*. 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	APC	ILS		IBE	HYS(1)	WPDE	WPUE
W																
Reset(2)	0	0	0	0	0	0	0	0	1	0	0	0	1	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	SSS							
W																
Reset(2)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. HYS bit in MSCR registers from offset 0x240 to 0x37C is set at the time of Reset
2. Reset values of MSCR (0-79) registers are given in *Table 125*.

Figure 63. I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR0–SIUL2_MSCR511)

Table 123. 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)
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
9 APC	Analog Pad Control Used only when the associated destination is a chip pin that supports analog I/O. Enables the pin's analog-input-path switch. 0 Disable (the switch is off) 1 Enable (another module can control the state of the switch)

Table 123. SIUL2_MSCR0–SIUL2_MSCR511 field description(Continued)

Field	Description
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 **Note:**
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
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 13.2.2.14* and *Section 13.2.2.15*.

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 MSCR reset state given in *Table 124* applies to all GPIO pins on the devices, which defaults to input/pull-up enabled.

Table 124. I/O MSCR Reset State Exceptions

Port	Critical Function	SMC	HYS	WPDE	WPUE
PA[5]	JCOMP	1	1	1	0
PA[6]	TCK	1	1	1	0
PA[7]	TMS	1	1	1	1
PA[8]	TDI	1	1	0	1
PA[9]	TDO	1	1	0	1(1)
PA[14]	—	1	1	0	1
PB[11]	ERROR	1	0	0	0
PC[10:15]	—	1	1	0	1
PD[6:7]	—	1	1	0	1
PE[12]	—	1	1	0	1
PF[13]	—	1	1	0	1
PI[14:15]	—	1	1	0	1
PK[12:15]	—	1	1	0	1
PM[0:8]	—	1	1	0	1

1. When JTAG is enabled (JCOMP asserted) during reset, the TDO pin is pulled up during reset, but hi-z after. If JTAG is disabled, the TDO pin defaults to input/pull-up during and after reset.

The Reset values of MSCR(0-79) registers are as mentioned in *Table 125* below:

Table 125. SIUL2_MSCRn Reset value

Reset field	Hexadecimal value
for n=0 to 17	
SIUL_MSCRn_RESET	0x008D0000
for n=18 to 33	
SIUL_MSCRn_RESET	0x00040000
for n=34 to 79	
SIUL_MSCRn_RESET	0x008D0000

13.2.2.13 Multiplexed Signal Configuration Registers for Multiplexed Input Selection (SIUL2_MSCR512–SIUL2_MSCR1023)

The multiplexed input selection MSCR definition is given in *Figure 64*. 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(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	INV	0	0	0	0	0	0	0	SSS							
W																
Reset(1)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. Reset values of most MSCR (512-944) registers are given in *Table 127*.

Figure 64. Multiplexed Signal Configuration Registers for Multiplexed Input Selection (SIUL2_MSCR512–SIUL2_MSCR1023)

Table 126. SIUL2_MSCR512–SIUL2_MSCR1023 field descriptions

Field	Description
16 INV	Invert The input source signal selected by the corresponding SSS field for this MSCR can be inverted before it connects to the destination input with the INV bit. 0 SSS selected input signal is connected directly to the destination input 1 SSS selected input signal is inverted before it is connected to the destination input
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 13.2.2.14* and *Section 13.2.2.15*.

The Reset values of some MSCR(512-944) registers are as mentioned in *Table 127* below:

Table 127. SIUL2_MSCRn – Reset value

Reset field	Hexdecimal value
for n=512 to 513	
SIUL_MSCRn_RESET	0x00000000
for n=560 to 561	
SIUL_MSCRn_RESET	0x00000000
for n=624 to 632	
SIUL_MSCRn_RESET	0x00000000
for n=688 to 719	
SIUL_MSCRn_RESET	0x00000000

Table 127. SIUL2_MSCRn – Reset value(Continued)

Reset field	Hexdecimal value
for n=752 to 763	
SIUL_MSCRn_RESET	0x00000000
for n= 767	
SIUL_MSCRn_RESET	0x00000000
for n=816 to 821	
SIUL_MSCRn_RESET	0x00000000
for n= 944	
SIUL_MSCRn_RESET	0x00000000

13.2.2.14 Chip-pin MSCR assignments and related information

For chip-pin MSCR assignments, pin types, APC support, and SSS values, please refer to *Chapter 4: Signal description*.

13.2.2.15 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.

13.2.2.16 SIUL2 GPIO Pad Data Out Register (SIUL2_GPDO0–SIUL2_GPDO79)

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: 0x1300-0x134F 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 65. SIUL2 GPIO Pad Data Out Register (SIUL2_GPDO0–SIUL2_GPDO79)

Table 128. SIUL2_GPDO0–SIUL2_GPDO79 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

13.2.2.17 SIUL2 GPIO Pad Data Input Registers (SIUL2_GPDI0–SIUL2_GPDI79)

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: 0x1500-0x154F 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 66. SIUL2 GPIO Pad Data Input Registers (SIUL2_GPDI0–SIUL2_GPDI79)

Note: GPDI registers reset values depend on the values on the PADs. After reset, registers will reflect the values on PADs .

Table 129. SIUL2_GPDI0–SIUL2_GPDI79 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

13.2.2.18 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 4) 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 67. SIUL2 Parallel GPIO Pad Data Out Register (SIUL2_PGPDO*n*)

Table 130. 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_PGPDO4.PPDO[15] = PDO[79]

13.2.2.19 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 4) 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 68. 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 131. 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]

13.2.2.20 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 4) 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 69. SIUL2 Masked Parallel GPIO Pad Data Out Register (SIUL2_MPGPDO*n*)

Table 132. 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]

13.3 Functional description

13.3.1 General

This section provides a complete functional description of the SIUL2.

13.3.2 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 (e.g. 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.

13.3.3 General purpose input or output pads (GPIO)

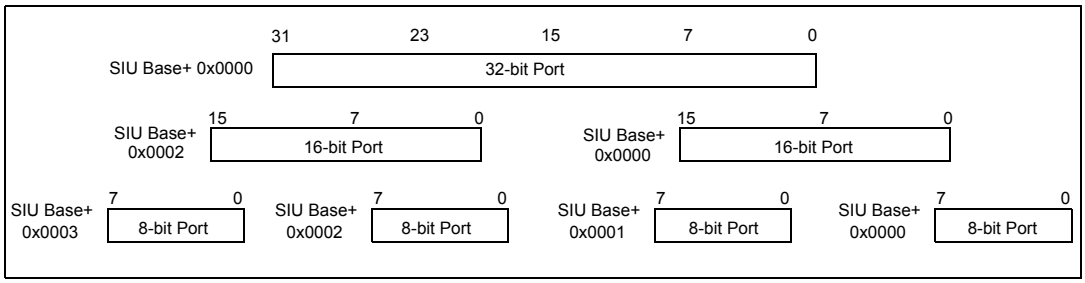
The SIUL2 allows each pad to be configured as either a General Purpose Input Output pad (GPIO), and as one or more alternate functions (input or output), the function of which is determined by the peripheral that will use the pad.

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 70*, all port accesses are identical with each read or write being performed only at a different location to access a different port width.

Figure 70. 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 bi-directional operation impacts some performances (e.g. ADC accuracy), or is not required for a specific pad function it is acceptable to limit the functionality of some pads to “Input Only”.

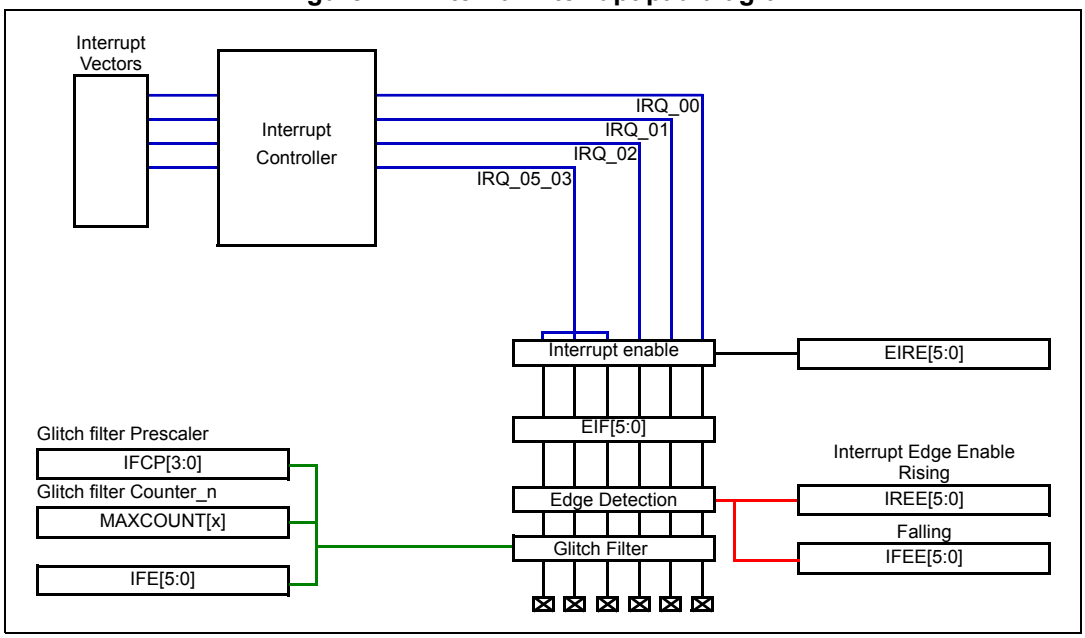
13.3.4 External interrupts/DMA requests (EIRQ pins)

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 71* for an overview of the external interrupt implementation.

Figure 71. 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.

13.3.4.1 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.

13.3.4.2 External interrupt management

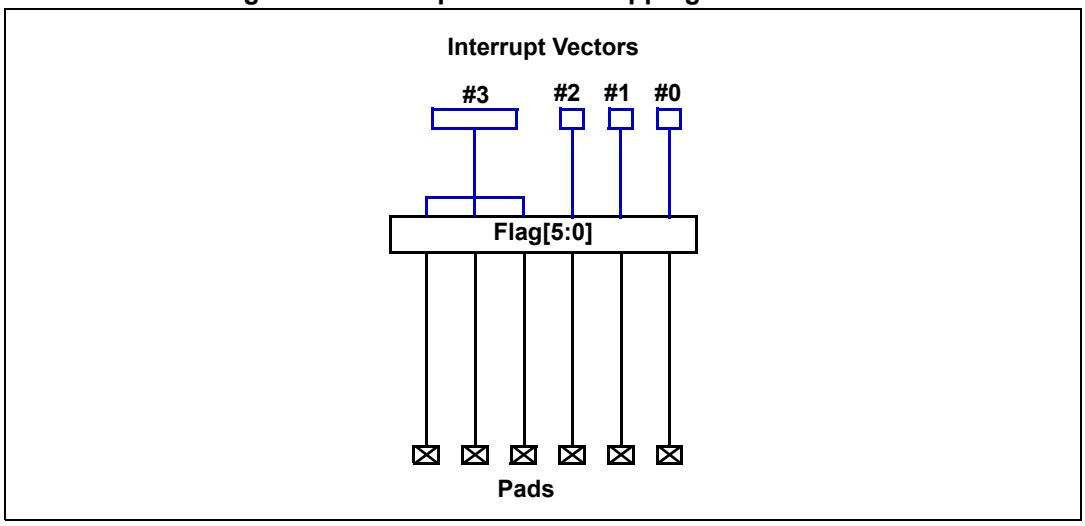
Each interrupt can be enabled or disabled independently. This can be performed using a single rolled up register (SIUL2_DIRER0 – *Figure 55*). A pad defined as an external interrupt can be configured by the user to recognize interrupts with an active rising edge, an active falling edge or both edges 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 SPC570Sx device supports 4 SIUL2 interrupt vectors aggregating a total of 6 external interrupt sources as shown in *Figure 72*.

Figure 72. Interrupt to vector mapping at MCU level



13.3.4.3 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 133*.

Table 133. Interrupt source mapping to SIUL2 interrupt request output

Interrupt requests	EIRQ pins
0	EIRQ0
1	EIRQ1
2	EIRQ2
3	EIRQ3 | EIRQ4 | EIRQ5

13.3.4.4 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 for the Interrupt Vector table.

13.3.4.5 DMA requests

The 6 EIRQ pins on the device map to an independent DMA request channel in the DMA controller.

Note: See Section 7.3.6: DMA controller configuration for DMA request mapping table.

14 Crossbar switch (XBAR)

14.1 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.

14.1.1 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

14.1.2 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 134. 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 14.1.2.1*
0x0010 + *n**0x100	XBAR0 Control Register *n* (XBAR_CRS*n*)	32	R/W	*Section 14.1.2.2*
(n=0 to 2)				
0x0800 + *n**0x100	Master General Purpose Control Register *n* (XBAR_MGPCR*n*)	32	R/W	*Section 14.1.2.3*

14.1.2.1 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	M7			0	M6			0	M5			0	M4		
W																
Reset	0	See note (1)			0	See note (1)			0	See note (1)			0	See note (1)		

	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	See note (1)			0	See note (1)			0	See note (1)			0	See note (1)		

1. See *Chapter 7: Device configuration*: *Section 7.3.2.1.3: XBAR_PRSn register reset values*.

Figure 73. Priority Registers Slave (XBAR_PRS*n*)

Table 135. XBAR_PRS*n* field descriptions

Field	Description
1:3 M7	Master 7 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 priority when accessing the slave port. 100 This master has level 5 priority when accessing the slave port. 101 This master has level 6 priority when accessing the slave port. 110 This master has level 7 priority when accessing the slave port. 111 This master has level 8 (lowest) priority when accessing the slave port.
5:7 M6	Master 6 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 priority when accessing the slave port. 100 This master has level 5 priority when accessing the slave port. 101 This master has level 6 priority when accessing the slave port. 110 This master has level 7 priority when accessing the slave port. 111 This master has level 8 (lowest) priority when accessing the slave port.
9:11 M5	Master 5 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 priority when accessing the slave port. 100 This master has level 5 priority when accessing the slave port. 101 This master has level 6 priority when accessing the slave port. 110 This master has level 7 priority when accessing the slave port. 111 This master has level 8 (lowest) priority when accessing the slave port.
13:15 M4	Master 4 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 priority when accessing the slave port. 100 This master has level 5 priority when accessing the slave port. 101 This master has level 6 priority when accessing the slave port. 110 This master has level 7 priority when accessing the slave port. 111 This master has level 8 (lowest) priority when accessing the slave port.
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 priority when accessing the slave port. 100 This master has level 5 priority when accessing the slave port. 101 This master has level 6 priority when accessing the slave port. 110 This master has level 7 priority when accessing the slave port. 111 This master has level 8 (lowest) priority when accessing the slave port.

Table 135. XBAR_PRS*n* field descriptions(Continued)

Field	Description
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 priority when accessing the slave port. 100 This master has level 5 priority when accessing the slave port. 101 This master has level 6 priority when accessing the slave port. 110 This master has level 7 priority when accessing the slave port. 111 This master has level 8 (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 priority when accessing the slave port. 100 This master has level 5 priority when accessing the slave port. 101 This master has level 6 priority when accessing the slave port. 110 This master has level 7 priority when accessing the slave port. 111 This master has level 8 (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 priority when accessing the slave port. 100 This master has level 5 priority when accessing the slave port. 101 This master has level 6 priority when accessing the slave port. 110 This master has level 7 priority when accessing the slave port. 111 This master has level 8 (lowest) priority when accessing the slave port.

14.1.2.2 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	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	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 74. XBAR Control Register (XBAR_CRS*n*)

Table 136. 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.
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

14.1.2.3 Master General Purpose Control Registers (XBAR_MGPCRn)

Offset: 0x0800 + n*0x100 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	0	0	0	0	0	0	0	0	0	0	0	0	AULB		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 75. Master General Purpose Control Registers (XBAR_MGPCRn)

Table 137. XBAR_MGPCRn field descriptions

Field	Description
29:31 AULB	Arbitrate on Undefined Length Bursts These bits are used to select the arbitration policy during undefined length bursts by this master. These bits are initialized by hardware reset. The reset value is 000. 000 No arbitration will be allowed during an undefined length burst. 001 Arbitration will be allowed at any time during an undefined length burst. 010 Arbitration will be allowed after 4 beats of an undefined length burst. 011 Arbitration will be allowed after 8 beats of an undefined length burst. 100 Arbitration(1) will be allowed after 16 beats of an undefined length burst. 101 Reserved 110 Reserved 111 Reserved

1. Undefined length bursts are supported only by CPU as bus master for instruction accesses

14.2 Functional description

This section provides the functional details of the crossbar switch.

14.2.1 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].

14.2.2 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.

14.2.3 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.

14.2.3.1 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

14.2.3.2 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.

14.3 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.

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 attribute information contains:

- Several AHB signals such as htrans, hmaster, hwrite, hsize, hprot, hunalign, hburst
- decorated attributes 'hdecorated/hdecor'

Note: Decorated attribute signals are all zeros.

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
- 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 76* and *Figure 77*.

Figure 76. XBIC system block diagram

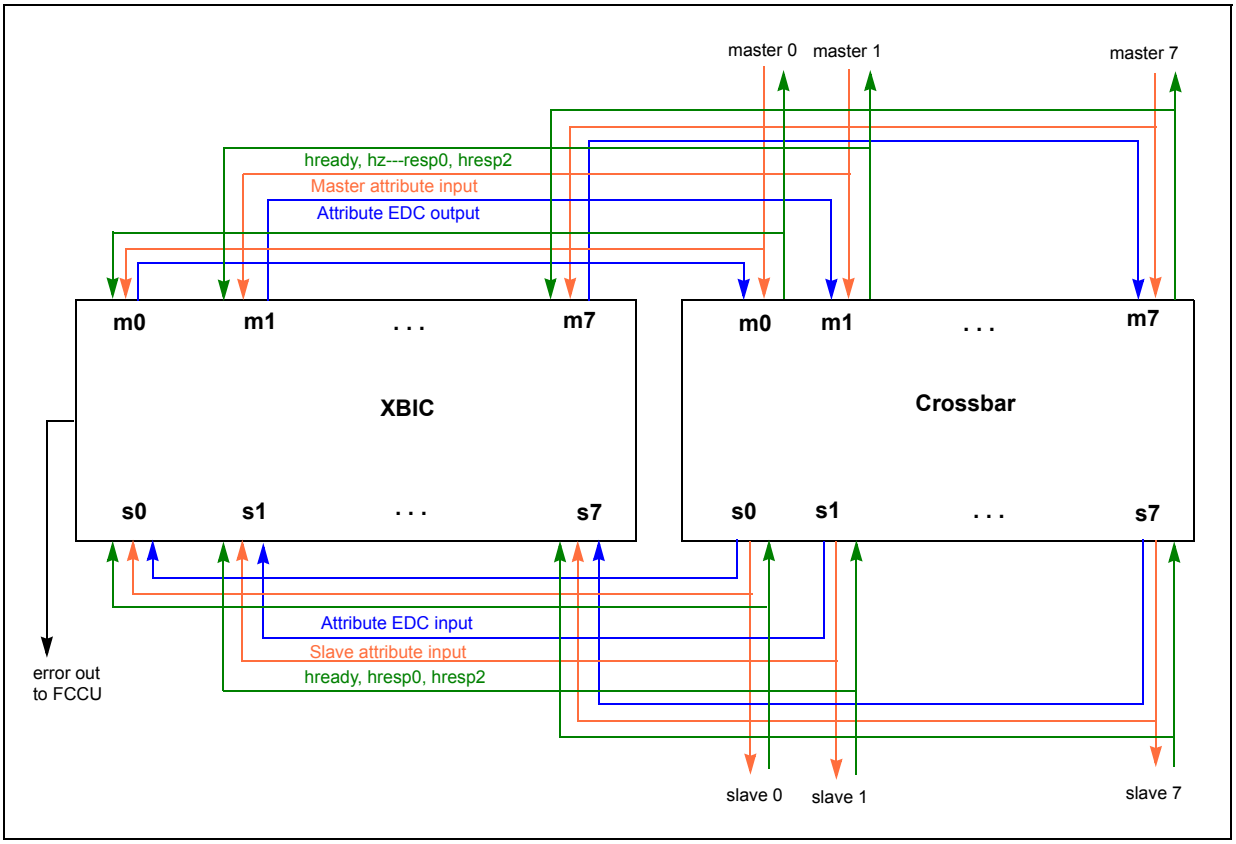
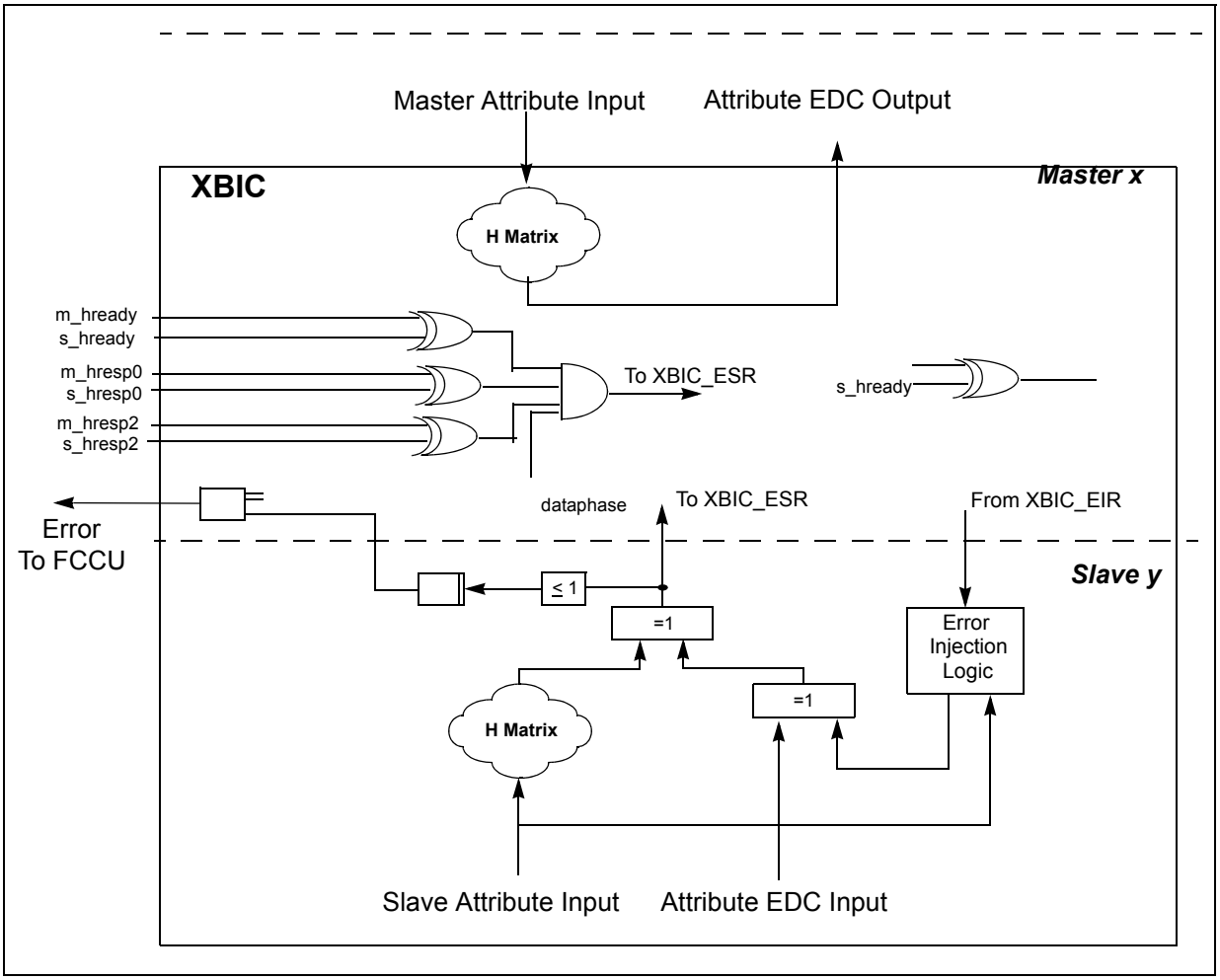


Figure 77. XBIC block diagram



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 138*.

Table 138. 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(1)	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 78. XBIC Module Control Register (XBIC_MCR)

Table 139. 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
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

Table 139. XBIC_MCR field descriptions(Continued)

Field	Description
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 79. XBIC Error Injection Register (XBIC_EIR)

Table 140. 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 Error injection is enabled for the designated slave port. Other slave ports are unaffected.

Table 140. XBIC_EIR field descriptions(Continued)

Field	Description
20:23 MST	Target Master ID 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.

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	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 80. XBIC Error Status Register (XBIC_ESR)

Table 141. 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

Table 141. XBIC_ESR field descriptions(Continued)

Field	Description
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
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 143*.

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 81. Error Address Register (XBIC_EAR)

Table 142. 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 143*. 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 143. 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

Table 143. Hexadecimal attribute single-bit error syndromes(Continued)

Signal	SYN	Signal	SYN	Signal	SYN	Signal	SYN
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 144*.

Table 144. 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 32-bit transactions
- 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.

Note: 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. *e200z0h is a 32-bit microprocessor. 64-bit accesses are not possible.*

16.2 Memory map/register definition

Table 145. 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)	
0x030–0x3F	Reserved(1)	
0x040–0x83	Reserved(2)	
0x084–0xFF	Reserved(1)	
0x100	Peripheral Access Control Register A (PBRIDGE_PACRA)	*Section 16.2.2*

Table 145. Peripheral bridge memory map(Continued)

Offset address	Register	Location
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	Peripheral Access Control Register G (PBRIDGE_PACRG)	Section 16.2.2
0x11C	Peripheral Access Control Register H (PBRIDGE_PACRH)	Section 16.2.2
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
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

Table 145. Peripheral bridge memory map(Continued)

Offset address	Register	Location
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 82. Master Privilege Register A (MPRA)

The MPROT*n* fields are defined as shown in *Figure 83*.

	0	1	2	3
R	0	MTR	MTW	MPL
W				

Figure 83. MPROT*n* fields

Table 146. 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
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 147 shows the top-level structure of the PACRs.

Table 147. 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	PACRE	PACR32	PACR33	PACR34	PACR35	PACR36	PACR37	PACR38	PACR39
0x0114	PACRF	PACR40	PACR41	PACR42	PACR43	PACR44	PACR45	PACR46	PACR47
0x0118	PACRG	PACR48	PACR49	PACR50	PACR51	PACR52	PACR53	PACR54	PACR55
0x011C	PACRH	PACR56	PACR57	PACR58	PACR59	PACR60	PACR61	PACR62	PACR63

Offset: 0x100 (PBRIDGE_PACRA)
0x104 (PBRIDGE_PACRB)
0x108 (PBRIDGE_PACRC)
0x10C (PBRIDGE_PACRD)
0x110 (PBRIDGE_PACRE)
0x114 (PBRIDGE_PACRF)
0x118 (PBRIDGE_PACRG)
0x11C (PBRIDGE_PACRH)

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 84. Peripheral Access Control Registers (PBRIDGE_PACR*x*)

Each PACR*n* field is defined as shown in *Figure 85*.

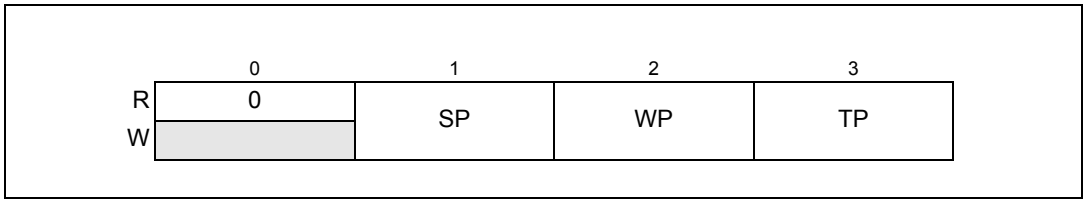


Figure 85. PACR*n* fields

Table 148. 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 149 shows the top-level structure of the OPACR registers.

Table 149. 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

Table 149. OPACR memory map(Continued)

Offset	Register	[0:3]	[4:7]	[8:11]	[12:15]	[16:19]	[20:23]	[24:27]	[28:31]
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
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

Table 149. OPACR memory map(Continued)

Offset	Register	[0:3]	[4:7]	[8:11]	[12:15]	[16:19]	[20:23]	[24:27]	[28:31]
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), 0x144 (PBRIDGE_OPACRB), 0x148 (PBRIDGE_OPACRC), 0x14C (PBRIDGE_OPACRD), 0x150 (PBRIDGE_OPACRE), 0x154 (PBRIDGE_OPACRF), 0x158 (PBRIDGE_OPACRG), 0x15C (PBRIDGE_OPACRH), 0x160 (PBRIDGE_OPACRI), 0x164 (PBRIDGE_OPACRJ), 0x168 (PBRIDGE_OPACRK), 0x16C (PBRIDGE_OPACRL), 0x170 (PBRIDGE_OPACRM), 0x174 (PBRIDGE_OPACRN), 0x178 (PBRIDGE_OPACRO), 0x17C (PBRIDGE_OPACRP), 0x140 (PBRIDGE_OPACRQ), 0x140 (PBRIDGE_OPACRR), 0x140 (PBRIDGE_OPACRS), 0x140 (PBRIDGE_OPACRT), 0x140 (PBRIDGE_OPACRU), 0x140 (PBRIDGE_OPACRV), 0x140 (PBRIDGE_OPACRW), 0x140 (PBRIDGE_OPACRX), 0x140 (PBRIDGE_OPACRIY), 0x140 (PBRIDGE_OPACRZ), 0x140 (PBRIDGE_OPACRAA), 0x140 (PBRIDGE_OPACRAB), 0x140 (PBRIDGE_OPACRAC), 0x140 (PBRIDGE_OPACRAD), 0x140 (PBRIDGE_OPACRAE), 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 86. Off-Platform Peripheral Access Control Registers (PBRIDGE_OPACR*x*)

The OPACR*n* field is defined as in *Figure 87*.

	0	1	2	3
R W	0	SP	WP	TP

Figure 87. OPACR*n* fields

Table 150. 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 150. 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.

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 Introduction

The System Memory Protection Unit (SMPU) provides hardware access control for all memory references generated in a device. Using region descriptors which define memory spaces and their associated access rights, the SMPU concurrently monitors all system bus transactions and evaluates the appropriateness of each transfer. 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 a protection error response.

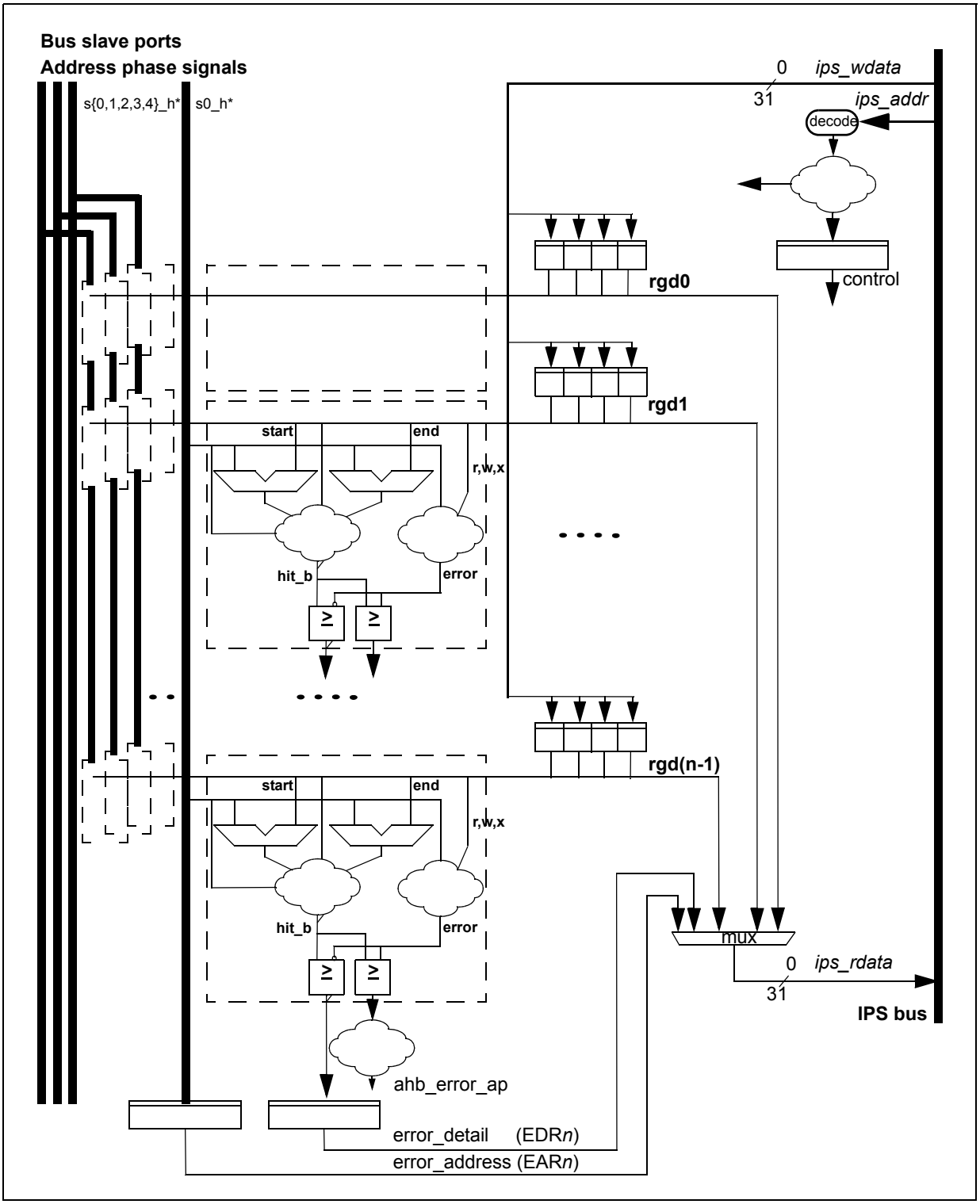
The SMPU implements a set of program-visible region descriptors which are used to monitor all system bus addresses. The result is a hardware structure with a two-dimensional connection matrix, where the region descriptors represent one dimension and the individual system bus addresses and attributes are the second dimension.

17.2 Block diagram

A simplified block diagram of the SMPU module is shown in *Figure 88*. The hardware's two-dimensional connection matrix is clearly visible with the basic access evaluation "macro" shown as the replicated submodule block. The AHB[j] bus slave ports (s{0,1,2,3,4}_h*) are shown on the left side of the diagram, the region descriptor registers in the middle and the IPS bus interface (ips_*) on the right side. The evaluation macro contains the two magnitude comparators connected to the start and end address registers from each region descriptor (rgd*n*) as well as the combinational logic blocks to determine the region hit and the access protection error. For information on the details of the access evaluation macro, see *Section 17.7.1*.

j. ARM's Advanced High-performance Bus

Figure 88. SMPU block diagram



17.3 Features

The SMPU implements a two-dimensional hardware array of memory region descriptors and the crossbar slave AHB ports to continuously monitor the legality of every memory reference generated by each bus master in the system. The feature set includes:

- Support for 8 memory region descriptors
  - Specification of start and end addresses provide granularity for region sizes from 1 byte to 4 GB
  - 3 bus masters that support the traditional {read, write, execute} permissions with independent definitions for supervisor and user mode accesses
  - Automatic hardware maintenance of the region descriptor valid bit removes issues associated with maintaining a coherent image of the descriptor
  - Alternate memory view of the access control word for each descriptor provides an efficient mechanism to dynamically alter only the access rights of a descriptor
  - For overlapping region descriptors, priority is given to permission granting over access denying as this approach provides more flexibility to system software. See *Section 17.7.2*, for details and *Section 17.10*, as example
- Support for 5 AHB slave port connections
  - SMPU hardware continuously monitors every AHB slave port access using the preprogrammed memory region descriptors
  - An access protection error is detected if a memory reference does not hit in any memory region or the reference is flagged as illegal in *all* memory regions where it does hit. In the event of an access error, the AHB reference is terminated with an error response and the SMPU inhibits the bus cycle being sent to the targeted slave device. This results in a Machine Check Exception
  - 64-bit error registers, one for each AHB slave port, capture the last faulting address, attributes and “detail” information
- Global SMPU enable/disable control bit provides a mechanism to easily load region descriptors during system startup or allow complete access rights during debug with the module disabled

17.4 Modes of operation

The SMPU module does not support any special modes of operation. As a memory-mapped device located on the platform’s high-speed system bus, it responds based strictly on the memory addresses of the connected system buses. The IPS bus is used to access the SMPU’s programming model and the memory protection functions are evaluated on a reference-by-reference basis using the addresses from the AHB system bus port(s).

Power dissipation is minimized when the SMPU’s global enable/disable bit is cleared (SMPU_CESR[VLD] = 0).

17.5 External signal description

The SMPU module does not included any external interface.

17.6 Memory map and register definition

The SMPU module provides an IPS programming model mapped to an SPP-standard on-platform 16 KB space. The programming model is partitioned into three groups:

- Control/status registers
- The data structure containing the region descriptors
- The alternate view of the region descriptor access control values

The programming model can only be referenced using 32-bit (word) accesses. Attempted references using different access sizes, to undefined (reserved) addresses, or with a non-supported access type (for example, a write to a read-only register or a read of a write-only register) generate an IPS error termination.

The SMPU programming model map is shown in *Table 151*.

Table 151. SMPU memory map

Offset address	Register description	Location
0x0000	SMPU Control/Error Status Register (SMPU_CESR)	*Section 17.6.1*.
0x0004–0x000F	Reserved	
ERROR registers (n=0 to 4)		
0x0010 + *n**0x08	SMPU Error Address Register, Slave Port n (SMPU_EAR*n*)	*Section 17.6.2*
0x0014 + *n**0x08	SMPU Error Detail Register, Slave Port n (SMPU_EDR*n*)	*Section 17.6.3*
0x0038–0x03FF	Reserved	
RGD registers (n=0 to 7)		
0x400 + *n**0x10	SMPU Region Descriptor n – Word 0 (SMPU_RGD*n*_WORD0)	*Section 17.6.4.1*
0x404 + *n**0x10	SMPU Region Descriptor n – Word 1 (SMPU_RGD*n*_WORD1)	*Section 17.6.4.2*
0x408 + *n**0x10	SMPU Region Descriptor n – Word 2 (SMPU_RGD*n*_WORD2_FMT0)	*Section 17.6.4.3*
0x40C + *n**0x10	SMPU Region Descriptor n – Word 3 (SMPU_RGD*n*_WORD3)	*Section 17.6.4.4*
0x0480–0x07FF	Reserved	
(n=0 to 7)		
0x0800 + *n**0x4	SMPU RGD Alternate Access Control n (SMPU_RGDAAC*n*)	*Section 17.6.5*
0x0820–0x3FFF	Reserved	

17.6.1 SMPU Control/Error Status Register (SMPU_CESR)

The SMPU_CESR provides one byte of error status plus three bytes of configuration information. A global SMPU enable/disable bit is also included in this register.

Offset: 0x0000 Access: Read/Partial 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	SPERR	0 0 0	1 0 0 0	HRL	NSP	NRGD	0 0 0 0	0 0 0 VLD
W	w1c							
Reset	0 0 0 0	0 0 0 0	1 0 0 0	0 0 0 0	0 1 0 1	0 0 0 0	0 0 0 0	0 0 0 0

Figure 89. SMPU Control/Error Status Register (SMPU_CESR)

Table 152. SMPU_CESR field descriptions

Field	Description
0:4 SPERR	Slave Port *n* Error, where the slave port number matches the bit number Each bit in this field represents a flag maintained by the SMPU for signaling the presence of a captured error contained in the SMPU_EAR*n* and SMPU_EDR*n* registers. The individual bit is set when the hardware detects an error and records the faulting address and attributes. It is cleared when the corresponding bit is written as a logical one. If another error is captured at the exact same cycle as a write of a logical one, this flag remains set. A “find first one” instruction (or equivalent) can be used to detect the presence of a captured error. 0 The corresponding SMPU_EAR*n*/SMPU_EDR*n* registers do not contain a captured error 1 The corresponding SMPU_EAR*n*/SMPU_EDR*n* registers do contain a captured error
12:15 HRL	Hardware Revision Level This 4-bit read-only field specifies the SMPU’s hardware and definition revision level. It can be read by software to determine the functional definition of the module.
16:19 NSP	Number of Slave Ports This 4-bit read-only field specifies the number of slave ports connected to the SMPU.
20:23 NRGD	Number of Region Descriptors This 4-bit read-only field specifies the number of region descriptors implemented in the SMPU. The defined encodings include: 0000 8 region descriptors 0001 12 region descriptors 1000 16 region descriptors
31 VLD	Valid This bit provides a global enable/disable for the SMPU. 0 The SMPU is disabled 1 The SMPU is enabled While the SMPU is disabled, all accesses from all bus masters are allowed.

17.6.2 SMPU Error Address Register, Slave Port *n* (SMPU_EAR*n*)

When the SMPU detects an access error on slave port n, the 32-bit reference address is captured in this read-only register and the corresponding bit in the SMPU_CESR[SPERR] field set. Additional information about the faulting access is captured in the corresponding SMPU_EDR*n* register at the same time. This register and the corresponding SMPU_EDR*n* register contain the most recent access error; there are no hardware interlocks with the SMPU_CESR[SPERR] field as the error registers are always loaded upon the occurrence of each protection violation.

Note: In case of instruction fetch violation, the captured address is the referenced address + 0x4 (because of pre-fetching in the pipeline).

Offset: 0x0010 + *n**0x8 (*n*=0 to 4) Access: 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	EADDR																															
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 90. SMPU Error Address Register, Slave Port *n* (SMPU_EAR*n*)

Table 153. SMPU_EAR*n* field descriptions

Field	Description
0:31 EADDR	Error Address This read-only field is the reference address from slave port *n* that generated the access error.

17.6.3 SMPU Error Detail Register, Slave Port *n* (SMPU_EDR*n*)

When the SMPU detects an access error on slave port n, 32 bits of error detail are captured in this read-only register and the corresponding bit in the SMPU_CESR[SPERR] field set. Information on the faulting address is captured in the corresponding SMPU_EAR*n* register at the same time.

Note: Corresponding SMPU_EARn register contain the most recent access error; there are no hardware interlocks with the SMPU_CESR[SPERR] field as the error registers are always loaded upon the occurrence of each protection violation.

Offset: 0x0014 + *n**0x8 (*n*=0 to 4) Access: 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	EACD																EPID								EMN				EATTR			ERW
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 91. SMPU Error Detail Register, Slave Port *n* (SMPU_EDR*n*)

Table 154. SMPU_EDR*n* field descriptions

Field	Description
0:15 EACD	Error Access Control Detail This 16-bit read-only field implements one bit per region descriptor and is an indication of the region descriptor hit where the error occurred. The SMPU performs a reference-by-reference evaluation to determine the presence/absence of an access error. When an error is detected, the hit-qualified access control vector is captured in this field. If the SMPU_EDR*n* register contains a captured error and the EACD field is all zeroes, this signals an access that did not hit in any region descriptor. All non-zero EACD values signal references that hit in a region descriptor(s), but failed due to a protection error as defined by the specific set bits. If only a single EACD bit is set, then the protection error was caused by a single non-overlapping region descriptor. If two or more EACD bits are set, then the protection error was caused in an overlapping set of region descriptors.
16:23 EPID	Error Process Identification This 8-bit read-only field records the process identifier of the faulting reference. The process identifier is typically driven only by processor cores; for other bus masters, this field is cleared.
24:27 EMN	Error Master Number This 4-bit read-only field records the logical master number of the faulting reference. This field is used to determine the bus master that generated the access error.
28:30 EATTR	Error Attributes This 3-bit read-only field records attribute information about the faulting reference. The supported encodings are defined as: 000 User mode, instruction access 001 User mode, data access 010 Supervisor mode, instruction access 011 Supervisor mode, data access All other encodings are reserved. For non-core bus masters, the access attribute information is typically wired to supervisor, data (0b011).
31 ERW	Error Read/Write This 1-bit read-only field signals the access type (read, write) of the faulting reference. 0 Read 1 Write

17.6.4 SMPU Region Descriptor *n* (SMPU_RGD)

Each 128-bit (four 32-bit word) region descriptor specifies a given memory space and the access attributes associated with that space. The descriptor definition is the very essence of the operation of the System Memory Protection Unit.

The region descriptors are organized sequentially in the SMPU's programming model and each of the four 32-bit words are detailed in the subsequent sections.

17.6.4.1 SMPU Region Descriptor n – Word 0 (SMPU_RGD*n*_WORD0)

The first word of the SMPU region descriptor defines the start address of the memory region. Writes to this word clear the region descriptor's valid bit (see *Section 17.6.4.4,* for more information).

Offset: 0x0400 + *n**0x10 (*n*=0 to 7) 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	SRTADDR																															
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 92. SMPU Region Descriptor *n*, Word 0 Register (SMPU_RGDn_WORD0)

Table 155. SMPU_RGDn_WORD0 field descriptions

Field	Description
0:31 SRTADDR	Start Address This field defines the most significant bits of the start address of the memory region.

17.6.4.2 SMPU Region Descriptor *n* – Word 1 (SMPU_RGDn_WORD1)

The second word of the SMPU region descriptor defines the end address of the memory region. Writes to this word clear the region descriptor's valid bit (see *Section 17.6.4.4*, for more information).

Offset: 0x0404 + *n**0x10 (*n*=0 to 7) 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	ENDADDR																															
Reset (n=0)	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
Reset (n>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	0

Figure 93. SMPU Region Descriptor *n* – Word 1 Register (SMPU_RGDn_WORD1)

Table 156. SMPU_RGDn_WORD1 field descriptions

Field	Description
0:31 ENDADDR	End Address This field defines the most significant bits of the end address of the memory region. There are no hardware checks to verify that ENDADDR >= SRTADDR: it is software's responsibility to properly load these region descriptor fields.

17.6.4.3 SMPU Region Descriptor *n* – Word 2 FMT0 (SMPU_RGDn_WORD2_FMT0)

The third word of the SMPU region descriptor defines the access control rights of the memory region. Bus masters 0–2 have a 6-bit field defining separate privilege rights for user and supervisor mode accesses as well as the optional inclusion of a process identification field within the definition. For these fields, the bus master number refers to the logical master number as specified in *Table 34: SMPU_0 logical bus master assignments* of *Chapter 7: Device configuration*.

For the processor privilege rights, there are three flags associated with this function: {read, write, execute}. In this context, these flags follow the traditional definition:

- Read (r) permission refers to the ability to access the referenced memory address using an operand (data) fetch.
- Write (w) permission refers to the ability to update the referenced memory address using a store (data) instruction.
- Execute (x) permission refers to the ability to read the referenced memory address using an instruction fetch.

Writes to this word clear the region descriptor's valid bit (see *Section 17.6.4.4,* for more information). Since it is also expected that system software may adjust only the access controls within a region descriptor (SMPU_RGDn_WORD2_FMT0) as different tasks execute, an alternate programming view of this 32-bit entity is provided. If only the access controls are being updated, this operation should be performed by writing to SMPU_RGDAAC*n* (Alternate Access Control *n*) as stores to these locations do not affect the descriptor's valid bit.

Offset: 0x0408 + *n**0x10 (*n*=0 to 7) 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(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)	Reserved(1)		Reserved(1)				0(1)	M1SM		M1UM			M0PE	M0SM		M0UM		
W																																
Reset (n=0)	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	1	1	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0
Reset (n>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	0

1. These bits should not be modified, even if they are writable by s / w. These bits have no effect on functionality.

Figure 94. SMPU Region Descriptor *n* – Word 2 Register (SMPU_RGDn_WORD2_FMT0)

Table 157. SMPU_RGDn_WORD2_FMT0 field descriptions

Field	Description
21:22 M1SM	Bus master 1 supervisor mode access control This 2-bit field defines the access controls for bus master 1 when operating in supervisor mode. The M1SM field is defined as: 00 r, w, x = read, write and execute allowed 01 r, –, x = read and execute allowed, but no write 10 r, w, – = read and write allowed, but no execute 11 Same access controls as that defined by M1UM for user mode
23:25 M1UM	Bus master 1 user mode access control This 3-bit field defines the access controls for bus master 1 when operating in user mode. The M1UM field consists of three independent bits, enabling read, write and execute permissions: {r,w,x}. If set, the bit allows the given access type to occur; if cleared, an attempted access of that mode may be terminated with an access error (if not allowed by any other descriptor) and the access not performed.

Table 157. SMPU_RGDn_WORD2_FMT0 field descriptions(Continued)

Field	Description
26 M0PE	Bus master 0 process identifier enable If set, this flag specifies that the process identifier and mask (defined in SMPU_RGDn_WORD3) are to be included in the region hit evaluation. If cleared, then the region hit evaluation does not include the process identifier.
27:28 M0SM	Bus master 0 supervisor mode access control This 2-bit field defines the access controls for bus master 0 when operating in supervisor mode. The M0SM field is defined as: 00 `r, w, x` = read, write and execute allowed 01 `r, -, x` = read and execute allowed, but no write 10 `r, w, -` = read and write allowed, but no execute 11 Same access controls as that defined by M0UM for user mode
29:31 M0UM	Bus master 0 user mode access control This 3-bit field defines the access controls for bus master 0 when operating in user mode. The M0UM field consists of three independent bits, enabling read, write and execute permissions: `{r,w,x}`. If set, the bit allows the given access type to occur; if cleared, an attempted access of that mode may be terminated with an access error (if not allowed by any other descriptor) and the access not performed.

Note: DMA works with master ID = 1. Nexus controller has master ID as 2.

17.6.4.4 SMPU Region Descriptor *n* – Word 3 (SMPU_RGDn_WORD3)

The fourth word of the SMPU region descriptor contains the optional process identifier and mask, plus the region descriptor's valid bit.

Since the region descriptor is a 4-word entity, there are potential coherency issues as this structure is being updated since multiple writes are required to update the entire descriptor. Accordingly, the SMPU hardware assists in the operation of the descriptor valid bit to prevent incoherent region descriptors from generating spurious access errors. In particular, it is expected that a complete update of a region descriptor is typically done with sequential writes to SMPU_RGDn_WORD0, then SMPU_RGDn_WORD1, then SMPU_RGDn_WORD2_FMT0 and finally SMPU_RGDn_WORD3. The SMPU hardware automatically clears the valid bit on any writes to words {0,1,2_FMT0} of the descriptor. Writes to this word set/clear the valid bit in a normal manner.

Since it is also expected that system software may adjust only the access controls within a region descriptor (SMPU_RGDn_WORD2_FMT0) as different tasks execute, an alternate programming view of SMPU_RGDn_WORD2_FMT0 is provided. If only the access controls are being updated, this operation should be performed by writing to SMPU_RGDAAC*n* (Alternate Access Control *n*) as stores to these locations do not affect the descriptor's valid bit.

Offset: 0x040C + *n**0x10 (*n*=0 to 7) 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	PID		PIDMASK		0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 VLD
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 95. SMPU Region Descriptor *n* – Word 3 Register (SMPU_RGDn_WORD3)

Table 158. SMPU_RGDn_WORD3 field descriptions

Field	Description
0:7 PID	Process Identifier This 8-bit field specifies that the optional process identifier is to be included in the determination of whether the current access hits in the region descriptor. This field is combined with the PIDMASK and included in the region hit determination if SMPU_RGDn_WORD2_FMT0[MxPE] is set.
8:15 PIDMASK	Process Identifier Mask This 8-bit field provides a masking capability so that multiple process identifiers can be included as part of the region hit determination. If a bit in the PIDMASK is set, then the corresponding bit of the PID is ignored in the comparison. This field is combined with the PID and included in the region hit determination if SMPU_RGDn_WORD2_FMT0[MxPE] is set. For more information on the handling of the PID and PIDMASK, see *Section 17.7.1.1*.
31 VLD	Valid. This bit signals the region descriptor is valid Any write to SMPU_RGDn_WORD{0,1,2_FMT0} clears this bit, while a write to SMPU_RGDn_WORD3 sets or clears this bit depending on bit 31 of the write operand. 0 Region descriptor is invalid 1 Region descriptor is valid

17.6.5 SMPU Region Descriptor Alternate Access Control *n* (SMPU_RGDAAC*n*)

As noted in *Section 17.6.4.3*, it is expected that since system software may adjust only the access controls within a region descriptor (SMPU_RGDn_WORD2_FMT0) as different tasks execute, *an alternate programming view of this 32-bit entity is desired*. If only the access controls are being updated, this operation should be performed by writing to SMPU_RGDAAC*n* (Alternate Access Control *n*) as stores to these locations do *not* affect the descriptor's valid bit.

The memory address therefore provides an alternate location for updating SMPU_RGDn_WORD2_FMT0.

Offset: 0x0800 + *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	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)	0(1)	0(1)	0(1)	0(1)	0(1)	0(1)	Reserved(1)	Reserved(1)	Reserved(1)				0(1)	M1SM	M1UM		M0PE	M0SM	M0UM					
W																																
Reset (n=0)	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	1	1	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0
Reset (n>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	0

1. These bits should not be written, even if they are writable by s / w. These bits have no effect on functionality.

Figure 96. SMPU RGD Alternate Access Control *n* (SMPU_RGDAAC*n*)

Since the SMPU_RGDAACn register is simply another memory mapping for SMPU_RGDn_WORD2_FMT0, the field definitions shown in *Table 159* are identical to those presented in *Table 157*.

Table 159. SMPU_RGDAAC*n* field descriptions

Field	Description
21:22 M1SM	Bus master 1 supervisor mode access control This 2-bit field defines the access controls for bus master 1 when operating in supervisor mode. The M1SM field is defined as: 00 `r, w, x` = read, write and execute allowed 01 `r, –, x` = read and execute allowed, but no write 10 `r, w, –` = read and write allowed, but no execute 11 Same access controls as that defined by M1UM for user mode
23:25 M1UM	Bus master 1 user mode access control This 3-bit field defines the access controls for bus master 1 when operating in user mode. The M1UM field consists of three independent bits, enabling read, write and execute permissions: `{r,w,x}`. If set, the bit allows the given access type to occur; if cleared, an attempted access of that mode may be terminated with an access error (if not allowed by any other descriptor) and the access not performed.
26 M0PE	Bus master 0 process identifier enable If set, this flag specifies that the process identifier and mask (defined in SMPU_RGDn_WORD3) are to be included in the region hit evaluation. If cleared, then the region hit evaluation does not include the process identifier.
27:28 M0SM	Bus master 0 supervisor mode access control This 2-bit field defines the access controls for bus master 0 when operating in supervisor mode. The M0SM field is defined as: 00 `r, w, x` = read, write and execute allowed 01 `r, –, x` = read and execute allowed, but no write 10 `r, w, –` = read and write allowed, but no execute 11 Same access controls as that defined by M0UM for user mode
29:31 M0UM	Bus master 0 user mode access control This 3-bit field defines the access controls for bus master 0 when operating in user mode. The M0UM field consists of three independent bits, enabling read, write and execute permissions: `{r,w,x}`. If set, the bit allows the given access type to occur; if cleared, an attempted access of that mode may be terminated with an access error (if not allowed by any other descriptor) and the access not performed.

Note: DMA works with master ID = 1. Nexus controller has master ID as 2.

17.7 Functional description

In this section, the functional operation of the SMPU is detailed. In particular, subsequent sections discuss the operation of the access evaluation macro as well as the handling of error-terminated AHB bus cycles.

17.7.1 Access evaluation macro

As previously discussed, 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 97*, the access evaluation macro inputs the AHB system bus address phase signals (AHB_ap) and the contents of a region descriptor (RGD*n*) and performs two major functions: region hit determination (*hit_b*) and detection of an access protection violation (*error*).

Figure 97. SMPU access evaluation macro

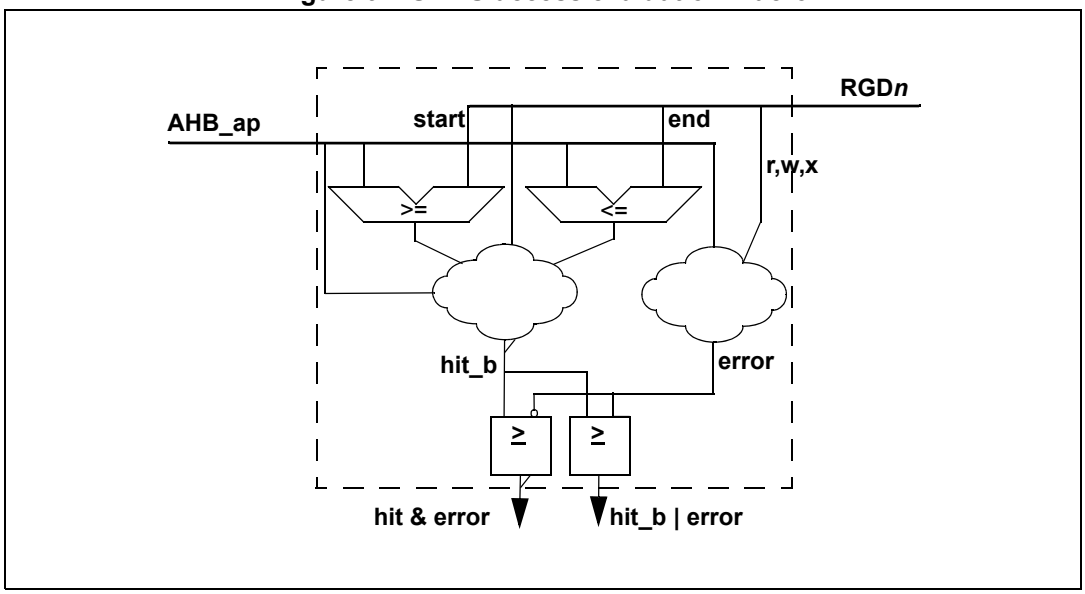


Figure 97 is not intended to be a schematic of the actual access evaluation macro, but rather a generalized block diagram showing the major functions included in this logic block.

17.7.1.1 Access evaluation: hit determination

17.7.1.1.1 Evaluation 1a

To determine if the current AHB reference hits in the given region, two magnitude comparators are used in conjunction with the region's start and end addresses. The boolean equation for this portion of the hit determination is defined as:

Equation 2 region_hit = ((haddr[0:26] >= rgdn.srtaddr[0:26]) & (haddr[0:26] <= rgdn.endaddr[0:26])) & rgdn.vld

where `haddr[*]` is the current AHB reference address, `rgdn.srtaddr[*]` and `rgdn.endaddr[*]` are the start and end addresses, and `rgdn.vld` is the valid bit, all from region descriptor n. Recall there are *no hardware checks* to verify that `rgdn.endaddr > rgdn.srtaddr`, and it is software's responsibility to properly load appropriate values into these fields of the region descriptor.

In addition to the algebraic comparison of the AHB reference address versus the region descriptor's start and end addresses, the optional process identifier is examined against the region descriptor's PID and PIDMASK fields. Using the `hmaster[*]` number to select the appropriate MxPE field from the region descriptor, a process identifier hit term is formed as:

Equation 3 pid_hit = ~rgdn.mxpe | ((current_pid[0:7] | rgdn.pidmask[0:7])== (rgdn.pid[0:7] | rgdn.pidmask[0:7]))

where the `current_pid[*]` is the selected process identifier from the current bus master, and `rgdn.pid[*]` and `rgdn.pidmask[*]` are the appropriate process identifier fields from the region descriptor n. For AHB bus masters that do *not* output a process identifier, the SMPU forces the `pid_hit` term to be asserted.

As shown in *Figure 97*, the access evaluation macro actually forms the logical complement (*hit_b*) of the combined region_hit and pid_hit boolean equations.

17.7.1.1.2 Evaluation 1b

Bye-wise Region Descriptors:

RGD hit occurs when Target_address >= RGD_start_address[0:31], AND Target_address+Offset(HBSTRB[3:0]) <= RGD_end_address[0:31]

Offset calculation rule:

- Offset = 0 when HBSTRB = 0001/0010/0100/1000
- Offset = 1 when HBSTRB = 0011/0110/1100
- Offset = 2 when HBSTRB = 0111/1110
- Offset = 3 when HBSTRB = 1111

HBSTRB[3:0] is the Data bus strobe signal as is driven by bus master on its AHB bus. (Should be used primarily by CPU as bus-master)

To help defining the RGD size, its necessary to understand the *Table 160* below:

Table 160. Byte Strobe Assertion for Transfers

Program size and byte offset	A(1:0)	HSIZE [1:0]	Data bus byte strobes(1)				HUNALIGN
			B0	B1	B2	B3	
Byte @ 00	0 0	0 0	X	—	—	—	0
Byte @ 01	0 1	0 0	—	X	—	—	0
Byte @ 10	1 0	0 0	—	—	X	—	0
Byte @ 11	1 1	0 0	—	—	—	X	0
Half @ 00	0 0	0 1	X	X	—	—	0
Half @ 01	0 1	1 0(2)	—	X	X	—	1
Half @ 10	1 0	0 1	—	—	X	X	0
Half @ 11 (2 bus transfers)	1 1 0 0	0 1(3) 0 0	— X	— —	— —	X —	1 0
Word @ 00	0 0	1 0	X	X	X	X	0
Word @ 01 (2 bus transfers)	0 1 0 0	1 0(3) 0 0	— X	X —	X —	X —	1 0
Word @ 10 (2 bus transfers)	10 00	1 0(3) 0 1	— X	— X	X —	X —	1 0
Word @ 11 (2 bus transfers)	1 1 0 0	1 0(3) 1 0(2)	— X	— X	— X	X —	1 1

1. X indicatyes byte lanes involved in the transfer: Other lanes will contain driven but unused data.

2. These misaligned tranfers drive size according to the size of the power of two aligned “container” in which the byte strobes are asserted.
3. These misaligned cases drive request size according to the size specified by the load or store instruction.

17.7.1.2 Access evaluation: privilege violation determination

While the access evaluation macro is making the region hit determination, the logic is also evaluating if the current access is allowed by the permissions defined in the region descriptor. Using the AHB `hmaster[*]` and `hprot[1]` (supervisor/user mode) signals, a set of effective permissions (`eff_rgd[r,w,x]`) is generated from the appropriate fields in the region descriptor. The protection violation logic then evaluates the access against the effective permissions using the specification shown in *Table 161*.

Table 161. Protection violation definition

Description	Inputs					Output
	hwrite	hprot[0]	eff_rgd[r]	eff_rgd[w]	eff_rgd[x]	Protection violation?
inst fetch read	0	0	—	—	0	yes, no x permission
inst fetch read	0	0	—	—	1	no, access is allowed
data read	0	1	0	—	—	yes, no r permission
data read	0	1	1	—	—	no, access is allowed
data write	1	—	—	0	—	yes, no w permission
data write	1	—	—	1	—	no, access is allowed

The resulting boolean equation for the processor protection violations is:

Equation 4 cpu_protection_violation = ~hwrite & ~hprot[0] & ~eff_rgdn[x]// ifetch & no x | ~hwrite & hprot[0] & ~eff_rgdn[r] // data_read & no r | hwrite & ~eff_rgdn[w] // data_write & no w

The resulting boolean equation for the non-processor protection violations is:

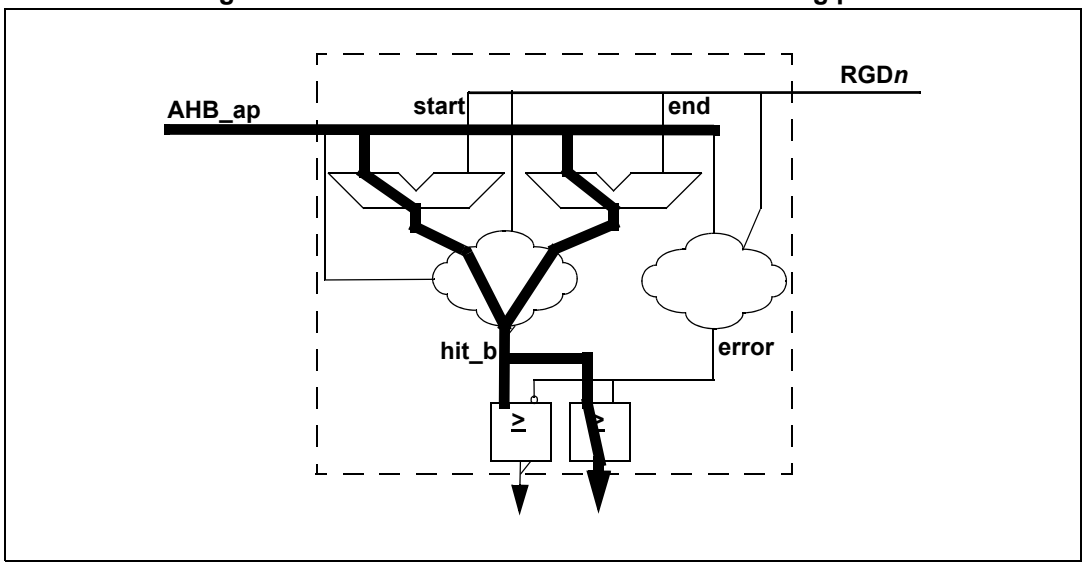
Equation 5 protection_violation = ~hwrite & ~eff_rgdn[r]// data_read & no r hwrite& ~eff_rgdn[w] // data_write& no w

As shown in *Figure 97*, the output of the protection violation logic is the *error* signal, that is, *error* = protection_violation.

The access evaluation macro then uses the *hit_b* and *error* signals to form two outputs. The combined *(hit_b | error)* signal is used to signal the current access is not allowed and *(~hit_b & error)* is used as the input to SMPU_EDR*n* (error detail register) in the event of an error.

The critical timing arc through the access evaluation macro involves the delay from the arrival of `haddr[*]` through the two magnitude comparators and through the *hit_b* generation as shown in *Figure 98*.

Figure 98. Access evaluation macro critical timing path



17.7.2 Putting it all together and AHB error terminations

For each AHB slave port being monitored, the SMPU performs a **reduction-AND** of all the individual *(hit_b | error)* terms from each access evaluation macro. This expression then terminates the bus cycle with an error and reports a protection error for three conditions:

1. If the access does not hit in *any* region descriptor, a protection error is reported
2. If the access hits in a single region descriptor and that region signals a protection violation, then a protection error is reported
3. If the access hits in multiple (overlapping) regions and *all* regions signal protection violations, then a protection error is reported

The third condition reflects that priority is given to *permission granting over access denying* for overlapping regions as this approach provides more flexibility to system software in region descriptor assignments. For an example of the use of overlapping region descriptors, see *Section 17.10*.

The handling of the AHB bus cycle with a protection error requires two distinct actions. First, recall the protection error logic resides within the AHB address phase pipeline stage. In the event of a protection error, the reference must be inhibited from entering the AHB data phase for the targeted slave device. Stated differently, the reference's address phase must be aborted as viewed by the targeted slave device. This is accomplished by dynamically revising the `htrans[*]` signal sent to the slave device. The `htrans[*]` attribute signals an idle or valid reference (as a non-sequential or sequential access), and this signal is "intercepted" by the SMPU. If the access is allowed, then the SMPU simply passes the original `htrans[*]` value to the slave device. However if a protection error is detected, then the SMPU forces `htrans[*]` sent to the slave to the IDLE encoding. This effectively cancels the transaction before it is seen by the slave device.

While forcing `htrans[*] = IDLE` effectively prevents the bus cycle from being transmitted to the slave device, the AHB transaction has already been "committed" in other portions of the platform, namely in the initiating bus master and the crossbar switch. To properly terminate the bus cycle for these modules, it is necessary to post the standard 2-cycle AHB error response. For this response, the `hresp[*]` signal is forced to the ERROR encoding

for two cycles, the first with `hready` negated and the second with `hready` asserted. To perform these termination functions, the SMPU “intercepts” the `hready` and `hresp[*]` signals from the slave devices, performs the required logic functions to force the response on the bus cycle with a protection error, and then outputs the adjusted values to the crossbar switch.

17.8 Functional safety

SMPU provides 2 alarms for functional safety purposes:

1. Protection violation
   SMPU generates this alarm whenever there is a protection error (as explained in previous *Section 17.7.2*) found on any access based on SMPU programming.
2. Monitor error
   a) SMPU faults are covered by a protocol checker. In order to protect from valid transactions affected due to SMPU internal failures, SMPU supervision is implemented based on a transaction monitor which checks that input and output transactions are consistent when no access violation is found.
   b) In the event of consistency mismatch between the input and output transactions across the SMPU, an RCCU generates error output to FCCU.

17.9 Initialization information

The reset state of SMPU_CESR[VLD] disables the entire module. Recall while the SMPU is disabled, all accesses from all bus masters are allowed. This state also minimizes the power dissipation of the SMPU. The power dissipation of each access evaluation macro is minimized when the associated region descriptor is marked as invalid or when SMPU_CESR[VLD] = 0.

Typically the appropriate number of region descriptors (SMPU_RGD*n*) are loaded at system startup, including the setting of the SMPU_RGDn_WORD3[VLD] bits, *before* SMPU_CESR[VLD] is set, enabling the module. This approach allows all the loaded region descriptors to be enabled simultaneously. Recall if a memory reference does not hit in *any* region descriptor, the attempted access is terminated with an error.

17.10 Application information

In an operational system, interfacing with the SMPU can generally be classified into the following activities:

1. Creation of a new memory region requires loading the appropriate region descriptor into an available register location. When a new descriptor is loaded into a RGD*n*, it would typically be performed using four 32-bit word writes. As discussed in *Section 17.6.4.4*, the hardware assists in the maintenance of the valid bit, so if this approach is followed, there are no coherency issues associated with the multi-cycle descriptor writes. Deletion/removal of an existing memory region is performed simply by clearing SMPU_RGDn_WORD3[VLD].
2. If only the access rights for an existing region descriptor need to change, a 32-bit write to the alternate version of the access control word (SMPU_RGDAACn) would typically be performed. Recall writes to the region descriptor using this alternate access control

location do not affect the valid bit, so there are, by definition, no coherency issues involved with the update. The access rights associated with the memory region switch instantaneously to the new value as the IPS write completes.

3. If the region's start and end addresses are to be changed, this would typically be performed by writing a minimum of three words of the region descriptor: SMPU_RGDn_WORD{0,1,3}, where the writes to Word0 and Word1 redefine the start and end addresses respectively and the write to Word3 re-enables the region descriptor valid bit. In many situations, all four words of the region descriptor would be rewritten.
4. Typically, references to the SMPU's programming model would be restricted to supervisor mode accesses from a specific processor(s), so a region descriptor would be specifically allocated for this purpose with attempted accesses from other masters or while in user mode terminated with an error.
5. When the SMPU detects an access error, the current AHB bus cycle is terminated with an error response and information on the faulting reference captured in the SMPU_EAR*n* and SMPU_EDR*n* registers. The error-terminated AHB bus cycle typically initiates some type of 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. In any event, the processor can retrieve the captured error address and detail information simply be reading the SMPU_E{A,D}R*n* registers. Information on which error registers contain captured fault data is signaled by SMPU_CESR[SPERR].
6. Finally, consider the use of overlapping region descriptors. Application of overlapping regions can often reduce the number of descriptors required for a given set of access controls. It is important to note that, in the overlapping memory space, the protection rights of the corresponding region descriptors are logically summed together (the boolean OR operator). In the following example of a dual-core system, there are 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 region descriptor assignments in *Table 162*. In this example, there are 8 descriptors used to span 9 regions in the three main spaces of the system memory map (flash, RAM and IPS peripheral space). Each region indicates the specific permissions for each of the four bus masters and this definition provides an appropriate 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 IPS 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 System Memory Protection Unit in a typical system.

Table 162. Overlapping region descriptor example

Region description	RGD*n*	CP0	CP1	DMA1	DMA2	System space
CP0 code	0	rwx	r--	--	--	Flash memory
CP1 code	1	r--	rwx	--	--	
CP0 data and stack	2	rw-	---	--	--	SRAM
CP0 → CP1 shared data	3	r--	r--	--	--	
CP1 → CP0 shared data						
CP1 data and stack	4	---	rw-	--	--	
Shared DMA data	5	rw-	rw-	rw	rw	
SMPU	6	rw-	rw-	--	--	IPS
Peripherals	7	rw-	rw-	rw	--	

18 Interrupt Controller (INTC)

18.1 Introduction

Note: For the chip-specific implementation details of this module's instances, please refer to the Section 7.3.5: Interrupt controller (INTC) configuration.

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 32 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.

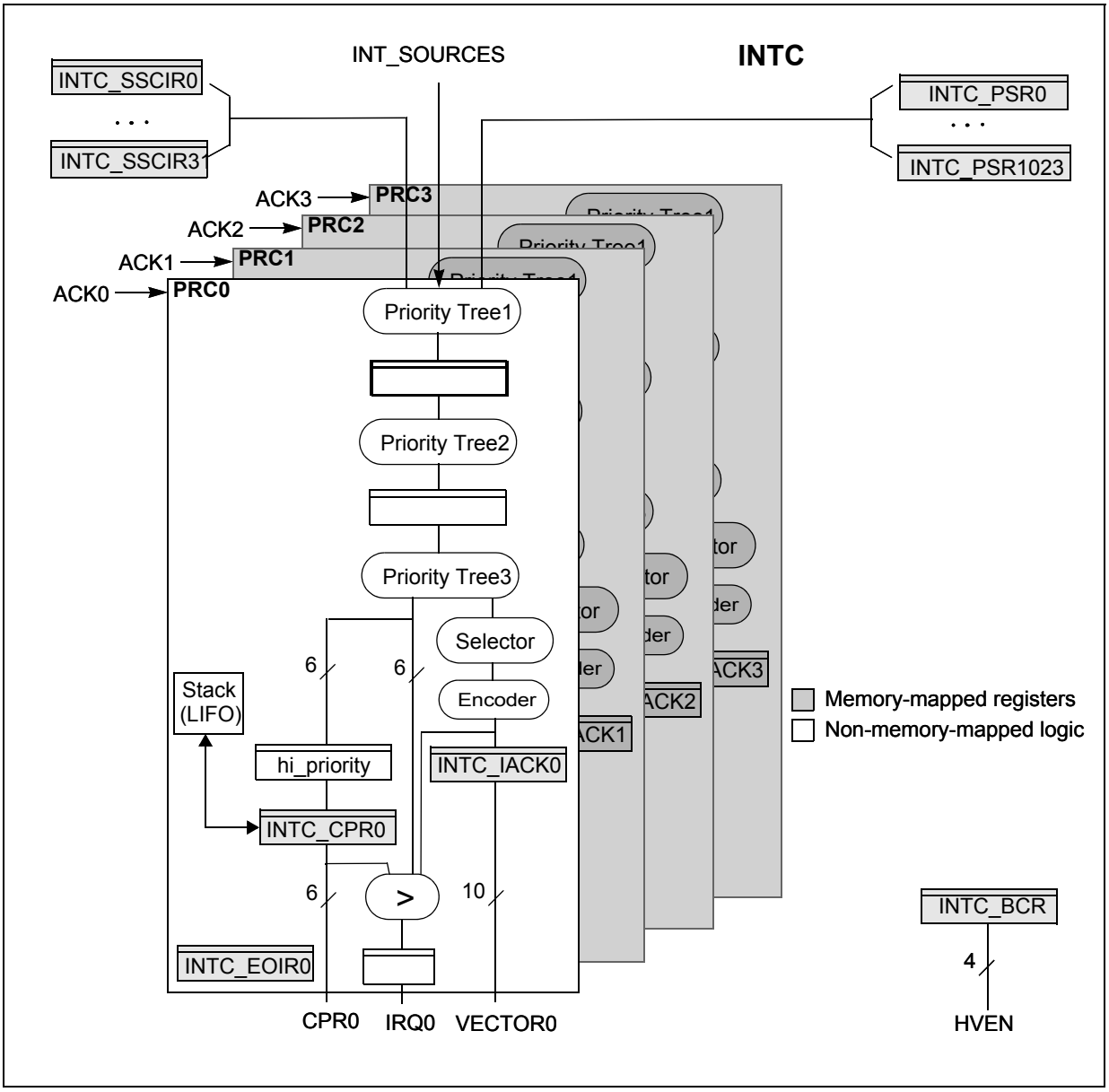
When multiple tasks share a resource, coherent accesses to that resource need to be supported. The INTC supports the Priority Ceiling Protocol for coherent accesses. By providing a modifiable priority mask, the priority can be raised temporarily so that all tasks which share the resource are unable to 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.

18.2 Block diagram

The block diagram for the INTC is shown in *Figure 99*.

Figure 99. INTC block diagram



18.3 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 32 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

18.4 Modes of operation

The interrupt controller operates with the following handshaking modes: software vector mode. 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).

Note: The value of n = 0 in this chapter, since this device contains only one core.

18.4.1 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 a given processor when its associated HVENn bit in the INTC_BCR (see *Section 18.5.2.1*) is negated. The hardware vector enable signal to the processor is driven as negated when its associated HVEN*n* bit is negated. The vector is read from the INTC_IACKR0 register (see *Section 18.5.2.3*). Reading the INTC_IACKR0 negates the interrupt request to the associated 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 the INTC_CPR0 (see *Section 18.5.2.2*) onto the associated LIFO, and updates PRI in the associated INTC_CPR_PRC*n* with the new priority.

18.5 Memory map and register definition

18.5.1 Memory map

Table 163 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 163. INTC memory map

Address offset	Register	Location
0x0000	INTC Block Configuration Register (INTC_BCR)	*Section 18.5.2.1*
0x0004–0x0007	Reserved(1)	
0x0008–0x000F	Reserved	
0x0010	INTC Current Priority Register for Processor 0 (INTC_CPR0)	*Section 18.5.2.2*
0x0020	INTC Interrupt Acknowledge Register for Processor 0 (INTC_IACKR0)	*Section 18.5.2.3*
0x0030	INTC End Of Interrupt Register for Processor 0 (INTC_EOIR0)	*Section 18.5.2.4*
0x0040–0x005F	INTC Software Set/Clear Interrupt Register 0–31 (INTC_SSCIR0–INTC_SSCIR31)	*Section 18.5.2.5*
0x0060–0x085F	INTC Priority Select Registers 0–1023 (INTC_PSR0–INTC_PSR1023)	*Section 18.5.2.6*
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.

18.5.2 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 vector mode, the size of a write to the INTC end-of-interrupt register (INTC_EOIR0) does not affect the operation of the write.

18.5.2.1 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		0	0	0		0	0	0		0	0	0	HVEN0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 100. INTC Block Configuration Register (INTC_BCR)

Table 164. 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 18.4*, for the details of the handshaking with the processor in each mode. 0 Software vector mode 1 Hardware vector mode **Note:** This bit can be written and read, but should not be written by software as Hardware Vector Mode is not supported for this device.

18.5.2.2 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 , 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.

The masking priority can be raised or lowered by writing to the PRI field, supporting the priority ceiling protocol. See *Section 18.7.5*.

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 18.7.5.2, 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	0	PRI				
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1

Figure 101. INTC Current Priority Register for Processor 0 (INTC_CPR0)

Table 165. INTC_CPR0 field descriptions

Field	Description
27:31 PRI	Priority of the currently executing ISR, according to the field values 31 (highest priority) down to 0 (lowest priority).

18.5.2.3 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.

Offset: 0x0020 (INTC_IACKR0) Access: User write always; non-speculative read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	VTBA															
W																
Reset	0	0	0	0	0	0	0	0	0	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

Figure 102. INTC Interrupt Acknowledge Register for Processor 0 (INTC_IACKR0)

Table 166. 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.

18.5.2.4 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 103. INTC End of Interrupt Register for Processor 0 (INTC_EOIR0)

Table 167. 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.

18.5.2.5 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 104 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 104. INTC Software Set/Clear Interrupt Register *n* (INTC_SSCIR*n*)

Table 168. 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.

18.5.2.6 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 105 and *Figure 106* 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

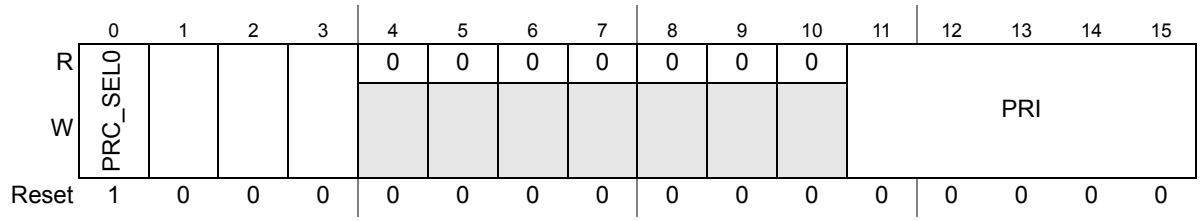


Figure 105. INTC Priority Select Register *n* (INTC_PSR*n*)

Table 169. 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.
11:15 PRI	Priority select Selects the priority for the interrupt requests. PRI has values 31 (highest priority) down to 0 (lowest priority).

Offset: 0x0060 + *n* * 0x2 (*n*=32 to 1023) Access: User read/write

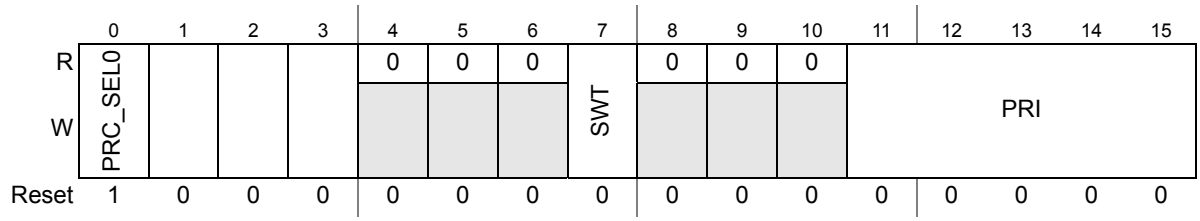


Figure 106. INTC Priority Select Register *n* (INTC_PSR*n*)

Table 170. 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.
11:15 PRI	Priority select Selects the priority for the interrupt requests. PRI has values 31 (highest priority) down to 0 (lowest priority).

18.6 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.

18.6.1 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*n* 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 the associated INTC_CPR0 is updated with the corresponding PRI*n* 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.

18.6.1.1 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.

18.6.1.2 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.

Any Processor n 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 n which has been assigned to handle that interrupt request, as determined by the setting of the INTC_PSR*n*[PRC_SEL*n*] bits.

18.6.1.3 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.

18.6.2 Priority management

The asserted interrupt requests are compared to each other based on their PRI*n* and PRC_SEL*n* values set in INTC_PSR*n*. The result of that comparison also is compared to PRI in the associated INTC_CPR0. The results of those comparisons are used to manage the priority of the ISR being executed by the associated processor. The associated LIFO also assists in managing that priority.

18.6.2.1 Current priority and preemption

The priority arbitrator, selector, encoder, and comparator logic shown in *Figure 99* 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 a given 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 the associated INTC_IACKR0.

18.6.2.1.1 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 associated selector logic.

18.6.2.1.2 Selector

If only one interrupt request from the associated priority tree is asserted, then it is passed as asserted to the associated encoder logic. If multiple interrupt requests from the associated priority tree are asserted, then only the one with the lowest vector is passed as asserted to the associated encoder logic. The lower vector is chosen regardless of the time order of the assertions of the peripheral or software-settable interrupt requests.

18.6.2.1.3 Encoder

The encoder logic generates the unique 10-bit vector for the asserted interrupt request from the request selector subblock for the associated processor.

18.6.2.1.4 Comparator

The comparator logic compares the highest priority output from the associated priority arbitrator subblock with PRI in the associated 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 the associated INTC_CPR0, or the PRI value in the associated INTC_CPR0 is lowered below this highest priority. This highest priority then becomes the new priority which is written to PRI in the associated INTC_CPR0 when the interrupt request to the processor is acknowledged. Interrupt requests whose PRI*n* in INTC_PSR*n* are zero will not cause a preemption because their PRI*n* will not be higher than PRI in the associated INTC_CPR0.

Another function of the comparator is to signal an update of the 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 the 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.

18.6.2.2 Stack (LIFO)

The LIFO stores the preempted PRI values from the associated 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 the associated INTC_CPR0 does not need to be loaded from the associated 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 into the associated INTC_CPR0.

The PRI value in the associated INTC_CPR0 is pushed onto the LIFO when the associated INTC_IACKR0 is read in software vector mode . The priority is popped into PRI in the associated INTC_CPR0 whenever the associated INTC_EOIR0 is written.

Although the INTC supports up to 32 priorities, an ISR executing with PRI in the INTC_CPR0 equal to 31 will not be preempted. Therefore, the LIFO supports the stacking of 31 priorities. However, the LIFO is only 30 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 31 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.

18.6.3 Handshaking with processor

18.6.3.1 Software vector mode handshaking

18.6.3.1.1 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 107*.

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 107*. The INTC examines the peripheral and software-settable interrupt requests. When it finds an asserted peripheral or software-settable interrupt request with a higher

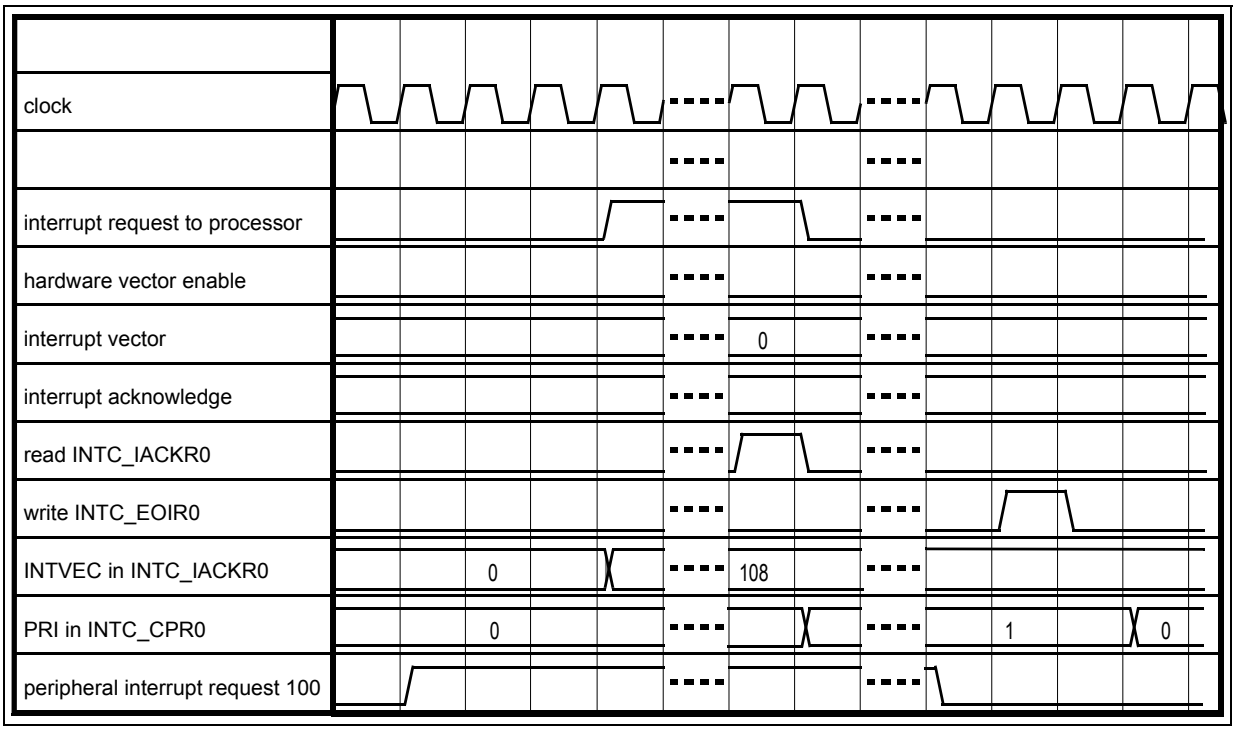
priority than PRI in the associated INTC_CPR0, it asserts the interrupt request to the processor. The INTVEC field in the INTC_IACKR0 is updated with the preempting interrupt request's vector when the interrupt request to the processor is asserted. The INTVEC field retains that value until the next time the interrupt request to the processor is asserted. The rest of the handshaking is described in *Section 18.4.1*.

18.6.3.1.2 End of interrupt exception handler

Before the interrupt exception handling completes, INTC_EOIR0 must be written. When it is written, the associated LIFO is popped so that the preempted priority is restored into PRI of the associated 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 the associated 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 107. Timing diagram of software vector mode handshaking



18.7 Initialization/application information

18.7.1 Initialization flow

After exiting reset, all of the PRI*n* and PRC_SEL*n* fields in INTC_PSR*n* are zero, and PRI in the INTC_CPR0 registers 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*n* fields and set the PRC_SEL*n* fields to the desired processor 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.

18.7.2 Interrupt exception handler

These example interrupt exception handlers excerpts use Power Architecture assembly code.

18.7.2.1 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
```

18.7.3 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*n* 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.

18.7.4 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 171* shows the order of execution of both cases, ISRs with different priorities and ISRs with the same priority.

Table 171. 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

Table 171. 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**
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
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.

18.7.5 Priority ceiling protocol

18.7.5.1 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_CPR0, 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.

18.7.5.2 Ensuring coherency

18.7.5.2.1 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:

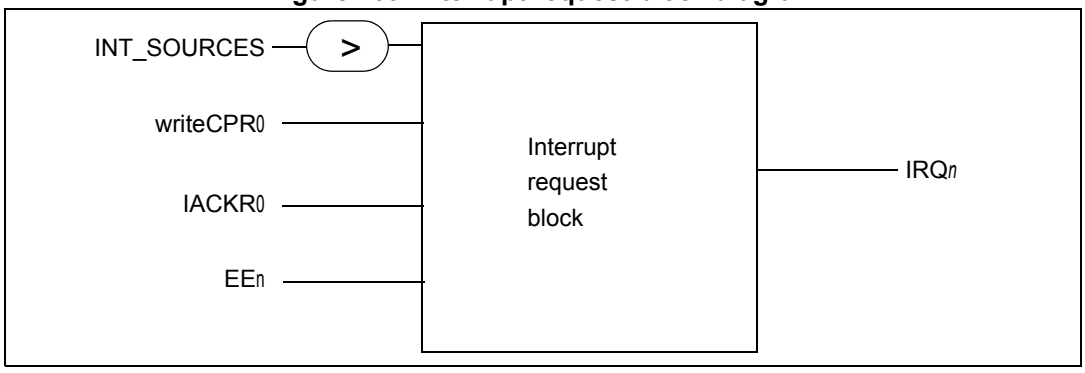
Figure 108. OSEK Code Sequence

```
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
```

18.7.5.2.1.1 Interrupt request to processor

Referencing *Figure 99*, the interrupt request logic to the processor is shown in *Figure 109*.

Figure 109. Interrupt request block diagram



Assuming that there are no IRQ*n* deasserted (no pending interrupt requests to processor *n*), if one of the external interrupt sources (INT_SOURCES) has a priority higher than the 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 108*), IRQ*n* will be re-evaluated when processor *n* writes to the INTC_CPR0 while processor *n* ignores IRQ*n*

This provides a safe way to guarantee that no interrupts will be recognized by processor n while updating the INTC_CPR0.

18.7.5.2.2 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 110* shows the timing diagram for this scenario, and *Table 172* explains the events. The example is for software vector mode.

Figure 110. Timing diagram of raised priority preserved

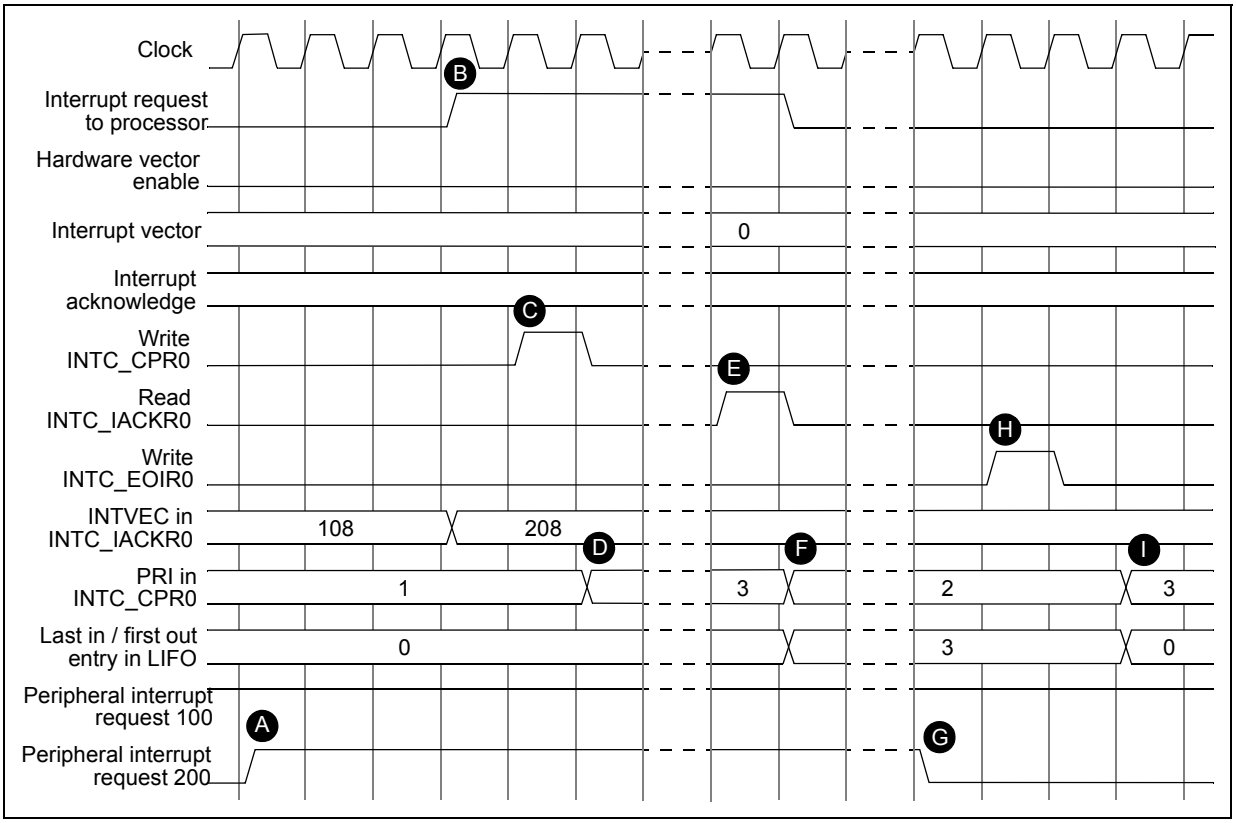


Table 172. Raised priority preserved events

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.
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.

18.7.6 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 32 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.

18.7.7 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.

18.7.7.1 Scheduling a lower priority portion of an ISR

A portion of an ISR needs to be executed at the PRI*n* 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*n* value in the INTC_PSR*n*, therefore will not cause preemptive scheduling inefficiencies.

18.7.8 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 18.7.7.1*) 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.

18.7.9 Negating an interrupt request outside of its ISR

18.7.9.1 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.

18.7.9.2 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.

18.7.9.3 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*n* 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 18.6.3.1.2* 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*n* value in INTC_PSR*n*.

18.7.10 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.

18.8 Interrupt sources

The list of interrupt sources is chip-specific. For this list, see the chapter that describes how modules are configured and connected.

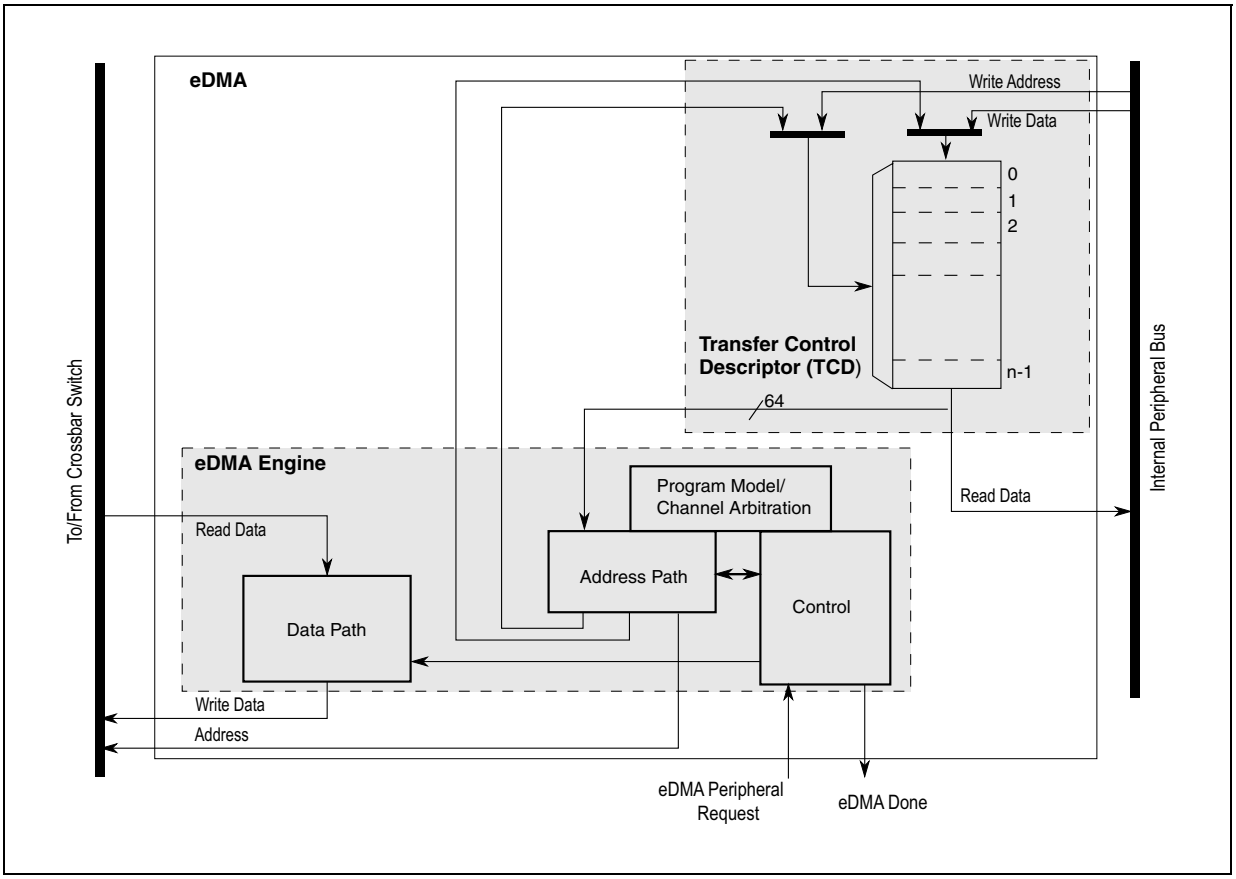
19 Enhanced direct memory access (eDMA)

19.1 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 16 channels.

Figure 111 illustrates the eDMA module.

Figure 111. eDMA block diagram



19.1.1 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
- 16-channel implementation that performs complex data transfers with minimal intervention from a host processor
  - Internal data buffer, used as temporary storage to support 16-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.

19.2 Modes of operation

The eDMA supports the following modes of operation.

19.2.1 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.

19.2.2 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.

19.2.3 Wait mode

Before entering Wait mode the DMA attempts to complete its current transfer. After the transfer completes the device enters Wait mode.

19.3 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.

Table 173 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 15. Each TCD*n* definition is presented as eleven 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 173. DMA memory map

Address offset	Register name	Width (in bits)	Location
0x0000	Control Register (DMA_CR)	32	*Section 19.3.1*
0x0004	Error Status Register (DMA_ES)	32	*Section 19.3.2*
0x0008–0x000B	Reserved		
0x000C	Enable Request Register (DMA_ERQL)	32	*Section 19.3.3*
0x0010–0x0013	Reserved		
0x0014	Enable Error Interrupt Register (DMA_EEIL)	32	*Section 19.3.4*
0x0018	Set Enable Request Register (DMA_SERQ)	8	*Section 19.3.5*
0x0019	Clear Enable Request Register (DMA_CERQ)	8	*Section 19.3.6*
0x001A	Set Enable Error Interrupt Register (DMA_SEEI)	8	*Section 19.3.7*
0x001B	Clear Enable Error Interrupt Register (DMA_CEEI)	8	*Section 19.3.8*
0x001C	Clear Interrupt Request Register (DMA_CINT)	8	*Section 19.3.9*
0x001D	Clear Error Register (DMA_CERR)	8	*Section 19.3.10*
0x001E	Set START Bit Register (DMA_SSRT)	8	*Section 19.3.11*
0x001F	Clear DONE Status Bit Register (DMA_CDNE)	8	*Section 19.3.12*

Table 173. DMA memory map(Continued)

Address offset	Register name	Width (in bits)	Location
0x0020–0x0023	Reserved		
0x0024	Interrupt Request Register (DMA_INTL)	32	*Section 19.3.13*
0x0028–0x002B	Reserved		
0x002C	Error Register (DMA_ERRL)	32	*Section 19.3.14*
0x0030–0x0033	Reserved		
0x0034	Hardware Request Status Register (DMA_HRSL)	32	*Section 19.3.15*
(n = 0 to 15)			
0x1*n+0x0100	Channel n Priority Register (DMA_DCHPRIn)	8	*Section 19.3.16*
0x0110–0x013F	Reserved		
0x1*n+0x0140	Channel n Master ID Register (DMA_DCHMIDn)	8	*Section 19.3.17*
0x0150–0x0FFF	Reserved		
0x20*n+0x1000	TCDn Source Address (DMA_TCDn_SADDR)	32	*Section 19.3.18*
0x20*n+0x1004	TCDn Transfer Attributes (DMA_TCDn_ATTR)	16	*Section 19.3.19*
0x20*n+0x1006	TCDn Signed Source Address Offset (DMA_TCDn_SOFF)	16	*Section 19.3.20*
0x20*n+0x1008	TCDn Minor Byte Count (Minor Loop Disabled) (DMA_TCDn_NBYTES_MLNO)	32	*Section 19.3.21*
	TCDn Signed Minor Loop Offset (Minor Loop Enabled and Offset Disabled) (DMA_TCDn_NBYTES_MLOFFNO)	32	*Section 19.3.22*
	TCDn Signed Minor Loop Offset (Minor Loop and Offset Enabled) (DMA_TCDn_NBYTES_MLOFFYES)	32	*Section 19.3.23*
0x20*n+0x100C	TCD Last Source Address Adjustment (DMA_TCDn_SLAST)	32	*Section 19.3.24*
0x20*n+0x1010	TCD Destination Address (DMA_TCDn_DADDR)	32	*Section 19.3.25*
0x20*n+0x1014	TCDn Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCDn_CITER_ELINKYES)	16	*Section 19.3.26*
	TCDn Current Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCDn_CITER_ELINKNO)	16	*Section 19.3.27*
0x20*n+0x1016	TCD Signed Destination Address Offset (DMA_TCDn_DOFF)	16	*Section 19.3.28*
0x20*n+0x1018	TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCDn_DLASTSGA)	32	*Section 19.3.29*

Table 173. DMA memory map(Continued)

Address offset	Register name	Width (in bits)	Location
0x20*n+0x101C	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCDn_BITER_ELINKYES)	16	*Section 19.3.30*
	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCDn_BITER_ELINKNO)	16	*Section 19.3.31*
0x20*n+0x101E	TCD Control and Status (DMA_TCDn_CSR)	16	*Section 19.3.32*
0x1200–0x17FF	Reserved		

19.3.1 Control Register (DMA_CR)

The CR defines the basic operating configuration of the DMA.

The DMA arbitrates channel service requests for the 16 channels.

Arbitration 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 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).

Offset: 0x0000 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	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						GRP0PRI		EMLM	CLM	HALT	HOE	ERGA	ERCA	EDBG	EBW
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 112. Control Register (DMA_CR)

Table 174. DMA_CR field descriptions

Field	Description
0:13 Reserved	This read-only field is reserved and always has the value zero.
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.
16:21 Reserved	This read-only field is reserved and always has the value zero.
22:23 GRP0PRI	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 being 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.

Table 174. DMA_CR field descriptions(Continued)

Field	Description
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.
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.

19.3.2 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 19.4.3*, for more details.

Offset: 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	0	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 113. Error Status Register (DMA_ES)

Table 175. 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
1:13 Reserved	This read-only bitfield is reserved and always has the value zero.
14 UCE	Uncorrectable ECC error 0 No uncorrectable ECC error 1 The last recorded error was an uncorrectable TCD RAM error

Table 175. DMA_ES field descriptions(Continued)

Field	Description
15 ECX	Transfer cancelled 0 No cancelled transfers 1 The last recorded entry was a cancelled transfer by the error cancel transfer input
16 Reserved	This read-only bitfield is reserved and always has the value zero.
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

19.3.3 Enable Request Register (DMA_ERQL)

The ERQL registers provide a bit map for the 16 implemented channels to enable the request signal for each channel. EQRL covers channels 15–00.

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.

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	ERQ15	ERQ14	ERQ13	ERQ12	ERQ11	ERQ10	ERQ9	ERQ8	ERQ7	ERQ6	ERQ5	ERQ4	ERQ3	ERQ2	ERQ1	ERQ0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 114. Enable Request Register (DMA_ERQL)

Table 176. DMA_ERQL field descriptions

Field	Description
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
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

Table 176. DMA_ERQL field descriptions(Continued)

Field	Description
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

19.3.4 Enable Error Interrupt Register (DMA_EEIL)

The Enable Error Interrupt Register (DMA_EEIL) provides a bit map for the 16 channels to enable the error interrupt signal for each channel. EEIL covers channels 15–00. 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.

Offset: 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	EEI15	EEI14	EEI13	EEI12	EEI11	EEI10	EEI9	EEI8	EEI7	EEI6	EEI5	EEI4	EEI3	EEI2	EEI1	EEI0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 115. Enable Error Interrupt Register (DMA_EEIL)

Table 177. DMA_EEIL field descriptions

Field	Description
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
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

Table 177. DMA_EEIL field descriptions(Continued)

Field	Description
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
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

19.3.5 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 zeroes.

Offset: 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 116. Set Enable Request Register (DMA_SERQ)

Table 178. DMA_SERQ field descriptions

Field	Description
0 NOP	0 Normal operation 1 No operation, ignore bits 1 to 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

19.3.6 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 zeroes.

Offset: 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 117. Clear Enable Request Register (DMA_CERQ)

Table 179. DMA_CERQ field descriptions

Field	Description
0 NOP	0 Normal operation 1 No operation, ignore bits 1 to 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

19.3.7 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 zeroes.

Offset: 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 118. Set Enable Error Interrupt Register (DMA_SEEI)

Table 180. DMA_SEEI field descriptions

Field	Description
0 NOP	0 Normal operation 1 No operation, ignore bits 1 to 7 of this register
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

19.3.8 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 zeroes.

Offset: 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 119. Clear Enable Error Interrupt Register (DMA_CEEI)

Table 181. DMA_CEEI field descriptions

Field	Description
0 NOP	0 Normal operation 1 No operation, ignore bits 1 to 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

19.3.9 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 zeroes.

Offset: 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 120. Clear Interrupt Request Register (DMA_CINT)

Table 182. DMA_CINT field descriptions

Field	Description
0 NOP	0 Normal operation 1 No operation, ignore bits 1 to 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

19.3.10 Clear Error Register (DMA_CERR)

The CERR provides a simple memory-mapped mechanism to clear a given bit in the ERR{H,L} to disable the error condition flag for a given channel. The given value on a register write causes the corresponding bit in the ERR{H,L} to be cleared. Setting the CAEI bit provides a global clear function, forcing the ERR{H,L} 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 zeroes.

Offset: 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 121. Clear Error Register (DMA_CERR)

Table 183. DMA_CERR field descriptions

Field	Description
0 NOP	0 Normal operation 1 No operation, ignore bits 1 to 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

19.3.11 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 zeroes.

Offset: 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 122. Set START Bit Register (DMA_SSRT)

Table 184. DMA_SSRT field descriptions

Field	Description
0 NOP	0 Normal operation 1 No operation, ignore bits 1 to 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]

19.3.12 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 zeroes.

Offset: 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 123. Clear DONE Status Bit Register (DMA_CDNE)

Table 185. DMA_CDNE field descriptions

Field	Description
0 NOP	0 Normal operation 1 No operation, ignore bits 1 to 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]

19.3.13 Interrupt Request Register (DMA_INTL)

The Interrupt Request Register (DMA_INTL) provides a bit map for the 16 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.

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	INT15	INT14	INT13	INT12	INT11	INT10	INT9	INT8	INT7	INT6	INT5	INT4	INT3	INT2	INT1	INT0
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 124. Interrupt Request Register (DMA_INTL)

Table 186. DMA_INTL field descriptions

Field	Description
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

Table 186. DMA_INTL field descriptions(Continued)

Field	Description
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
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

19.3.14 Error Register (DMA_ERRL)

The ERRL provides a bit map for the 16 channels, signaling the presence of an error for each channel. ERRL covers channels 15–00. 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 to form a group of 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.

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	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	ERR15	ERR14	ERR13	ERR12	ERR11	ERR10	ERR9	ERR8	ERR7	ERR6	ERR5	ERR4	ERR3	ERR2	ERR1	ERR0
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 125. Error Register (DMA_ERRL)

Table 187. DMA_ERRL field descriptions

Field	Description
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

Table 187. DMA_ERRL field descriptions(Continued)

Field	Description
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
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

19.3.15 Hardware Request Status Register (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 15–00. 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.

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	HRS15	HRS14	HRS13	HRS12	HRS11	HRS10	HRS9	HRS8	HRS7	HRS6	HRS5	HRS4	HRS3	HRS2	HRS1	HRS0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 126. Hardware Request Status Register (DMA_HRSL)

Table 188. DMA_HRSL field descriptions

Field	Description
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
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

Table 188. DMA_HRSL field descriptions(Continued)

Field	Description
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

19.3.16 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.

Offset: 0x0100 + n * 0x1 (n = 0 to 15) Access: User read/write

	0	1	2	3	4	5	6	7
R	ECP	DPA	0	0	CHPRI			
W								
Reset(1)	0	0	x	x	x	x	x	x

1. x = Undefined at reset

Figure 127. Channel *n* Priority Register (DMA_DCHPRI*n*)

Table 189. 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
4:7 CHPRI	Channel *n* arbitration priority Channel priority when fixed-priority arbitration is enabled **Note:** Reset value is equal to the corresponding channel number for each priority register, that is DCHPRI31[CHPRI] equals 0b1111.

19.3.17 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 (a core, for example) 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.

Offset: 0x0140 + n * 0x1 (n = 0 to 15) Access: User read/write

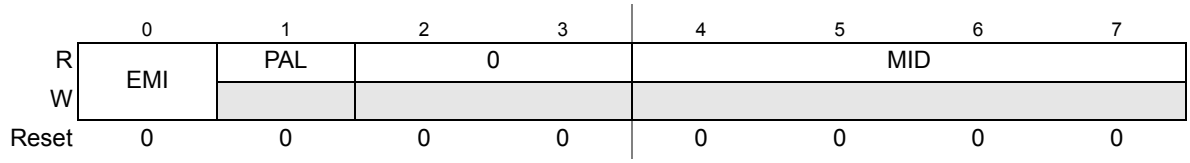


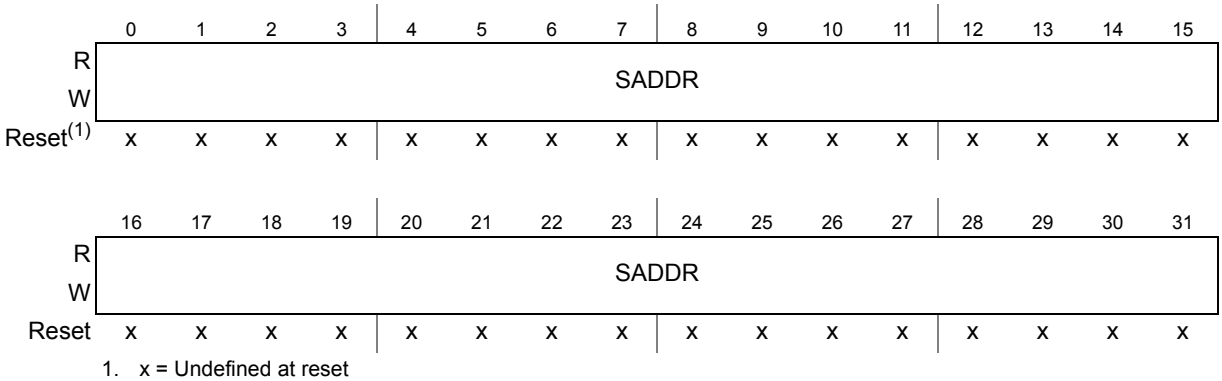
Figure 128. Channel *n* Master ID Register (DMA_DCHMID*n*)

Table 190. 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
2:3 Reserved	This read-only bitfield is reserved and always has the value zero.
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.

19.3.18 TCD Source Address (DMA_TCD*n*_SADDR)

Offset: 0x1000 + n * 0x20 (n = 0 to 15) Access: User read/write



1. x = Undefined at reset

Figure 129. TCD Source Address (DMA_TCD*n*_SADDR)

Table 191. DMA_TCD*n*_SADDR field descriptions

Field	Description
0:31 SADDR	Source address Memory address pointing to the source data.

19.3.19 TCD Transfer Attributes (DMA_TCD*n*_ATTR)

Offset: 0x1004 + n * 0x20 (n = 0 to 15) 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 130. TCD Transfer Attributes (DMA_TCD*n*_ATTR)

Table 192. DMA_TCD*n*_ATTR field descriptions

Field	Description
0:4 SMOD	Source address modulo. 0 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 Reserved 100 16-byte 101 Reserved 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

19.3.20 TCD Signed Source Address Offset (DMA_TCD*n*_SOFF)

Offset: 0x1006 + n * 0x20 (n = 0 to 15) 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 131. TCD Signed Source Address Offset (DMA_TCD*n*_SOFF)

Table 193. 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.

19.3.21 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.

Offset: 0x1008 + n * 0x20 (n = 0 to 15) 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	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x

1. x = Undefined at reset

Figure 132. TCD Minor Byte Count (Minor Loop Disabled) (DMA_TCD*n*_NBYTES_MLNO)

Table 194. 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.

19.3.22 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

Offset: 0x1008 + n * 0x20 (n = 0 to 15) Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	SMLOE	DMLOE	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	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x

1. x = Undefined at reset

Figure 133. TCD Signed Minor Loop Offset (Minor Loop Enabled and Offset Disabled) (DMA_TCD*n*_NBYTES_MLOFFNO)

Table 195. 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
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.

19.3.23 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)

Offset: 0x1008 + n * 0x20 (n = 0 to 15) 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	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x

1. x = Undefined at reset

Figure 134. TCD Signed Minor Loop Offset (Minor Loop and Offset Enabled) (DMA_TCD*n*_NBYTES_MLOFFYES)

Table 196. 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.

19.3.24 TCD Last Source Address Adjustment (DMA_TCD*n*_SLAST)

Offset: 0x100C + n * 0x20 (n = 0 to 15) 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	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x

1. x = Undefined at reset

Figure 135. TCD Last Source Address Adjustment (DMA_TCD*n*_SLAST)

Table 197. 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.

19.3.25 TCD Destination Address (DMA_TCD*n*_DADDR)

Offset: 0x1010 + n * 0x20 (n = 0 to 15) 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	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x

1. x = Undefined at reset

Figure 136. TCD Destination Address (DMA_TCD*n*_DADDR)

Table 198. DMA_TCD*n*_DADDR field descriptions

Field	Description
0:31 DADDR	Destination address Memory address pointing to the destination data.

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

Offset: 0x1014 + n * 0x20 (n = 0 to 15) Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	ELINK	0	0	LINKCH					CITER							
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 137. TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD*n*_CITER_ELINKYES)

Table 199. 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
3:6 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.

19.3.27 TCD Current Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD*n*_CITER_ELINKNO)

Offset: 0x1014 + n * 0x20 (n = 0 to 15) 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 138. TCD Current Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD*n*_CITER_ELINKNO)

Table 200. 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.

19.3.28 TCD Signed Destination Address Offset (DMA_TCD*n*_DOFF)

Offset: 0x1016 + n * 0x20 (n = 0 to 15) 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 139. TCD Signed Destination Address Offset (DMA_TCD*n*_DOFF)

Table 201. 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.

19.3.29 TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD*n*_DLASTSGA)

Offset: 0x1018 + n * 0x20 (n = 0 to 15) 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	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x

1. x = Undefined at reset

Figure 140. TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD*n*_DLASTSGA)

Table 202. 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.

19.3.30 TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD*n*_BITER_ELINKYES)

Offset: 0x101C + n * 0x20 (n = 0 to 15) Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	ELINK	0	0	LINKCH					BITER							
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 141. TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD*n*_BITER_ELINKYES)

Table 203. 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
3:6 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.

19.3.31 TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD*n*_BITER_ELINKNO)

Offset: 0x101C + n * 0x20 (n = 0 to 15) 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 142. TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD*n*_BITER_ELINKNO)

Table 204. 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.

19.3.32 TCD Control and Status (DMA_TCD*n*_CSR)

Offset: 0x101E + n * 0x20 (n = 0 to 15) 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 143. TCD Control and Status (DMA_TCD*n*_CSR)

Table 205. 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 205. 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

19.4 Functional description

This section provides a description of the DMA request sources, an overview of the microarchitecture, and functional operation of the eDMA module.

19.4.1 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.

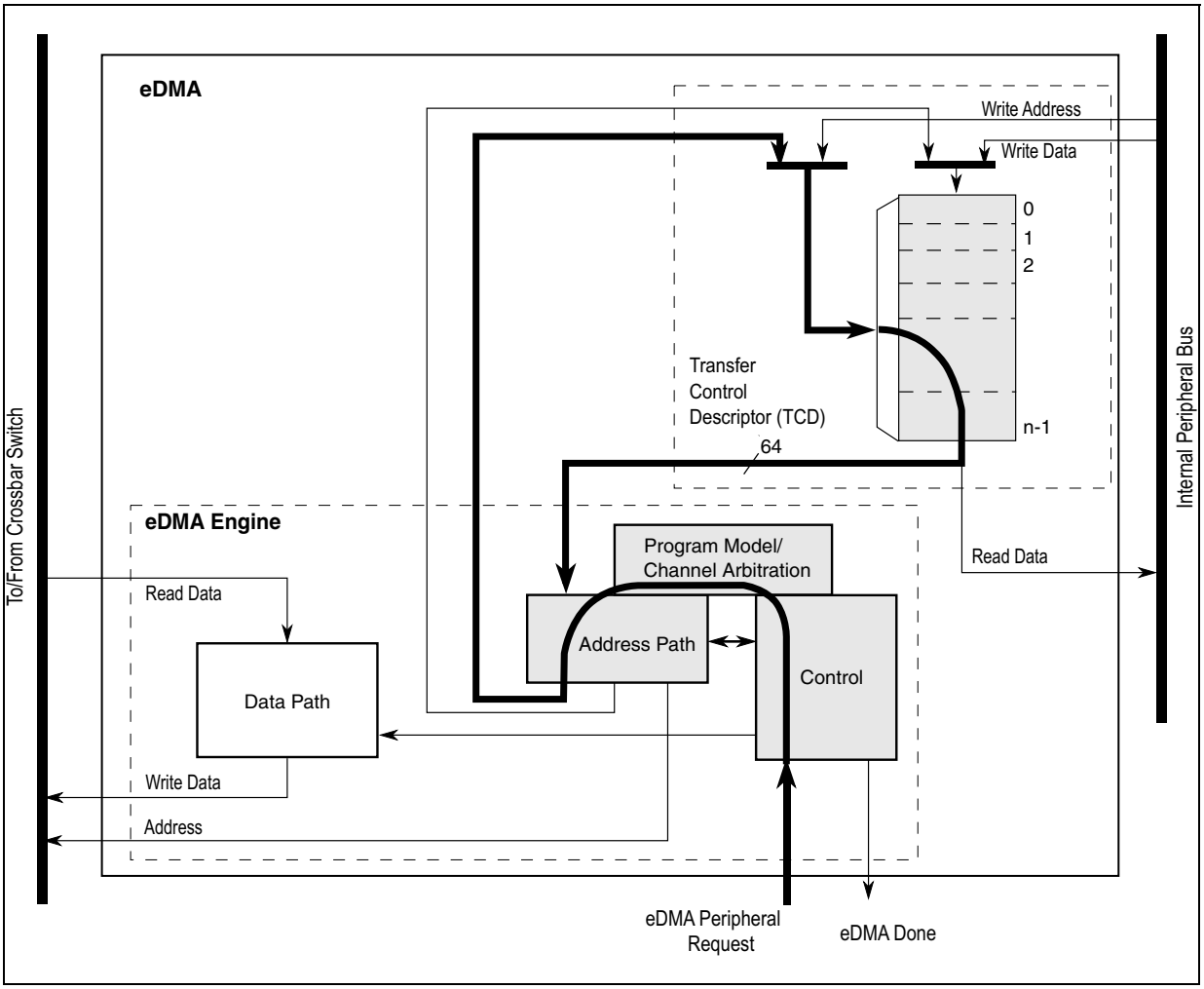
    As noted earlier, in the event of simultaneous accesses, the eDMA engine is given priority and the peripheral transaction is stalled.
  - Memory array:

    TCD storage is implemented using a single-port, synchronous RAM array.

19.4.2 eDMA basic data flow

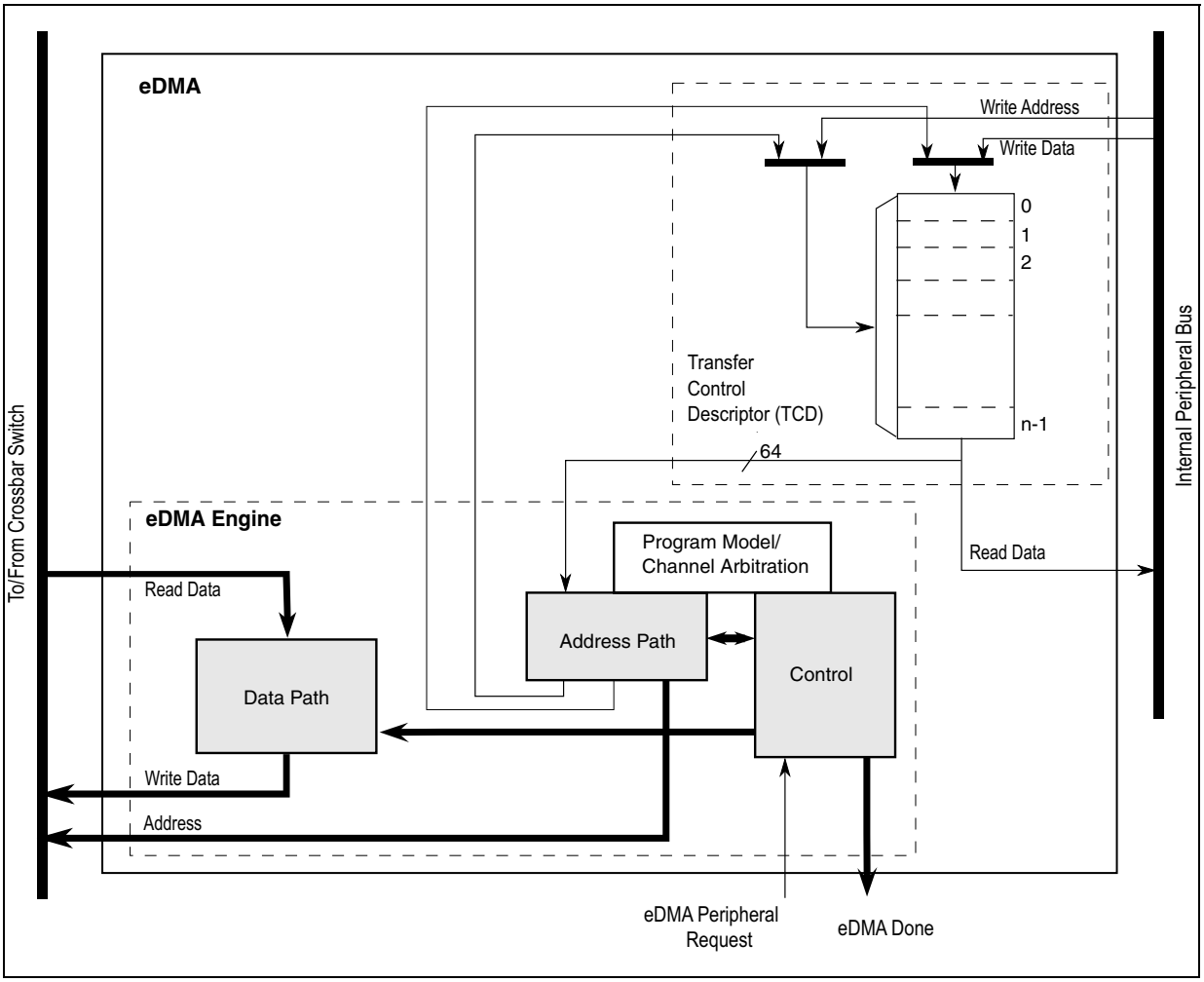
The basic flow of a data transfer can be partitioned into three segments. As shown in *Figure 144*, the first segment involves the channel activation. In the diagram, this example uses the assertion of the eDMA peripheral request signal to request service for channel *n*. Channel activation via software 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 144. DMA operation, part 1



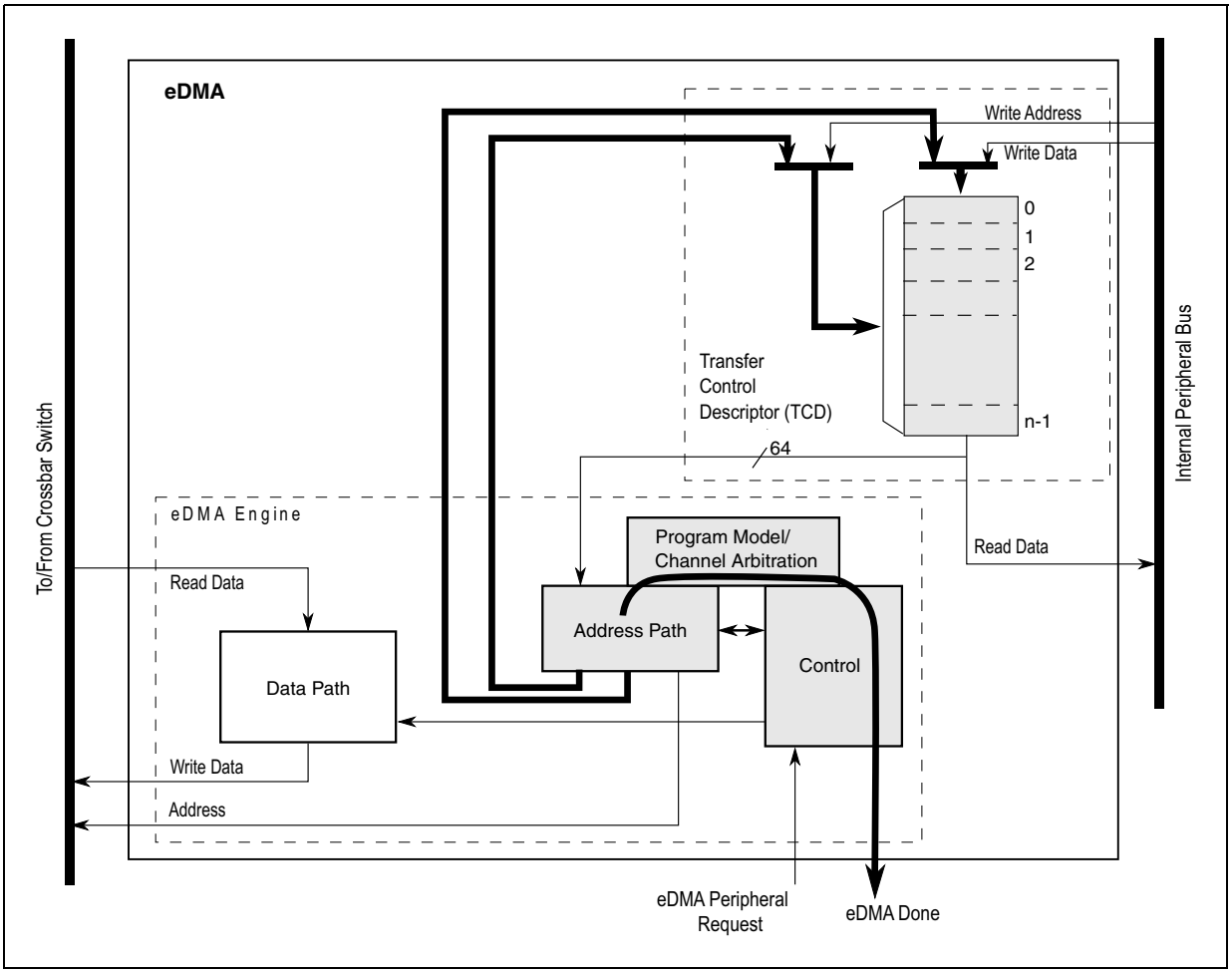
In the second part of the basic data flow, as shown in *Figure 145*, the modules associated with the data transfer (address path, data path, and control) sequence through source reads and destination writes to perform the actual data movement. The source reads are initiated and the fetched data is temporarily 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 145. 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 146*.

Figure 146. DMA operation, part 3



19.4.3 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.

19.4.4 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.

19.4.5 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 206* and *Table 207*. 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 32 bits in size.

Table 206 presents a peak transfer rate comparison, measured in megabytes per second.

Table 206. 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
80 MHz, 32 bits	160.0	60.0	60.0
100 MHz, 64 bits	400.0	160.0	160.0

Table 206. 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 $x^{(k)}$: 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. For a single SRAM read and two 64-bit internal peripheral bus writes, the combined data phase time is five cycles.
- Cycle $x^{(k)}$ + 1: This cycle represents the data phase of the last destination write.
- Cycle $x^{(k)}$ + 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 $x^{(k)}$ + 3: The appropriate fields in the first part of the TCD*n* are written back into the local memory.

k. "*x*" represents an undefined cycle value dependant on transfer sizes and response times for the channel's read and write accesses.

- Cycle $x^{(k)}$ + 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 $x^{(k)}$ + 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 207*. 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 207. eDMA peak request rate (MReq/s)

System frequency (MHz)	Request rate	
	zero wait states	with wait states
80.0	8.8	8.0
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 6 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 7 PEAKreq = 150 MHz / [4 + (1 + 0) + (1 + 1) + 3] cycles = 15.0 Mreq/s

For an internal peripheral bus-to-SRAM transfer,

Equation 8 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.

19.5 Initialization/application information

The following sections discuss initialization of the eDMA and programming considerations.

19.5.1 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 208*) 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 208. 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

Table 208. TCD control and status fields(Continued)

TCD*n*_CSR field name	Description
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 147 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 147. Example of multiple loop iterations

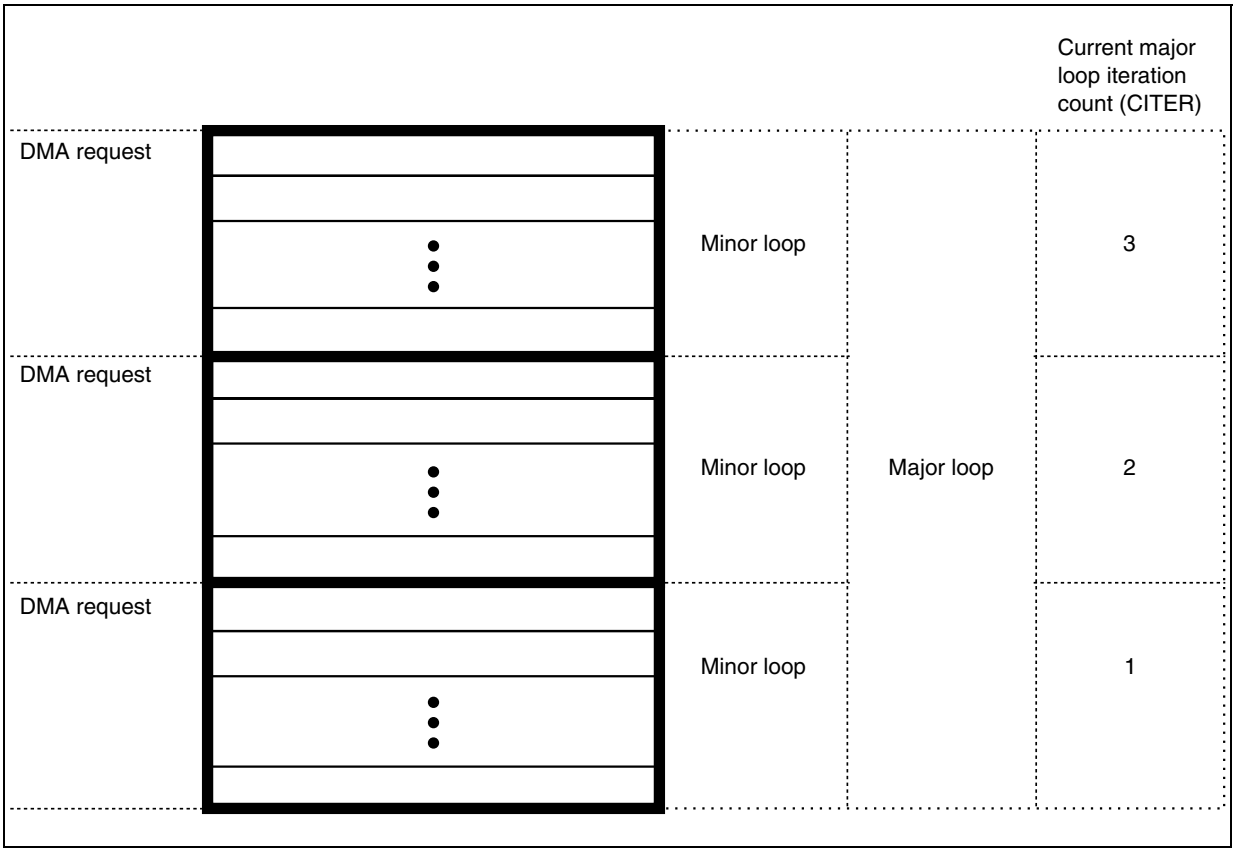
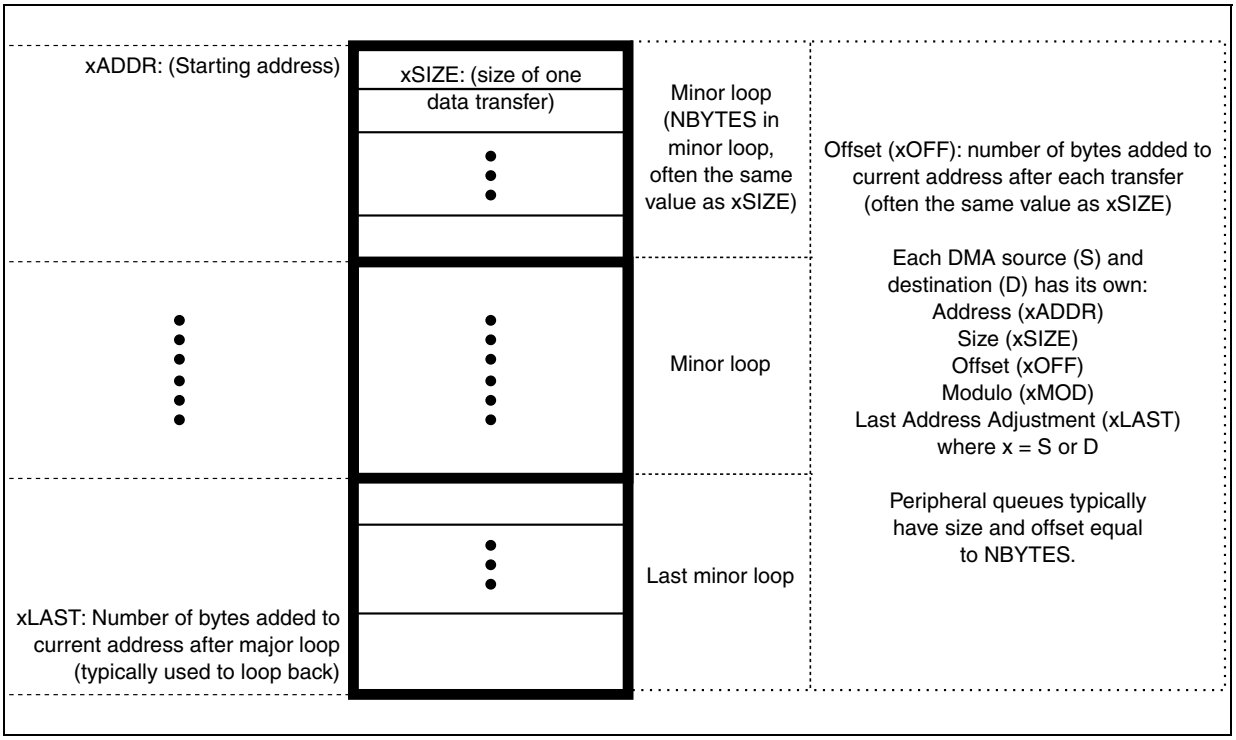


Figure 148 lists the memory array terms and how the TCD settings interrelate.

Figure 148. Memory array terms



19.5.2 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 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.

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.

19.5.3 DMA Arbitration mode considerations

This section discusses arbitration considerations for the eDMA.

19.5.3.1 Fixed-channel arbitration

In this mode, the channel service request from the highest priority channel is selected to execute. 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.

19.5.3.2 Round-robin channel arbitration

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.

19.5.4 DMA transfer

This section discusses how to perform DMA transfers with the eDMA.

19.5.4.1 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.

19.5.4.2 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.

19.5.4.3 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 209 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 209. Modulo feature example

Transfer number	Address
1	0x12345670
2	0x12345674
3	0x12345678
4	0x1234567C
5	0x12345670
6	0x12345674

19.5.5 eDMA TCD*n* status monitoring

This section discusses how to monitor eDMA status.

19.5.5.1 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 210. 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
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 211. 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.

19.5.5.2 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.

19.5.5.3 Preemption status

Preemption is available only when fixed arbitration is selected. 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 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.

19.5.6 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 212 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 212. 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)
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

19.5.7 Dynamic programming

19.5.7.1 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, then change the channel priorities, then enable the appropriate channels.

19.5.7.2 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 213* is recommended when executing a dynamic channel link request.

Table 213. 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.

19.5.7.3 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.

19.5.7.3.1 Method 1 (channel not using major loop channel linking)

For a channel not using major loop channel linking, the coherency model in *Table 214* 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 214. 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.
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).

19.5.7.3.2 Method 2 (channel using major loop linking)

For a channel using major loop channel linking, the coherency model in *Table 215* may be used for a dynamic scatter/gather request. This method uses the TCD.dlast_sga field as a TCD identification (ID).

Table 215. 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.

Table 215. Coherency model for method 2(Continued)

Step	Action
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).

20 DMA channel multiplexer (DMACHMUX)

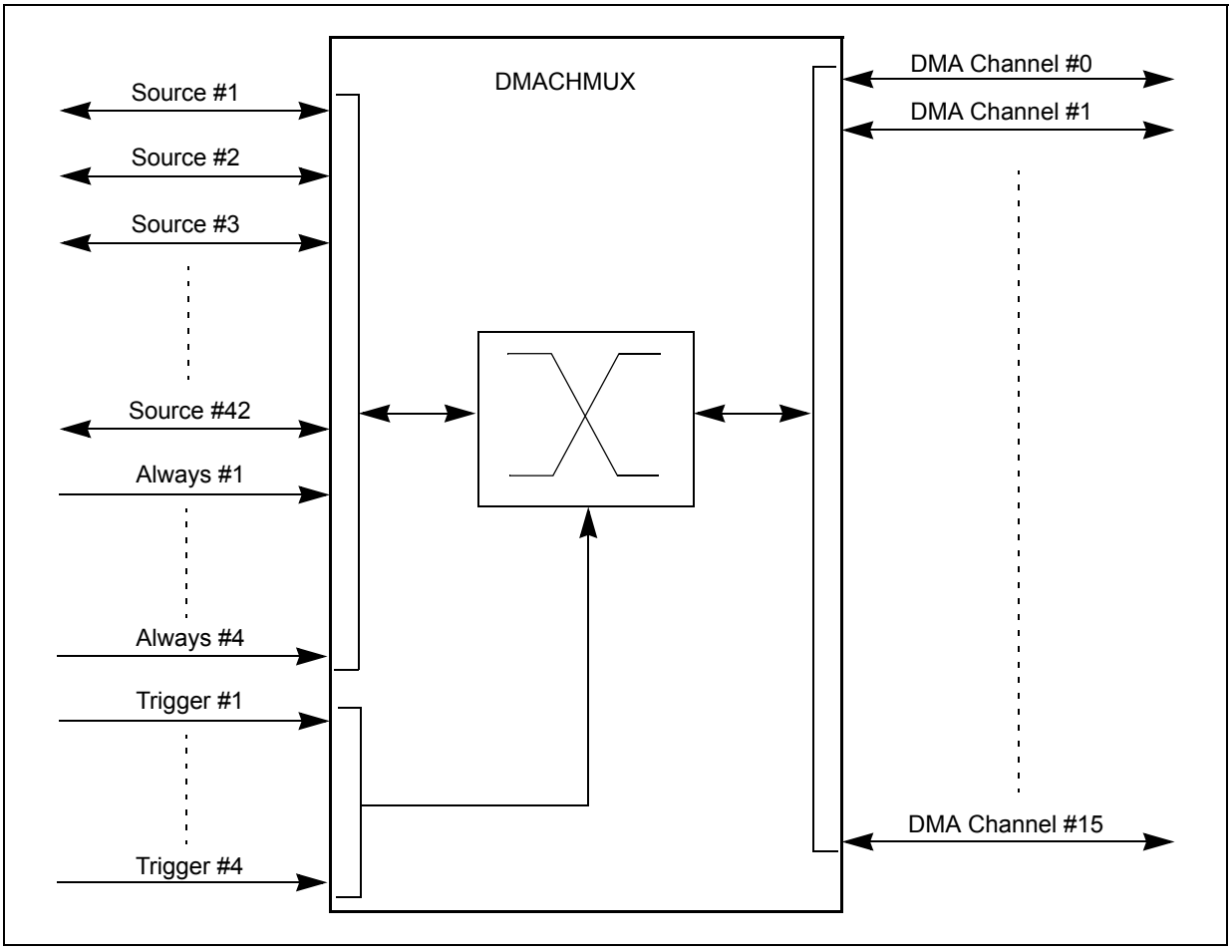
20.1 Introduction

Note: For the chip-specific implementation details of this module's instances, see the chip configuration information.

20.1.1 Overview

The DMA channel multiplexer (DMACHMUX) allows to route 42 DMA peripheral sources (called slots) to 16 DMA channels. This is illustrated in *Figure 149*.

Figure 149. DMACHMUX block diagram



20.1.2 Features

The DMACHMUX provides these features:

- 42 peripheral slots + 4 always-on slots can be routed to 16 channels
- 16 independently selectable DMA channels routers
  - the first 4 channels additionally provide a trigger functionality
- Each channel router can be assigned to one of 42 possible peripheral DMA slots or to one of the 4 always-on slots.

20.1.3 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 (e.g., 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 0–3 for each DMACHMUX instances.

20.2 External signal description

20.2.1 Overview

The DMACHMUX has no external pins.

20.3 Memory map and register definition

Table 216 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 216. DMACHMUX memory map

Address offset	Register	Access	Location
eDMA channel registers per channel (n = 0 to 15)			
0x0000 + n * 0x1	Channel n Configuration (CHCFG*n*)	R/W	*Section 20.3.1.1*

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.

20.3.1 Register descriptions

This section provides a detailed description of the memory-mapped registers in the DMACHMUX

20.3.1.1 Channel 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 150. Channel Configuration Registers (CHCFG*n*)

Table 217. 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 218. 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.

20.4 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 20.5.2: Enabling and configuring sources* 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:

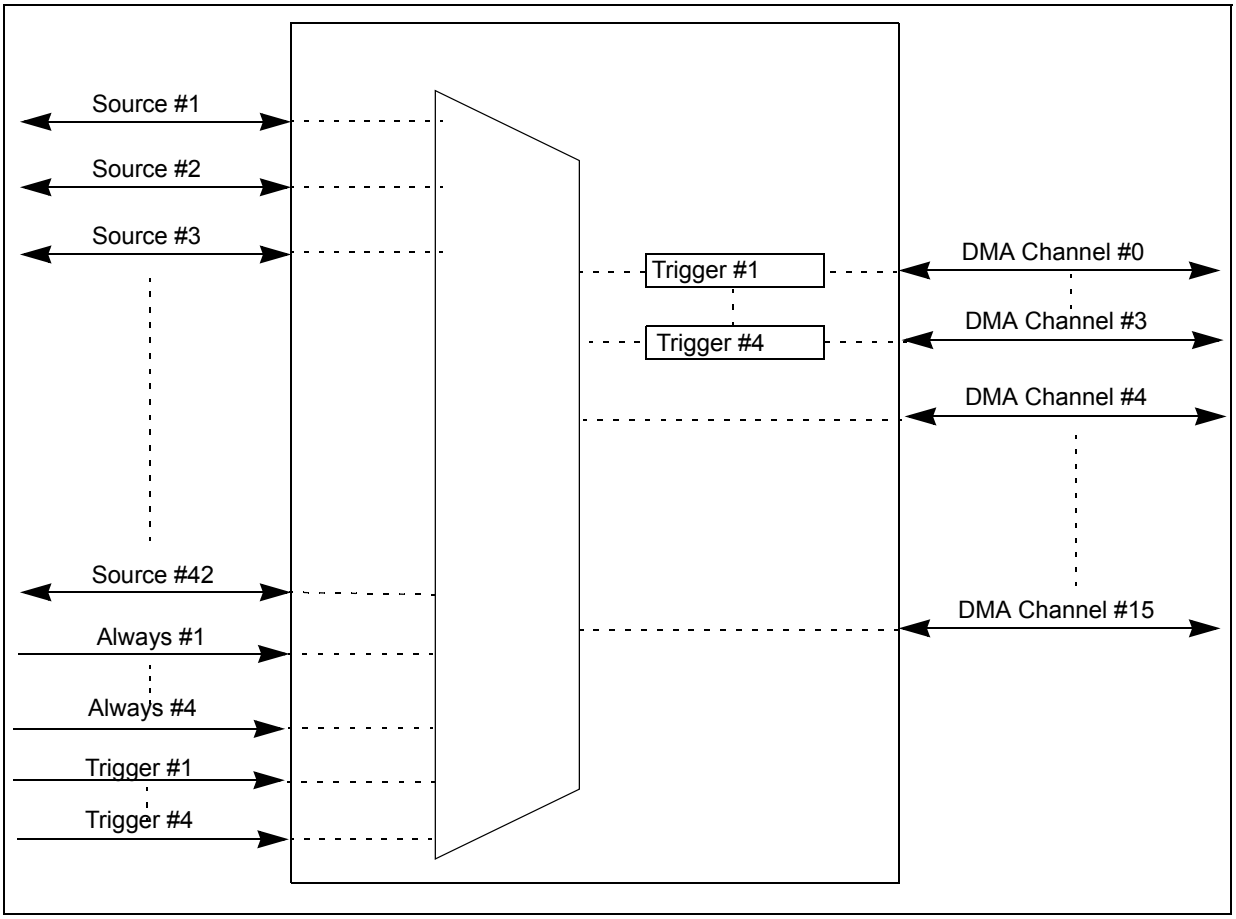
- channels which implement the normal routing functionality plus periodic triggering capability.
- channels which implement only the normal routing functionality.

20.4.1 DMA channels with periodic triggering capability

Besides the normal routing functionality, the first 4 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): as such, 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.

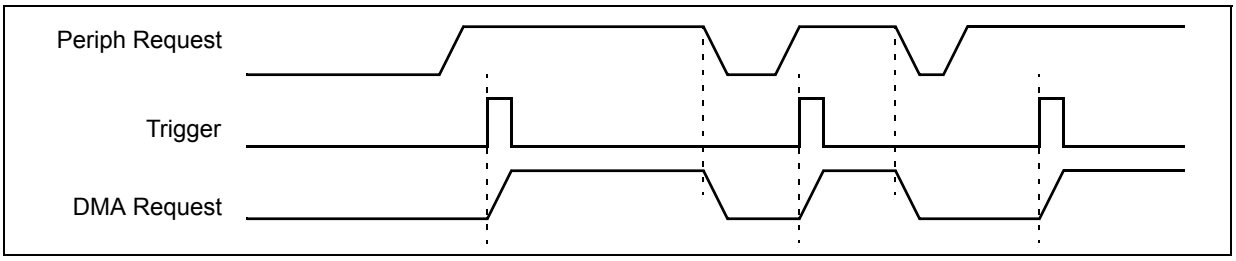
Note: Because of the dynamic nature of the system (i.e., DMA channel priorities, bus arbitration, interrupt service routine lengths, etc.), the number of clock cycles between a trigger and the actual DMA transfer cannot be guaranteed.

Figure 151. DMACHMUX triggered channels



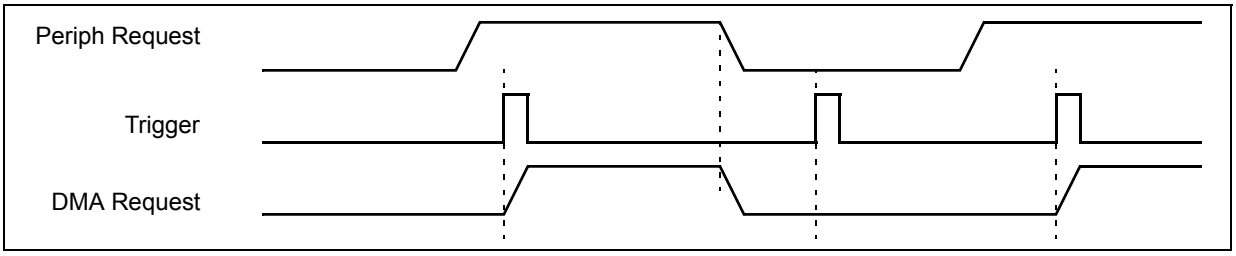
The DMA channel triggering capability allows the system to schedule regular DMA transfers, usually on the transmit side of certain peripherals, without the intervention of the processor. This trigger works by gating the request from the Peripheral to the DMA until a trigger event has been seen. This is illustrated in *Figure 152*.

Figure 152. 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 153*.

Figure 153. 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 (i.e., resolution, range of values, etc.) may be found in the Periodic Interrupt Timer (PIT) chapter.

20.4.2 DMA channels with no triggering capability

The other channels of the DMACHMUX provide the normal routing functionality as described in *Section 20.1.3: Modes of operation*.

20.4.3 “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 (i.e., 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.

20.5 Initialization/Application information

20.5.1 Reset

The reset state of each individual bit is shown within the Register Description section (See *Section 20.3.1: Register descriptions*). In summary, after reset, all channels are disabled and must be explicitly enabled before use.

20.5.2 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 4 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;
```

21 Wakeup unit (WKPU)

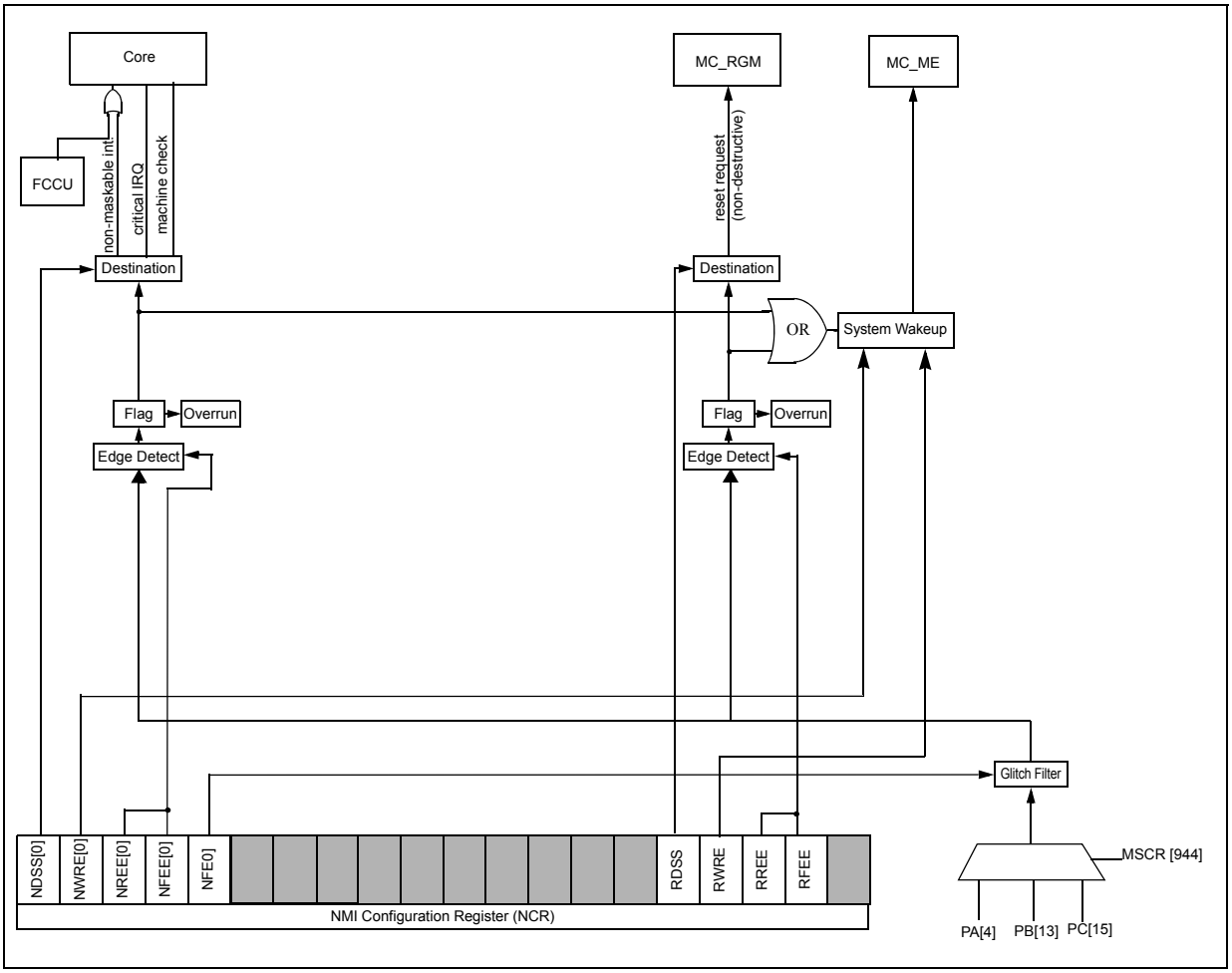
21.1 Introduction

21.1.1 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 154* 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 154. Wakeup unit Block Diagram



21.1.2 Features

The Wakeup unit supports these distinctive features:

Non-maskable Interrupt support with

- up to 4 NMI sources
- up to 4 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

21.2 Memory Map and Register Description

This section provides a detailed description of all registers accessible in the WKPU module.

21.2.1 Memory Map

Table 219 gives an overview on the WKPU registers implemented.

Table 219. WKPU Memory Map

Address Offset	Registers	Supported Access Sizes	Location
0x0000	NMI Status Flag Register (NSR)	32/16/8	*Section 21.2.2.1*
0x0004 - 0x0007	Reserved		
0x0008	NMI Configuration Register (NCR)	32/16/8	*Section 21.2.2.2*
0x000C - 0x03FF	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.

21.2.2 Register Descriptions

This section describes in address order all the Wakeup unit registers.

21.2.2.1 NMI Status Flag Register (NSR)

This register holds the non-maskable interrupt status flags.

Offset: 0x0000 Access: User read/write (write 1 to clear)

	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 155. NMI Status Flag Register (NSR)

Table 220. 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
16 NIF2	NMI Status Flag 2. This flag can be cleared only by writing a 1. Writing a 0 has no effect. If enabled (NREE2 or NFEE2 set), NIF2 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 NREE2 and NFEE2 has occurred 0 No event has occurred on the pad
17 NOVF2	NMI Overrun Status Flag 2. This flag can be cleared only by writing a 1. Writing a 0 has no effect. It will be a copy of the current NIF2 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 (NREE2 or NFEE2 set), NOVF2 causes an interrupt request. 1 An overrun has occurred on NMI input 2 0 No overrun has occurred on NMI input 2

Table 220. NSR Field Descriptions(Continued)

Field	Description
24 RIF	Reset status Flag This flag can be cleared only by writing a 1. Writing a 0 has no effect. If enabled (i.e., 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 device configuration chapter to see how the NMI Status Flag Register (NSR) is used in the device.

21.2.2.2 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
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 (1)

Figure 156. NMI Configuration Register (NCR)

1. Writing this bit has no functional impact.

Table 221. 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 A set NIF0 bit or set NOVF0 bit causes a system wakeup request 0 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.
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 A set NIF1 bit or set NOVF1 bit causes a system wakeup request 0 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.

Table 221. NCR Field Descriptions(Continued)

Field	Description
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 A set NIF2 bit or set NOVF2 bit causes a system wakeup request 0 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
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 device configuration chapter to see how the 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)!

21.3 Functional Description

21.3.1 General

This section provides a complete functional description of the Wakeup unit.

21.3.2 Non-Maskable Interrupts

The Wakeup unit supports up to 4non-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.

21.3.2.1 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 156*). 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 221* for details.

Each NMI supports a status flag and an overrun flag which are located in the NSR register (see *Figure 155*). 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.

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.

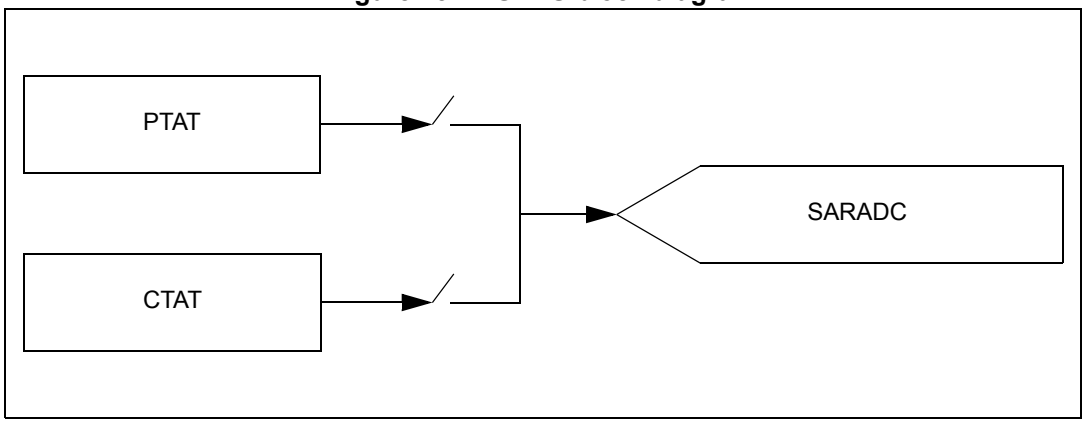
22 Temperature sensor (TSENS)

22.1 Introduction

The TSENS module provides an analog voltage that is proportional to the internal temperature of the SPC570Sx. This voltage can be sampled by the ADC and converted to an actual temperature by using factory-programmed calibration constants.

Figure 157 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 157. TSENS block diagram



22.2 Features

The TSENS has the following features:

- Temperature monitoring range: –40 to +150 °C
- Sensitivity: approximately 5.14 mV/°C

22.3 Signals

The TSENS does not use any input signals external to the device.

See *Table 336: SARADC_B analog test channel assignment* for information on which ADC channels are connected to V_{TSENS}.

22.4 TSENS outputs

TSENS produces two outputs from separate sources:

- Proportional to absolute temperature (PTAT).
- Complementary to absolute temperature (CTAT).

22.5 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 SPC570Sx test flash memory (see *Section 22.5.1*).
- Measure PTAT and CTAT voltages through the ADC.
- Calculate the device temperature using the equations in *Section 22.5.2*.

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.

22.5.1 TSENS calibration constants

The equations needed to convert V_{TSENS} into device temperature information depend on four calibration constants as described in *Table 222*. These constants are determined during factory testing and stored in the test flash memory.

Table 222. 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

22.5.2 Equations for converting TSENS voltage to device temperature

In the equations below:

- P_n and C_n are the calibration constants described in *Section 22.5.1*
- 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 9

$$A = P_x C_2 - P_2 C_x$$

Equation 10

$$B = C_x P_1 - P_x C_1$$

The temperature is calculated as:

Equation 11

$$T = T_2 + \frac{(T_1 - T_2) \cdot A}{A + B}$$

22.6 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 222*
- 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 12

$$\text{ref_code} = P_x + \left(\frac{(P_2 - P_1)}{(C_1 - C_2)}\right) . Cx$$

Note: It is mandatory to measure Px and Cx at the same ADC reference voltage and at the same temperature.

23 Clocking

23.1 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 48: 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. Four Auxiliary Clock Selectors are utilized, allowing developers to select an independent clock source for each of the system peripherals.

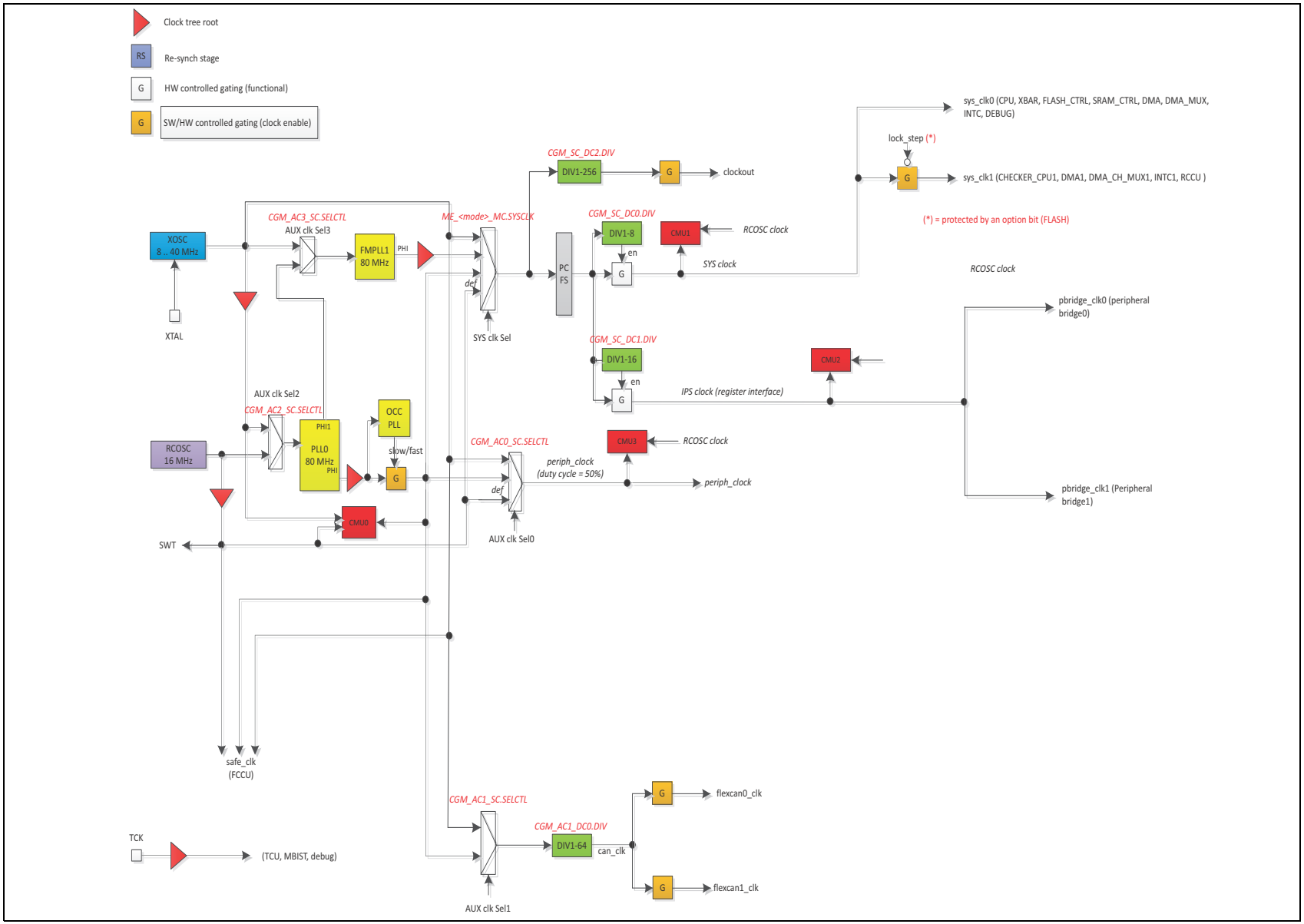
Each of the outputs of the module clock selectors have individual dividers that allow for more clock frequency granularity with the ability to divide the clock selector outputs by up to 512 for a given peripheral. Clock Monitor Units (CMUs) are connected to outputs of clock dividers as shown in *Figure 158: 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.

23.2 Clock generation

The top-level clock generation architecture can be seen in *Figure 158*. 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 158*.

Note: CMU_1, CMU_2 and CMU_3 in Figure 158 use IRCOSC for their reference clock, whereas the user can select either IRCOSC or XOSC for the reference in CMU_0.

Figure 158. Clock generation architecture



23.2.1 MC_CGM registers

Table 223 shows the relationship between the clocks in *Figure 158* and the MC_CGM registers.

Table 223. 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_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	SARADC (SAR_CLK)	—
CGM_AC0_DC2 CGM_AC0_SC	Aux Clock 0 Divider Configuration 2 Aux Clock 0 Select Control	CTU clock divisor	—
CGM_AC0_DC3 CGM_AC0_SC	Aux Clock 0 Divider Configuration 3 Aux Clock 0 Select Control	DSPI clock (DSPI_CLK)	—
CGM_AC0_DC4 CGM_AC0_SC	Aux Clock 0 Divider Configuration 4 Aux Clock 0 Select Control	LINFlexD clock (LIN_CLK)	—
CGM_AC0_DC5 CGM_AC0_SC	Aux Clock 0 Divider Configuration 5 Aux Clock 0 Select Control	eTimer clock divisor	—
CGM_AC1_DC0 CGM_AC1_SC	Aux Clock 1 Divider Configuration 0 Aux Clock 1 Select Control	FlexCAN clock divisor	—
CGM_AC2_SC	Aux Clock 2 Select Control	PLL0	—
CGM_AC3_SC	Aux Clock 3 Select Control	PLL1	—

1. See the 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, DMA, DMACHMUX, XBAR, INTC, PFlash controller, PRAM controller, Debug.

23.3 System clock frequency limitations

The maximum frequency of operation for the device system level clocks is given in *Table 224*.

Table 224. Maximum system level clock frequencies

System clock	Max frequency (MHz)(1)
Core	80
XBAR, system RAM	80
PBRIDGE_0	40
DMA	80

Table 224. Maximum system level clock frequencies(Continued)

System clock	Max frequency (MHz)[1]
Interrupt controller	40
DSPI_CLK, LIN_CLK	80
Peripheral clock	80
ADCs	12
FlexCAN	40
CTU	80
eTimer	80
PIT	80

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 (40 MHz).

The ratio between CORE_CLK and PBRIDGE0_CLK shall not be greater than 2.

23.3.1 JTAG frequencies

Table 225 shows the maximum frequencies of the JTAG interface.

Table 225. JTAG frequencies

Configuration	e200z0 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 80 MHz	40 MHz		JTAG usable at half (1/2) CPU frequency

23.4 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.

23.5 Clock sources

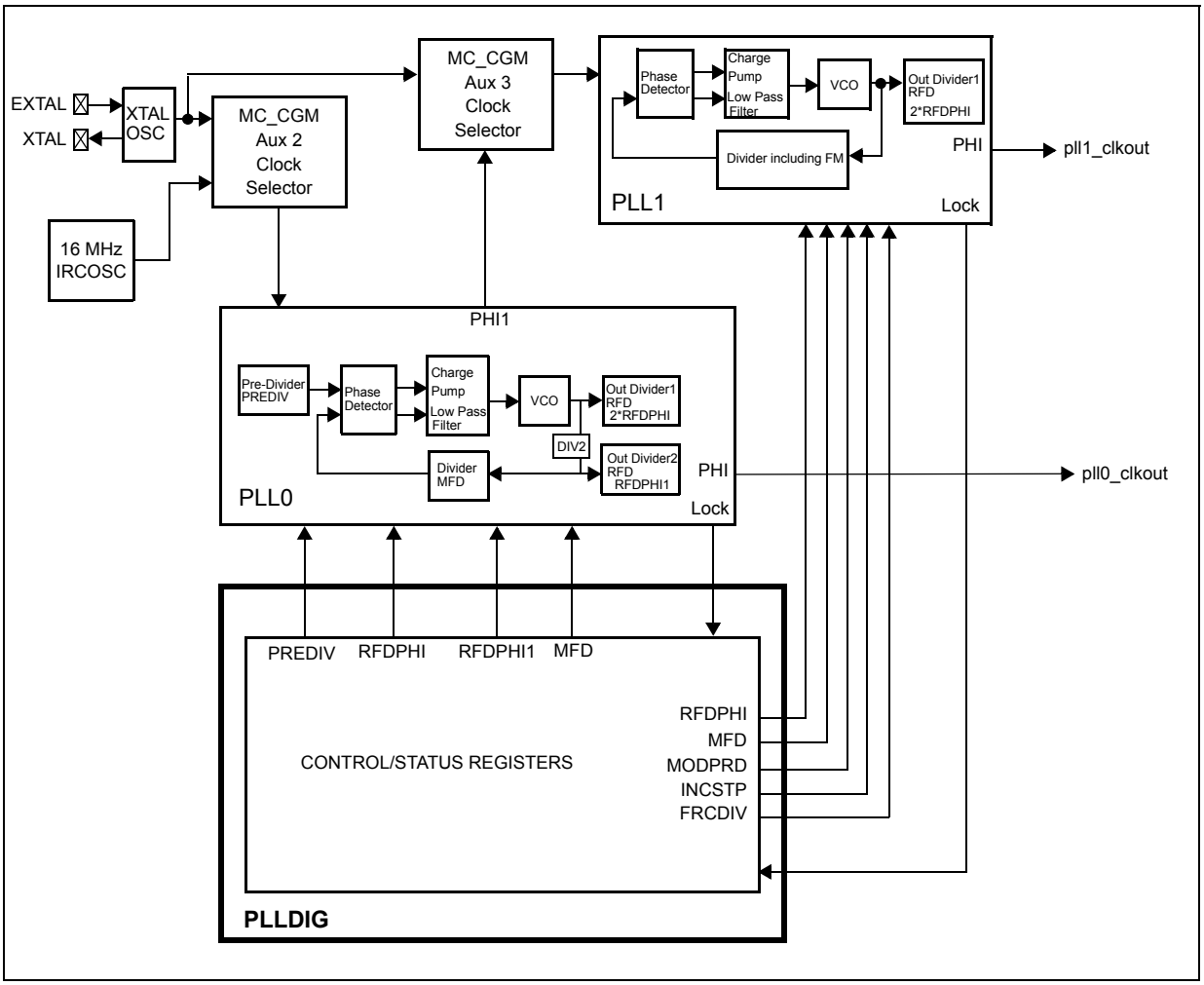
The following clock sources are described in this section:

- PLL0
- PLL1 (FMPLL)
- External oscillator (XOSC)
- External clock (EXTAL bypass)
- 16 MHz internal RC oscillator (IRCOSC)

23.5.1 PLL

The PLL is a dual-PLL providing separate system and peripheral clocks as shown in *Figure 158*. 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 159*.

Figure 159. SPC570Sx dual PLL digital interface block diagram



23.5.1.1 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.

23.5.1.1.1 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.

23.5.1.1.2 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.

23.5.1.2 PLL1 — FMPLL

PLL1 is a Frequency Modulated PLL (FMPLL) that is used to drive the system clock.

23.5.1.2.1 PLL1 input clocks

The input clocks to PLL1 are the PHI1 output clock from PLL0 and the external reference clock.

23.5.1.2.2 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.

23.5.1.2.3 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 7.10.4.4: PLL Digital Interface (PLL_DIG) protected registers of the Chapter 7: Device configuration for details.

23.5.1.3 PLL register reset values

All reset values for the PLL_DIG registers that are device-specific are shown in *Table 226*. All other register reset values are shown in the Dual PLL Digital Interface (PLLDIG) chapter.

Table 226. 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

23.5.1.4 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.

23.5.2 External oscillator (XOSC)

The external oscillator (XOSC) is provided as a reference for the on-chip PLLs. It can also can 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 – 44 MHz crystal inputs, and has integrated load capacitors.

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.

23.5.2.1 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 7.10.4.5: Oscillator Digital Interface (XOSC) protected registers of the Chapter 7: Device configuration for details.

23.5.2.2 XOSC reset value

Table 227 shows the default reset value for the XOSC registers.

Table 227. XOSC register reset values

Offset	Register	Reset value
00h	XOSC_CTL—XOSC Control Register	0030_0000h

23.5.3 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.

23.5.3.1 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. Refer to *Section 28.3.1: Register descriptions* of *Chapter 28: IRCOSC digital interface* for details about IRCOSC registers.

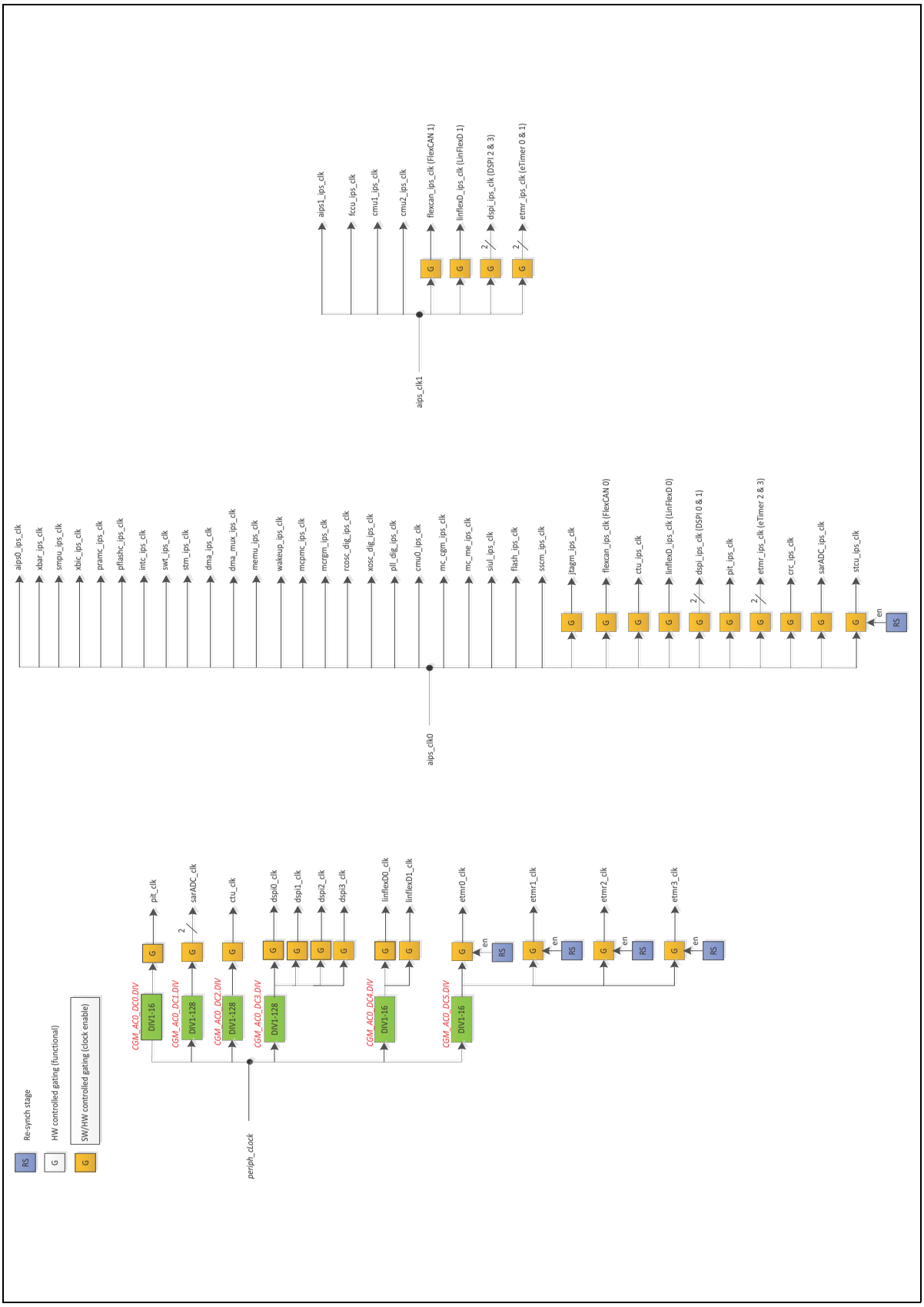
23.5.3.2 IRCOSC register write-protection

The IRCOSC registers do not utilize write-protection.

23.6 Peripheral clocks

Figure 160 shows the clock distribution to the cores and peripheral modules.

Figure 160. Clock distribution



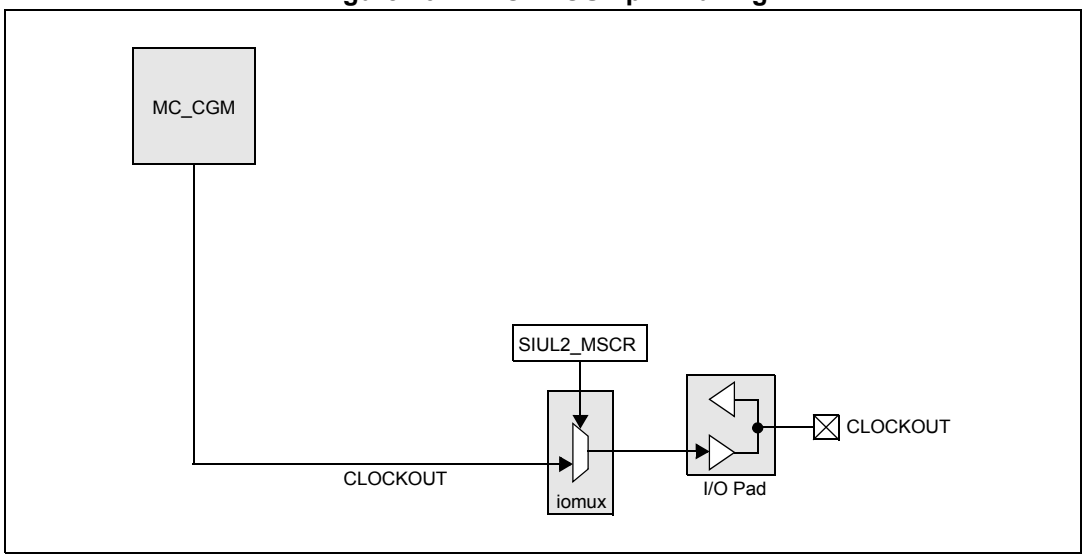
23.6.1 FlexCAN clocking

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).

23.6.2 System clock output pin

There is one system clock output pin on the device: CLOCKOUT. The output clock from PLL0 and external oscillator (XOSC) can be selected for CLOCKOUT. See *Figure 158*.

Figure 161. CLOCKOUT pin muxing



23.7 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 158* 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.

23.7.1 CMU configuration

This section explains the CMU configuration.

Figure 162 shows the block diagram for CMU_0 on the SPC570Sx.

Figure 162. CMU_0 block diagram

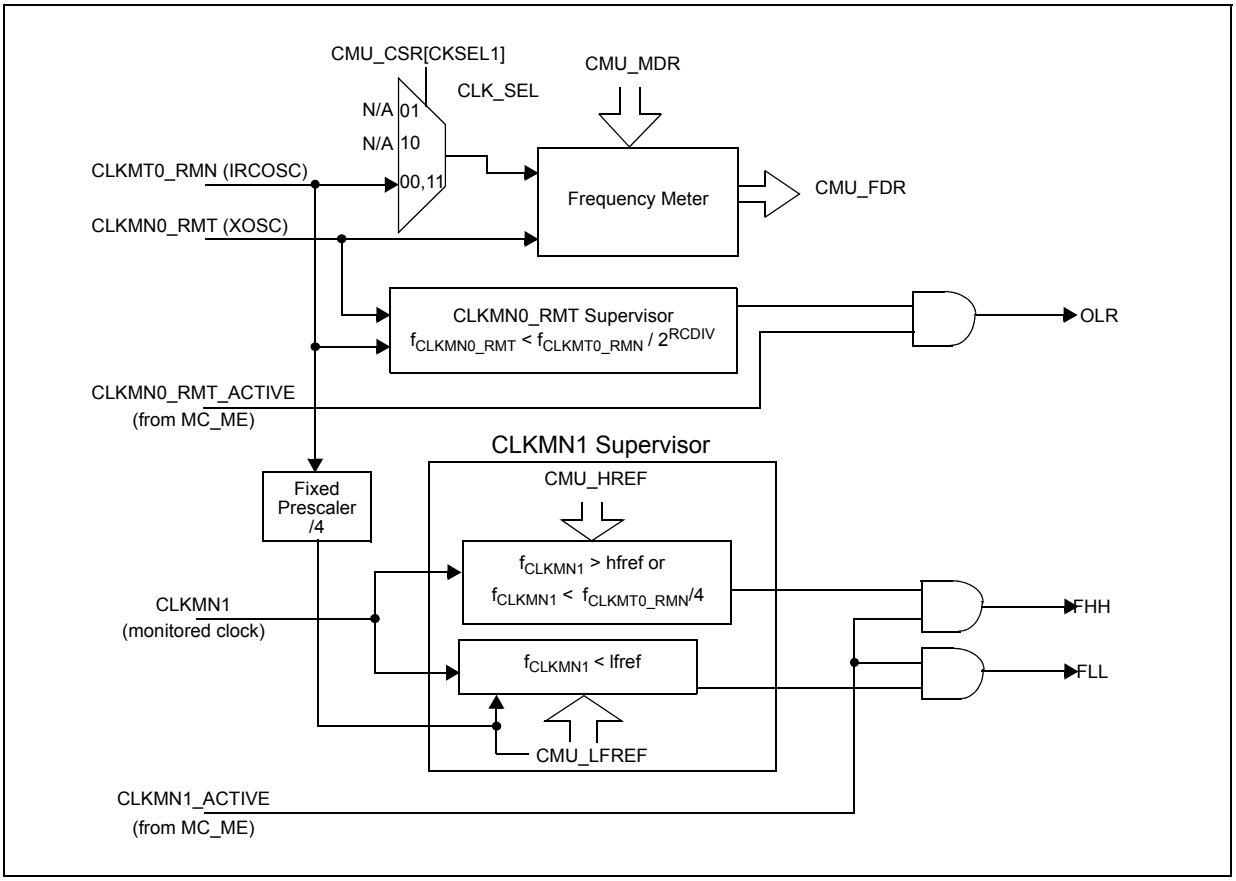
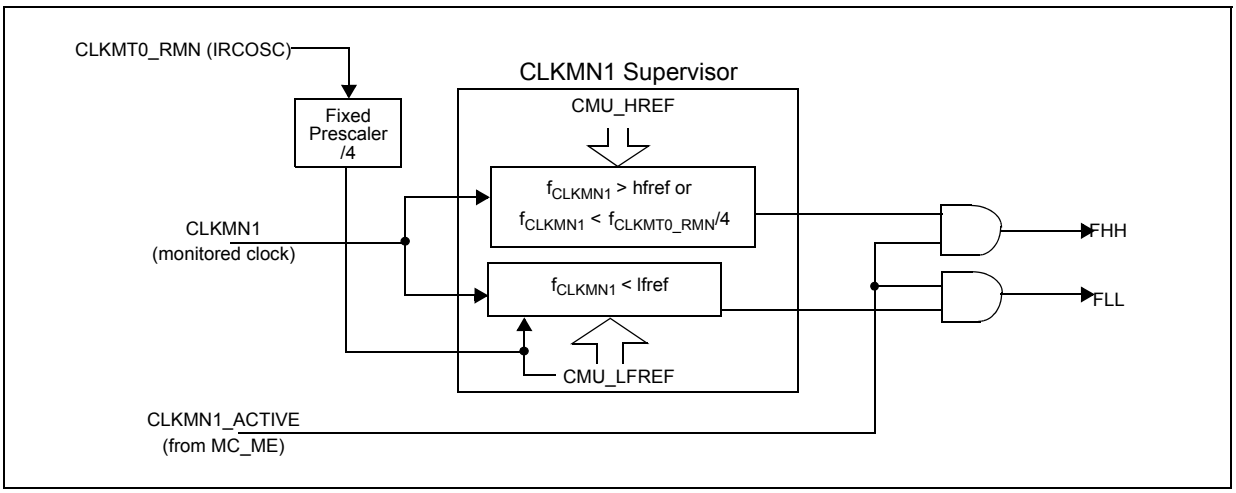


Figure 163 shows the block diagram for CMU[1:3] on the SPC570Sx.

Figure 163. CMU[1:3] block diagram



23.7.1.1 Clock input sources

Table 228 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 228. 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

23.7.1.2 CMU registers and field availability

Table 229 specifies which registers and fields are available on a given CMU for this device.

Table 229. CMU registers availability

Address offset	Register	CMU	Notes
0000h	CMU_CSR	All	Fields CMU_CSR[SFM] and CMU_CSR[CKSEL1] are present in CMU_0 only
0004h	CMU_FDR	CMU_0	—
0008h	CMU_HFREFR	All	—
000Ch	CMU_LFREFR	All	—
0010h	CMU_ISR	All	Fields CMU_ISR[FLCI] and CMU_ISR[OLRI] are present in CMU_0 only.
0014h	Reserved	—	—
0018h	CMU_MDR	All	—

Note: Certain CMU registers can be write-protected. Refer to Section 7.10.4.2: Clock Monitor Unit (CMU) protected registers of the Chapter 7: Device configuration for details.

23.7.2 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.

23.7.3 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.

23.7.4 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.

23.7.5 System clock monitors

A CMU is assigned to monitor the frequency of the core, XBAR, and both peripheral bridges (refer to *Figure 158*).

23.8 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 164* gives a high-level view of the loss of clock logic.

23.8.1 Loss of PLL/XOSC clock

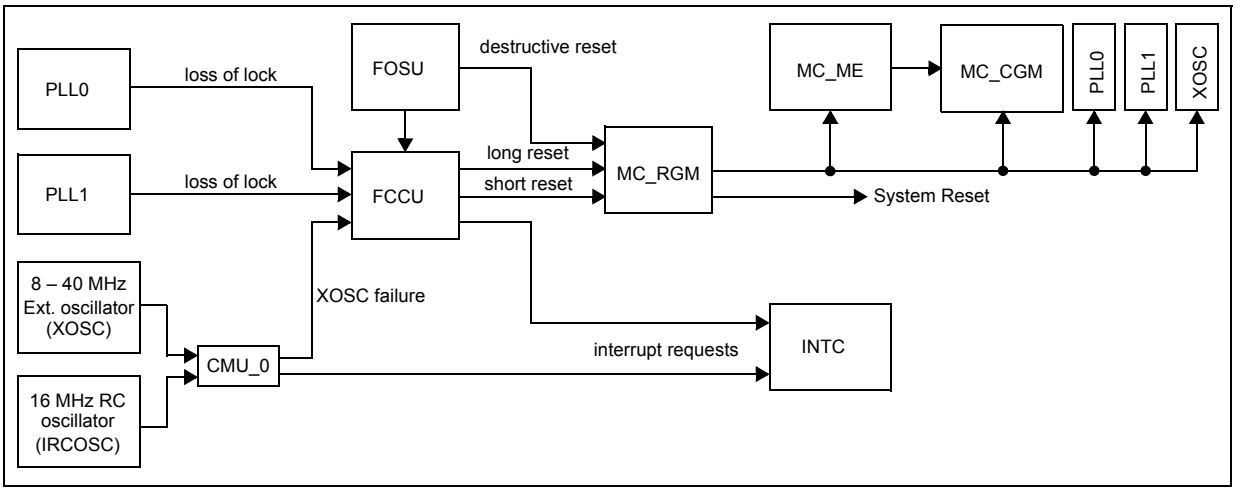
As shown in *Figure 164*, each loss of lock signal from each PLL and the XOSC failure signals are monitored by the FCCU. The FCCU can be programmed to generate a short or long reset sequence when a clock failure occurs, or simply to generate an interrupt. In the case of the interrupt, only the backup 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.

23.8.2 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 164. Loss of clock logic



24 Dual PLL Digital Interface (PLLDIG)

24.1 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 23: 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 the PLL1 PHI frequency modulated output.

24.2 Block diagram

The block diagram of the PLLDIG block is found in *Chapter 23: Clocking*.

24.3 Features

The dual PLL 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
- Input clock frequency range from 8 MHz to 56 MHz before the prescaler, and output clock from 4.762 MHz to 625 MHz for PLL0
- Input clock frequency range from 37.5 MHz to 78.125 MHz, and output clock from 4.762 MHz to 625 MHz for PLL1
- Programmable frequency modulation on PLL1
- Lock detect circuitry reports when PLL0 and PLL1 achieves 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

24.4 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.

24.4.1 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 modulating or not can be programmed by writing to the PLL1FM register, *Section 24.5.2.7: 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.

Only when the PLL0 achieves lock should PLL1 be configured to work in either mode by programming the appropriate MC_ME_<mode>_MC register.

24.5 Memory map and register definition

This section provides the memory map and detailed descriptions of all registers for configuring the PLLs.

24.5.1 Memory map

Table 230 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 230. Dual PLL Digital Interface memory map

Offset	Register	Access	Reset value	Location
0x0000	PLL0CR (PLL0 Control Register)	R/W	0x0000_0000	*Section 24.5.2.1*
0x0004	PLL0SR (PLL0 Status Register)	R/W	0x0000_0000	*Section 24.5.2.2*
0x0008	PLL0DV (PLL0 Divider Register)	R/W	0xzzzz_z0zz(1)	*Section 24.5.2.3*
0x000C–0x001F	Reserved			
0x0020	PLL1CR (PLL1 Control Register)	R/W	0x0000_0000	*Section 24.5.2.4*
0x0024	PLL1SR (PLL1 Status Register)	R/W	0x0000_0000	*Section 24.5.2.5*
0x0028	PLL1DV (PLL1 Divider Register)	R/W	0xzzzz_z0zz(1)	*Section 24.5.2.6*
0x002C	PLL1FM (PLL1 Frequency Modulation Register)	R/W	0x0000_0000	*Section 24.5.2.7*
0x0030	PLL1FD (PLL1 Fractional Divide Register)	R/W	0x0000_0000	*Section 24.5.2.8*
0x0034–0x003B	Reserved			

1. See *Chapter 23: Clocking* for default reset values.

24.5.2 Register descriptions

Some of the register reset values are specifically configured for each unique device by external configuration signals or parameters.

24.5.2.1 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(1)		EXPDIE	0	0	0	LOLIE	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. CLKCFG can be written, but writes have no effect. Mode changes are implemented by writing to the appropriate MC_ME_<*mode*>_MC register.

Figure 165. PLL0 Control Register (PLL0CR)

Table 231. 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 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.

24.5.2.2 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	Reserved	0	LOLF	LOCK	Reserved	0
W									w1c				w1c			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 166. PLL0 Status Register (PLL0SR)

Table 232. 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 remains 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.

24.5.2.3 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 loses lock and generates an interrupt, based on which is enabled. The reduced frequency divider fields (RFDPHI, RFDPHI1) can be modified at any time, but the changes only become effective after PLL0 is disabled, then re-enabled.

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 *Chapter 23: Clocking* for reset value information.

Figure 167. PLL0 Divider Register (PLL0DV)

Table 233. 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. 0000 Reserved 0001 Reserved 0010 Reserved 0011 Reserved 0100 Divide by 4 0101 Divide by 5 0110 Divide by 6 0111 Divide by 7 1000 Divide by 8 1001 Divide by 9 1010 Divide by 10 1011 Divide by 11 1100 Divide by 12 1101 Divide by 13 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. 0x00 Reserved 0x01 Divide by 2 0x02 Divide by 4 0x03 Divide by 6 ... 0x3E Divide by 124 0x3F Divide by 126

Table 233. 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 2 011 Divide by 3 100 Divide by 4 101 Divide by 5 110 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. 0x00 Reserved 0x01 Reserved 0x02 Reserved 0x03 Reserved 0x04 Reserved 0x05 Reserved 0x06 Reserved 0x07 Reserved 0x08 Divide by 8 0x09 Divide by 9 ... 0x7E Divide by 126 0x7F Divide by 127

24.5.2.4 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(1)		EXPDIE	0	0	0	LOLIE	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. CLKCFG can be written, but writes have no effect. Mode changes are implemented by writing to the appropriate MC_ME_<*mode*>_MC register.

Figure 168. PLL1 Control Register (PLL1CR)

Table 234. PLL1CR field descriptions

Field	Description
22:23 CLKCFG	Clock Configuration This field 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 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.

24.5.2.5 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	Reserved	0	LOLF	LOCK	Reserved	0
W									w1c				w1c			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 169. PLL1 Status Register (PLL1SR)

Table 235. 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 remains 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

24.5.2.6 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 loses lock and generates an interrupt, based on which is enabled. The reduced frequency divider field (RFDPHI) can be modified at any time, but the changes only become effective after PLL1 is disabled, then re-enabled.

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					
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						
W																
Reset	0	0	0	0	0	0	0	0	0	_(1)	_(1)	_(1)	_(1)	_(1)	_(1)	_(1)

1. See *Chapter 23: Clocking* for reset value information.

Figure 170. PLL1 Divider Register (PLL1DV)

Table 236. 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. 0x00 Reserved 0x01 Divide by 2 0x02 Divide by 4 ... 0x3E Divide by 124 0x3F Divide by 126
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. 0x10 Divide by 16 0x11 Divide by 17 0x12 Divide by 18 0x13 Divide by 19 0x14 Divide by 20 0x15 Divide by 21 0x16 Divide by 22 0x17 Divide by 23 0x18 Divide by 24 0x19 Divide by 25 0x1A Divide by 26 0x1B Divide by 27 0x1C Divide by 28 0x1D Divide by 29 0x1E Divide by 30 0x1F Divide by 31 0x20 Divide by 32 0x21 Divide by 33 0x22 Divide by 34 All other settings are Reserved

24.5.2.7 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. 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 device datasheet).

Offset: 0x002C Access: User read/write

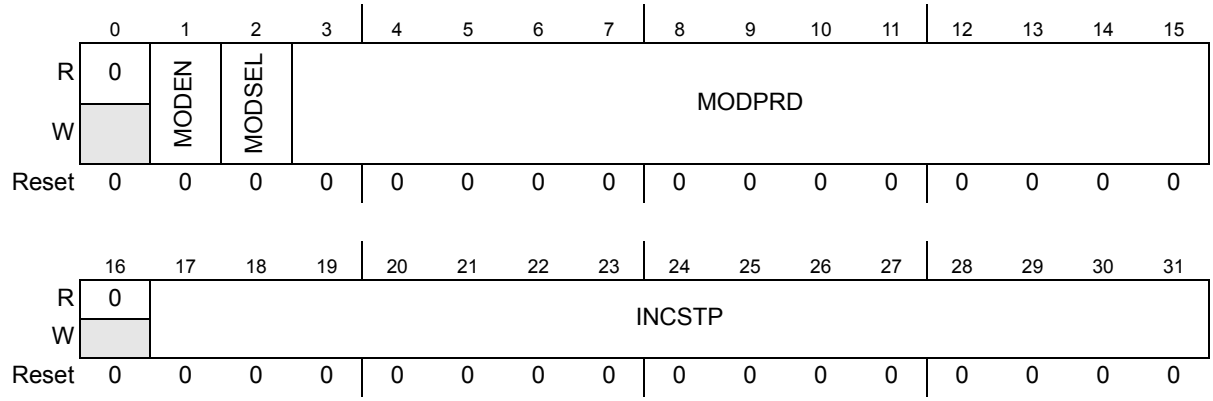


Figure 171. PLL1 Frequency Modulation Register (PLL1FM)

Table 237. 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: $MODPRD = round(f_{ref}/(4 \times f_{mod}))$ where f_{ref} represents the frequency of the feedback divider and f_{mod} represents the modulation frequency.
17:31 INCSTP	Increment step This 15-bit field is the binary equivalent of the INCSTP variable derived from the formula: $INCSTEP = round(((2^{15} - 1) \times md \times PLL1CR[MFD])/(100 \times 5 \times MODPRD))$ where md represents the peak modulation depth in percentage (+/- md for centered modulation, -2 x md for modulation below nominal frequency), and MFD represents the nominal value of the feedback loop divider.

24.5.2.8 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. 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: 0030h 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 172. PLL1 Fractional Divider Register (PLL1FD)

Table 238. 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 MSB = H disables the Rectangular PDF dither input to SDM LSB = H disables the Triangular PDF dither input to SDM
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 VCO clock frequency is given by: VCO frequency = InputClock x (PLL1DV[MFD] + (FRCDIV / 2^{12})) where FRCDIV is the decimal equivalent of the FRCDIV bits.

24.5.3 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 *Chapter 6: Functional safety*.

24.6 Functional description

This section explains the operation and configuration of the Dual PLLDIG module.

24.6.1 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 device datasheet.

24.6.2 Clock configuration

The relationship between input and output frequency is determined by programming the PLL0DV and PLL1DV registers, and calculated according to the following equations:

Equation 13

$$f_{pll0_phi} = \frac{f_{pll0_ref} \times (PLL0DV[MFD])}{PLL0DV[PREDIV] \times (PLL0DV[RFDPHI])}$$

Equation 14

$$f_{pll0_phi1} = \frac{f_{pll0_ref} \times PLL0DV[MFD]}{PLL0DV[PREDIV] \times PLL0DV[RFDPHI1]}$$

Equation 15

$$f_{pll1_phi} = \frac{f_{pll1_ref} \times PLL1DV[MFD]}{2 \times PLL1DV[RFDPHI]}$$

When PLL0s PHI1 is used as the reference for PLL1:

Equation 16

$$f_{pll1_phi} = \frac{f_{pll0_ref} \times PLL0DV[MFD] \times PLL1DV[MFD]}{PLL0DV[PREDIV] \times PLL0DV[RFDPHI1] \times (2 \times PLL1DV[RFDPHI])}$$

The relationship between the VCO frequency f_{VCO} and the output frequency of the PLLs is determined by the value of PLL1DV[RFDPHI], PLL0DV[RFDPHI] and PLL0DV[RFDPHI1], according to the following equations:

Equation 17

$$f_{pll0_VCO} = f_{pll0_phi} \times (2 \times PLL0DV[RFDPHI])$$

Equation 18

$$f_{pll1_VCO} = f_{pll1_phi} \times (2 \times PLL1DV[RFDPHI])$$

Equation 19

$$f_{pll0_VCO} = f_{pll0_phi1} \times (2 \times PLL0DV[RFDPHI1])$$

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. 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.

Upon any of these events 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.

Note: Each time the input reference clock, *PLL0DV[PREDIV], PLL0DV[MFD] or PLL1DV[MFD]* values are modified, a power down pulse is required to re-lock the analog circuitry of the PLLs.

The recommended procedure to program the PLLs and engage normal mode is shown in *Section 24.7*.

24.6.3 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 interrupt is 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.

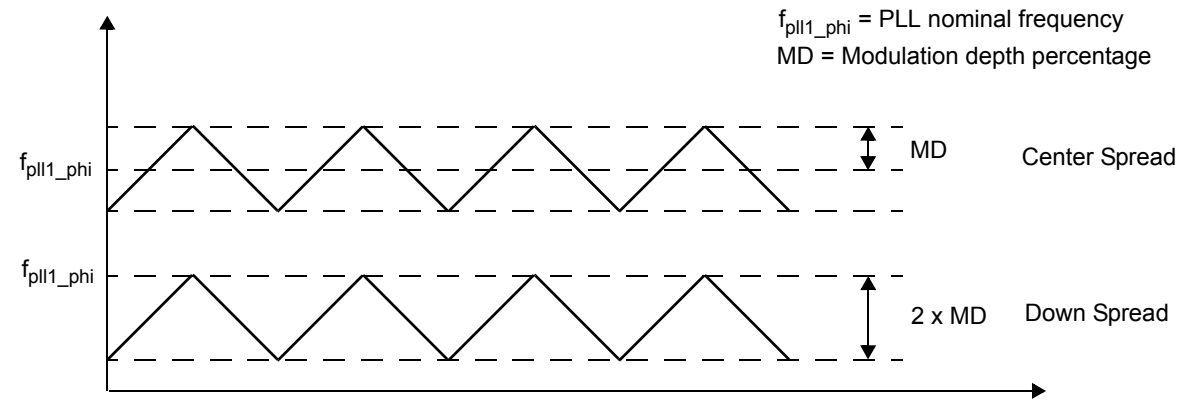
The lock indication from the analog PLL is synchronized and stored in the status register. When the analog PLL loses lock or regains lock, this is immediately indicated in the status register (after synchronization time).

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.

24.6.4 Frequency modulation

Frequency modulation uses a triangular profile as shown in *Figure 173*. The modulation frequency and depth are controlled using PLL1FM[MODPRD] and PLL1FM[INCSTP].

Figure 173. Triangular frequency modulation



Note: The device maximum operating frequency includes the frequency modulation. If center modulation is selected, the f_{sys} must be below f_{max} by MD percentage such that $f_{max} = f_{sys}(1 + MD\%)$.

The following equations define how to calculate PLL1FM[MODPRD] (*Equation 20*) and PLL1FM[INCSTP] (*Equation 21*) based on the frequency of the feedback divider (f_{ref}), the modulation frequency (f_{mod}) and the modulation depth percentage (MD):

Equation 20

$$\text{PLL1FM[MODPRD]} = \text{round}\left(\frac{f_{ref}}{4 \times f_{mod}}\right)$$

Equation 21

$$\text{PLL1FM[INCSTP]} = \text{round}\left(\frac{(2^{15} - 1) \times \text{MD} \times \text{PLL1DV[MFD]}}{100 \times 5 \times \text{PLL1FM[MODPRD]}}\right)$$

PLL1FM[MODPRD] and PLL1FM[INCSTP] are subject to the following restriction:

Equation 22

$$(\text{PLL1FM[MODPRD]} \times \text{PLL1FM[INCSTP]}) < 2^{15}$$

Because of the above rounding operations, the effective modulation depth applied to the FMPLL is given by the following formula:

Equation 23

$$\text{ModulationDepth} = \text{round}\left(\frac{\text{PLL1FM[MODPRD]} \times \text{PLL1FM[INCSTP]} \times 100 \times 5}{(2^{15} - 1) \times \text{PLL1DV[MFD]}}\right)$$

As an example, suppose the following configuration:

- Input frequency: 40 MHz
- Loop divider (PLL*n*CR[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[l]
2. Program the PLL1FM while PLL1 is powered down
3. Power up the PLL1[m]

24.7 Initialization information

Coming out of reset PLL0 and PLL1 are enabled. The recommended procedure to program the PLLs and enter normal mode is:

1. Program PLL0DV[PREDIV], PLL0DV[RFDPHI1], PLL0DV[MFD] and PLL0DV[RFDPHI].
2. Power up PLL0 (controlled by configuring the appropriate MC_ME_<*mode*>_MC register).
3. Poll PLL0SR[LOCK] until it asserts.
4. Program PLL1DV[MFD] and PLL1DV[RFDPHI].
5. 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.
6. Power up PLL1 (controlled by configuring the appropriate MC_ME_<*mode*>_MC register).
7. Poll PLL1SR[LOCK] until it asserts.
8. Engage normal mode.

l. By writing MC_ME_<*mode*>_MC[PLL1ON] = 0b.

m. By writing MC_ME_<*mode*>_MC[PLL1ON] = 1b.

At this point, the PLL is configured and is generating a output clock; however, the PLL is not the system clock until it is configured in the Mode Entry Module (MC_ME) and a mode change has taken place. See *Chapter 48: Mode Entry Module (MC_ME)* for more details.

25 Clock Monitor Unit (CMU)

25.1 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/CLKMT1/CLKMT2 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, CLKMT1, CLKMT2), 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 “Clocking” chapter for the number of instances of the CMU in this device.

25.1.1 Main features

- CLKMT0_RMN, CLKMT1, CLKMT2 frequency measurement with CLKMN0_RMT as the reference clock.
- Event generation for various failures detected inside monitoring unit.

25.2 Block diagram

The block diagram of the CMU module(s) is shown in the “Clocking" chapter of this Reference Manual.

25.3 Signals

Table 239 describes the signals on the boundary of the CMU (in alphabetical order).

Table 239. 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.
CLKMT1	I	Metered Clock Signal 1: Receives a clock signal that the CMU measures against a reference clock frequency.
CLKMT2	I	Metered Clock Signal 2: Receives a clock signal that the CMU measures against a reference clock frequency.

Note: See the “Clocking” chapter for device specific clock sources of each CMU.

25.4 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 240*.

Table 240. CMU memory map

Offset (hex)	Register	Location
0000	CMU Control Status Register (CMU_CSR)	*Section 25.4.1.1*
0004	CMU Frequency Display Register (CMU_FDR)	*Section 25.4.1.2*
0008	CMU High Frequency Reference Register CLKMN1 (CMU_HFREFR)	*Section 25.4.1.3*
000C	CMU Low Frequency Reference Register CLKMN1 (CMU_LFREFR)	*Section 25.4.1.4*
0010	CMU Interrupt Status Register (CMU_ISR)	*Section 25.4.1.5*
0014	Reserved	—
0018	CMU Measurement Duration Register (CMU_MDR)	*Section 25.4.1.6*

Note: See the “Clocking” chapter for register and field availability details.

25.4.1 Register descriptions

25.4.1.1 CMU Control Status Register (CMU_CSR)

Address: Base + 0000h 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 the “Clocking” chapter for device specific CMU implementation details.

Figure 174. CMU Control Status Register (CMU_CSR)

Table 241. 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. 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 CLKMT1 is selected 10 CLKMT2 is selected 11 CLKMT0_RMN is selected
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.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 0 CLKMN1 monitor is disabled 1 CLKMN1 monitor is enabled

25.4.1.2 CMU Frequency Display Register (CMU_FDR)

The FDR is used to determine the measured frequency being monitored by the CMU.

Address: Base + 0004h 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[19:16]			
W																
Reset	0	0	0	0	0	0	0	0	0	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[15:0]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 175. CMU Frequency Display Register (CMU_FDR)

Table 242. 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: $f_{sel} = (f_{CLKMN0_RMT} \times CMU_MDR[MD]) / CMU_FDR[FD]$

25.4.1.3 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 176* and *Table 243* shows the CMU_HFREFR register.

Address: Base + 0008h 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 176. CMU High Frequency Reference Register CLKMN1 (CMU_HFREFR)

Table 243. 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).

25.4.1.4 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 177* and *Table 244* shows the CMU_LFREFR register.

Address: Base + 000Ch 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 177. CMU Low Frequency Reference Register CLKMN1 (CMU_LFREFR)

Table 244. CMU_LFREFR 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).

25.4.1.5 CMU Interrupt Status Register (CMU_ISR)

Address: Base + 0010h 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	FLCI	FHHI	FLLI	OLRI(1)
W													w1c	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 the “Clocking” chapter for device specific CMU implementation details.
2.

Figure 178. CMU Interrupt Status Register (CMU_ISR)

Table 245. CMU_ISR field descriptions

Field	Description
28 FLCI	CLKMN1 frequency less than reference clock event status. This bit is set by hardware when CLKMN1 frequency becomes lower than reference clock frequency (f_{CLKMT0_RMN}/4) value and CLKMN1 is 'ON' as signaled by the MC_ME. It can be cleared by software by writing '1'. 0 No FLC event 1 FLC event occurred
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
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 While entering STOP mode, OLRI may be triggered. To avoid this when the system is running on the PLL, disable XOSC before entering STOP mode.

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.

25.4.1.6 CMU Measurement Duration Register (CMU_MDR)

Address: Base + 0018h 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[19:16]			
W																
Reset	0	0	0	0	0	0	0	0	0	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[15:0]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 179. CMU Measurement Duration Register (CMU_MDR)

Table 246. 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, CLKMT1, CLKMT2) cycles. This value is loaded in the frequency meter down-counter. The down-counter starts counting when CMU_CSR [SFM] = 1.

25.5 Functional description

This section describes the functionality of the CMU.

25.5.1 Frequency meter

The purpose of frequency meter is to evaluate the deviation from the nominal metered source (such as, CLKMT0_RMN, CLKMT2, CLKMT1) 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 24 $\mathbf{f_{sel} = (f_{CLKMN0_RMT} \times MDR[MD]) / FDR[FD]}$

25.5.2 CLKMN0_RMT supervisor

- The CMU writes 1 to CMU_ISR[OLRI].
- The CMU asserts the OLR signal.

Note: f_{CLKMN0_RMT} must be greater than $f_{CLKMT0_RMN} / 2^{RCDIV}$ by at least 0.5 MHz in order to guarantee correct f_{CLKMN0_RMT} monitoring.

25.5.3 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: The high 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.*

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).

Note: The CMU writes 1 to CMU_ISR[FLCI].

f_{CLKMN1} must be greater than f_{CLKMT0_RMN} / 4 by at least 0.5 MHz in order to guarantee correct CLKMN1 monitoring.

- 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.

26 Clock Generation Module (MC_CGM)

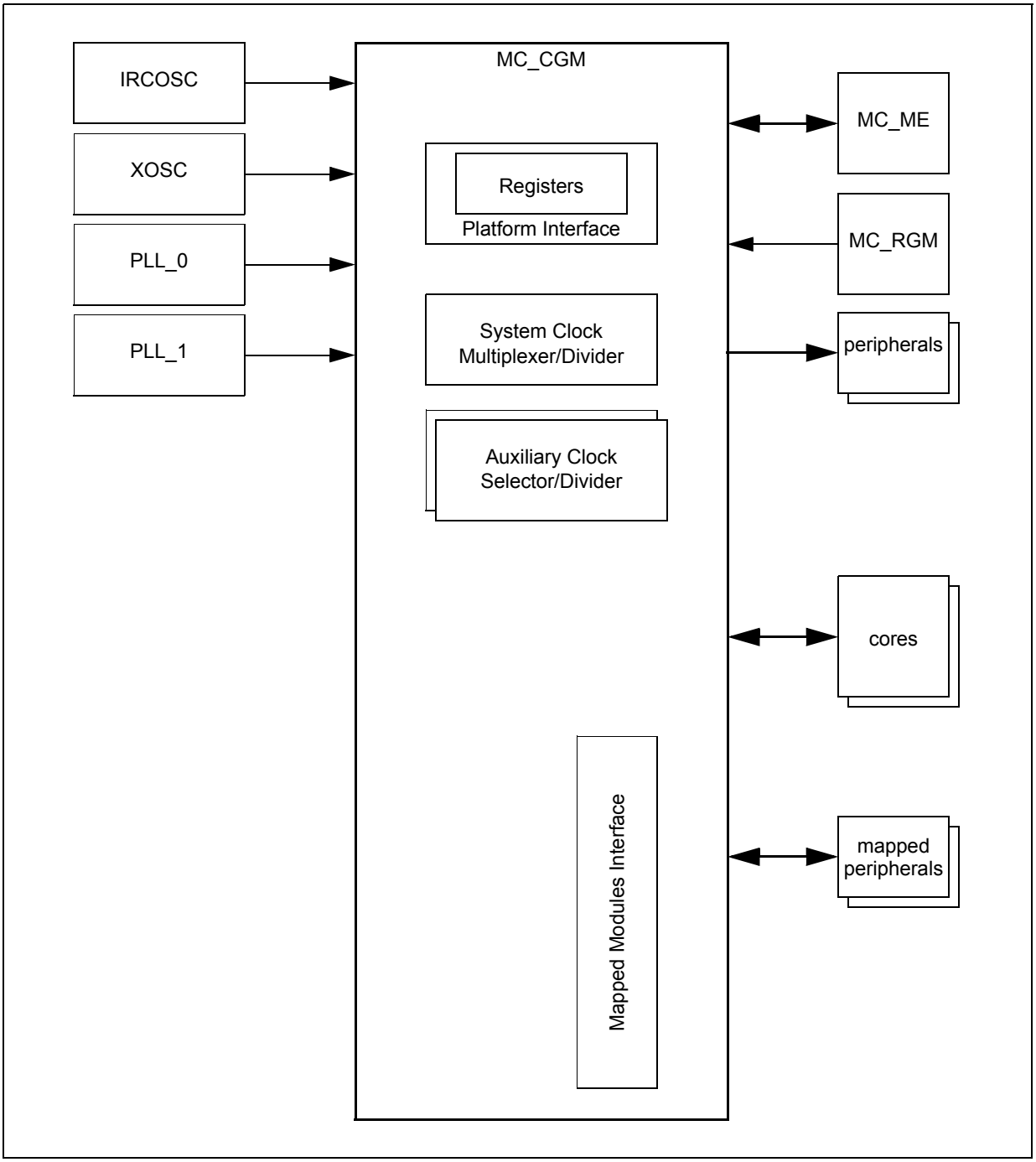
26.1 Introduction

26.1.1 Overview

The clock generation module (MC_CGM) generates reference clocks for all the modules on the chip. The MC_CGM selects one of the system clock sources to supply the system clock. The MC_ME controls the system clock selection (see *Chapter 48: Mode Entry Module (MC_ME)* for more details). The memory spaces of system and peripheral clock sources which have addressable memory spaces are accessed through the MC_CGM memory space.

The following figure shows the MC_CGM block diagram.

Figure 180. MC_CGM block diagram



26.1.2 Features

The MC_CGM includes the following features:

- 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-less clock transitions when changing the system clock selection
- supports 8, 16, and 32-bit wide read/write accesses

26.2 Memory map and register definition

Table 247. MC_CGM register description

Address offset	Name	Location
0x0000	PCS Switch Duration Register (CGM_PCS_SDUR)	*Section 26.2.1.1*
0x0004	PCS Divider Change Register 1 (CGM_PCS_DIVC1)	*Section 26.2.1.2*
0x0008	PCS Divider Start Register 1 (CGM_PCS_DIVS1)	*Section 26.2.1.3*
0x000C	PCS Divider End Register 1 (CGM_PCS_DIVE1)	*Section 26.2.1.4*
0x0010	PCS Divider Change Register 2 (CGM_PCS_DIVC2)	*Section 26.2.1.5*
0x0014	PCS Divider Start Register 2 (CGM_PCS_DIVS2)	*Section 26.2.1.6*
0x0018	PCS Divider End Register 2 (CGM_PCS_DIVE2)	*Section 26.2.1.7*
0x001C-0x0027	Reserved	
0x0028	PCS Divider Change Register 4 (CGM_PCS_DIVC4)	*Section 26.2.1.8*
0x002C	PCS Divider Start Register 4 (CGM_PCS_DIVS4)	*Section 26.2.1.9*
0x0030	PCS Divider End Register 4 (CGM_PCS_DIVE4)	*Section 26.2.1.10*
0x0034-0x00CF	Reserved	
0x00D0	System Clock Divider Ratio Change Register (CGM_SC_DIV_RC)	*Section 26.2.1.11*
0x00D4	Divider Update Type Register (CGM_DIV_UPD_TYPE)	*Section 26.2.1.12*
0x00D8	Divider Update Trigger Register (CGM_DIV_UPD_TRIG)	*Section 26.2.1.13*
0x00DC	Divider Update Status Register (CGM_DIV_UPD_STAT)	*Section 26.2.1.14*
0x00E0-0x00E3	Reserved	
0x00E4	System Clock Select Status Register (CGM_SC_SS)	*Section 26.2.1.15*
0x00E8	System Clock Divider 0 Configuration Register (CGM_SC_DC0)	*Section 26.2.1.16*
0x00EC	System Clock Divider 1 Configuration Register (CGM_SC_DC1)	*Section 26.2.1.17*
0x00F0	System Clock Divider 2 Configuration Register (CGM_SC_DC2)	*Section 26.2.1.18*

Table 247. MC_CGM register description(Continued)

Address offset	Name	Location
0x00F4-0x00FF	Reserved	
0x0100	Auxiliary Clock 0 Select Control Register (CGM_AC0_SC)	*Section 26.2.1.19*
0x0104	Auxiliary Clock 0 Select Status Register (CGM_AC0_SS)	*Section 26.2.1.20*
0x0108	Auxiliary Clock 0 Divider 0 Configuration Register (CGM_AC0_DC0)	*Section 26.2.1.21*
0x010C	Auxiliary Clock 0 Divider 1 Configuration Register (CGM_AC0_DC1)	*Section 26.2.1.22*
0x0110	Auxiliary Clock 0 Divider 2 Configuration Register (CGM_AC0_DC2)	*Section 26.2.1.23*
0x0114	Auxiliary Clock 0 Divider 3 Configuration Register (CGM_AC0_DC3)	*Section 26.2.1.24*
0x0118	Auxiliary Clock 0 Divider 4 Configuration Register (CGM_AC0_DC4)	*Section 26.2.1.25*
0x011C	Auxiliary Clock 0 Divider 5 Configuration Register (CGM_AC0_DC5)	*Section 26.2.1.26*
0x0120	Auxiliary Clock 1 Select Control Register (CGM_AC1_SC)	*Section 26.2.1.27*
0x0124	Auxiliary Clock 1 Select Status Register (CGM_AC1_SS)	*Section 26.2.1.28*
0x0128	Auxiliary Clock 1 Divider 0 Configuration Register (CGM_AC1_DC0)	*Section 26.2.1.29*
0x012C-0x013F	Reserved	
0x0140	Auxiliary Clock 2 Select Control Register (CGM_AC2_SC)	*Section 26.2.1.30*
0x0144	Auxiliary Clock 2 Select Status Register (CGM_AC2_SS)	*Section 26.2.1.31*
0x0148-0x015F	Reserved	
0x0160	Auxiliary Clock 3 Select Control Register (CGM_AC3_SC)	*Section 26.2.1.32*
0x0164	Auxiliary Clock 3 Select Status Register (CGM_AC3_SS)	*Section 26.2.1.33*

Note: Any access to unused registers as well as write accesses to read-only registers will :

- *Not change register content*
- *Cause a transfer error*

26.2.1 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.

26.2.1.1 PCS Switch Duration Register (CGM_PCS_SDUR)

This register contains the progressive system clock switching duration for each step. See *Section 26.3.1.2: 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	8	9	10	11	12	13	14	15
R	SDUR								0	0	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 181. PCS Switch Duration Register (CGM_PCS_SDUR)

Table 248. 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 cycles.

26.2.1.2 PCS Divider Change Register 1 (CGM_PCS_DIVC1)

This register defines the rate of frequency change and initial change value for the progressive system clock switching when switching the system clock source to or from the external crystal oscillator (XOSC) on ramp-up and ramp-down, respectively. See *Section 26.3.1.2: Progressive system clock switching*, for details on how to set these values.

Note: Do not attempt a byte write access to the INIT field of this register. Such an access does 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	INIT															
W																
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 182. PCS Divider Change Register 1 (CGM_PCS_DIVC1)

Table 249. CGM_PCS_DIVC1 field descriptions

Field	Description
0:15 INIT	Divider Change Initial Value This is initial change value of the clock divider for the clock ramp-up phase when switching to the external crystal oscillator (XOSC).
24:31 RATE	Divider Change Rate This value controls the change value of the clock divider for the clock ramp-up and ramp-down phase when switching to/from the external crystal oscillator (XOSC).

26.2.1.3 PCS Divider Start Register 1 (CGM_PCS_DIVS1)

This register defines the initial division value for the progressive system clock switching when switching the system clock source to the external crystal oscillator (XOSC) on ramp-up. Register *n* corresponds to system clock source *n*. See *Section 26.3.1.2: Progressive system clock switching*, for details on how to set these values.

Note: Do not attempt a byte or an half-word write accesses to this register. Such an access does 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	DIVS[19:16]			
W																
Reset	0	0	0	0	0	0	0	0	0	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[15:0]															
W																
Reset	0	0	0	0	0	0	1	1	1	1	1	0	0	1	1	1

Figure 183. PCS Divider Start Register 1 (CGM_PCS_DIVS1)

Table 250. CGM_PCS_DIVS1 field descriptions

Field	Description
12:31 DIVS	Divider Start Value This is the start value of the clock divider for the clock ramp-up phase when switching to the external crystal oscillator (XOSC).

26.2.1.4 PCS Divider End Register 1 (CGM_PCS_DIVE1)

This register defines the final division value for the progressive system clock switching when switching the system clock source from the external crystal oscillator (XOSC) on ramp-down. See *Section 26.3.1.2: Progressive system clock switching*, for details on how to set this value.

Note: Do not attempt a byte or an half-word write accesses to this register. Such an access does 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	DIVE[19:16]			
W																
Reset	0	0	0	0	0	0	0	0	0	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[15:0]															
W																
Reset	0	0	0	0	0	0	1	1	1	1	1	0	0	1	1	1

Figure 184. PCS Divider End Register 1 (CGM_PCS_DIVE1)

Table 251. CGM_PCS_DIVE1 field descriptions

Field	Description
12:31 DIVE	Divider End Value This is the clock divider end value for the clock ramp-down phase when switching from the external crystal oscillator (XOSC).

26.2.1.5 PCS Divider Change Register 2 (CGM_PCS_DIVC2)

This register defines the rate of frequency change and initial change value for the progressive system clock switching when switching the system clock source to or from the PLL_0 on ramp-up and ramp-down, respectively. See *Section 26.3.1.2: Progressive system clock switching*, for details on how to set these values.

Note: Do not attempt a byte write access to the INIT field of this register. Such an access does 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	INIT															
W																
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 185. PCS Divider Change Register 2 (CGM_PCS_DIVC2)

Table 252. CGM_PCS_DIVC2 field descriptions

Field	Description
0:15 INIT	Divider Change Initial Value This is initial change value of the clock divider for the clock ramp-up phase when switching to the PLL_0.
24:31 RATE	Divider Change Rate This value controls the change value of the clock divider for the clock ramp-up and ramp-down phase when switching to/from the PLL_0.

26.2.1.6 PCS Divider Start Register 2 (CGM_PCS_DIVS2)

This register defines the initial division value for the progressive system clock switching when switching the system clock source to the PLL_0 on ramp-up. See *Section 26.3.1.2: Progressive system clock switching*, for details on how to set this value.

Note: Do not attempt a byte or an half-word write accesses to this register. Such an access does 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	DIVS[19:16]			
W																
Reset	0	0	0	0	0	0	0	0	0	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[15:0]															
W																
Reset	0	0	0	0	0	0	1	1	1	1	1	0	0	1	1	1

Figure 186. PCS Divider Start Register 2 (CGM_PCS_DIVS2)

Table 253. CGM_PCS_DIVS2 field descriptions

Field	Description
12:31 DIVS	Divider Start Value This is the start value of the clock divider for the clock ramp-up phase when switching to the PLL_0.

26.2.1.7 PCS Divider End Register 2 (CGM_PCS_DIVE2)

This register defines the final division value for the progressive system clock switching when switching the system clock source from the PLL_0 on ramp-down. See *Section 26.3.1.2: Progressive system clock switching* for details on how to set this value.

Note: Do not attempt a byte or an half-word write accesses to this register. Such an access does 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	DIVE[19:16]			
W																
Reset	0	0	0	0	0	0	0	0	0	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[15:0]															
W																
Reset	0	0	0	0	0	0	1	1	1	1	1	0	0	1	1	1

Figure 187. PCS Divider End Register 2 (CGM_PCS_DIVE2)

Table 254. CGM_PCS_DIVE2 field descriptions

Field	Description
12:31 DIVE	Divider End Value This is the clock divider end value for the clock ramp-down phase when switching from the PLL_0.

26.2.1.8 PCS Divider Change Register 4 (CGM_PCS_DIVC4)

This register defines the rate of frequency change and initial change value for the progressive system clock switching when switching the system clock source to or from the PLL_1 on ramp-up and ramp-down, respectively. See *Section 26.3.1.2: Progressive system clock switching*, for details on how to set these values.

Note: Do not attempt a byte write access to the INIT field of this register. Such an access does 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	INIT															
W																
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 188. PCS Divider Change Register 4 (CGM_PCS_DIVC4)

Table 255. CGM_PCS_DIVC4 field descriptions

Field	Description
0:15 INIT	Divider Change Initial Value This is initial change value of the clock divider for the clock ramp-up phase when switching to the PLL_1.
24:31 RATE	Divider Change Rate This value controls the change value of the clock divider for the clock ramp-up and ramp-down phase when switching to/from the PLL_1.

26.2.1.9 PCS Divider Start Register 4 (CGM_PCS_DIVS4)

This register defines the initial division value for the progressive system clock switching when switching the system clock source to the PLL_1 on ramp-up. See *Section 26.3.1.2: Progressive system clock switching* for details on how to set this value.

Note: Do not attempt a byte or an half-word write accesses to this register. Such an access does 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	DIVS[19:16]			
W																
Reset	0	0	0	0	0	0	0	0	0	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[15:0]															
W																
Reset	0	0	0	0	0	0	1	1	1	1	1	0	0	1	1	1

Figure 189. PCS Divider Start Register 4 (CGM_PCS_DIVS4)

Table 256. CGM_PCS_DIVS4 field descriptions

Field	Description
12:31 DIVS	Divider Start Value This is the start value of the clock divider for the clock ramp-up phase when switching to the PLL_1.

26.2.1.10 PCS Divider End Register 4 (CGM_PCS_DIVE4)

This register defines the final division value for the progressive system clock switching when switching the system clock source from the PLL_1 on ramp-down. See *Section 26.3.1.2: Progressive system clock switching*, for details on how to set this value.

Note: Do not attempt a byte or an half-word write accesses to this register. Such an access does 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	DIVE[19:16]			
W																
Reset	0	0	0	0	0	0	0	0	0	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[15:0]															
W																
Reset	0	0	0	0	0	0	1	1	1	1	1	0	0	1	1	1

Figure 190. PCS Divider End Register 4 (CGM_PCS_DIVE4)

Table 257. CGM_PCS_DIVE4 field descriptions

Field	Description
12:31 DIVE	Divider End Value This is the clock divider end value for the clock ramp-down phase when switching from the PLL_1.

26.2.1.11 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 191. CGM_SC_DIV_RC

Table 258. 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

26.2.1.12 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	0	0	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 192. CGM_DIV_UPD_TYPE

Table 259. 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
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 1divider 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

26.2.1.13 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															
Reset	0	0	0	0	0	0	0	0	0	0	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															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 193. CGM_DIV_UPD_TRIG

Table 260. 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.

26.2.1.14 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	0	0	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 194. CGM_DIV_UPD_STAT

Table 261. 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
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

26.2.1.15 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 195. System Clock Select Status Register (CGM_SC_SS)

Table 262. 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. 0000 16 MHz internal RC oscillator 0001 external crystal oscillator (XOSC) 0010 PLL_0 0011 reserved 0100 PLL_1 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

Table 262. CGM_SC_SS field descriptions(Continued)

Field	Description
12:14 SWTRG	Switch Trigger cause This value indicates the cause for the latest clock source switch. 000 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 16 MHz internal RC oscillator due to SAFE mode request or reset succeeded 101 switch to 16 MHz internal RC oscillator due to SAFE mode request or reset, but current clock source was inactive 110 reserved 111 reserved
15 SWIP	Switch In Progress 0 clock source switching has completed 1 clock source switching is in progress

26.2.1.16 System Clock Divider 0 Configuration Register (CGM_SC_DC0)

This register controls system clock divider 0. See *Section 26.3.1.4.1: System clock divider synchronization*, for details on division value limitations.

Note: Byte and half-word write accesses are not allowed for this register. Such an access will not result in an exception; however, the value will not be loaded with the new value.

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 196. System Clock Divider 0 Configuration Register (CGM_SC_DC0)

Table 263. CGM_SC_DC0 field descriptions

Field	Description
0 DE	Divider 0 Enable This divider is always enabled. Therefore, this bit is read-only and always returns a value of ‘1’.
13:15 DIV	Divider 1 Division Value The resultant XBAR clock will have a period ‘DIV + 1’ times that of the system clock. If DE is set to ‘0’ (divider 0 is disabled), any write access to the DIV field is ignored and the XBAR clock remains disabled.

26.2.1.17 System Clock Divider 1 Configuration Register (CGM_SC_DC1)

This register controls system clock divider 1. See *Section 26.3.1.4.1: System clock divider synchronization*, for details on division value limitations.

Note: Do not attempt a byte or an half-word write accesses to this register. Such an access does 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 197. System Clock Divider 1 Configuration Register (CGM_SC_DC1)

Table 264. CGM_SC_DC1 field descriptions

Field	Description
0 DE	Divider 1 Enable This divider is always enabled. Therefore, this bit is read-only and always returns a value of ‘1’.
12:15 DIV	Divider 1 Division Value The resultant PBRIDGE Clock will have a period ‘DIV + 1’ times that of the system clock. If DE is set to ‘0’ (divider 1 is disabled), any write access to the DIV field is ignored and the PBRIDGE Clock remains disabled.

26.2.1.18 System Clock Divider 2 Configuration Register (CGM_SC_DC2)

This register controls system clock divider 2. See *Section 26.3.1.4.1: System clock divider synchronization*, for details on division value limitations.

Note: Byte and half-word write accesses are not allowed for this register. Such an access will not result in an exception; however, the value will not be loaded with the new value.

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	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 198. System Clock Divider 2 Configuration Register (CGM_SC_DC2)

Table 265. CGM_SC_DC2 field descriptions

Field	Description
0 DE	Divider Enable 0 Disable system clock divider 2 1 Enable system clock divider 2
8:15 DIV	Divider 1 Division Value The resultant CLOCKOUT will have a period ‘DIV + 1’ times that of the system clock. If DE is set to ‘0’ (divider 2 is disabled), any write access to the DIV field is ignored and the CLOCKOUT remains disabled.

26.2.1.19 Auxiliary Clock 0 Select Control Register (CGM_AC0_SC)

This register is used to select the current clock source for the following clocks:

- undivided: (unused)
- divided by auxiliary clock 0 divider 0: PIT clock
- divided by auxiliary clock 0 divider 1: SAR ADC clock
- divided by auxiliary clock 0 divider 2: CTU clock
- divided by auxiliary clock 0 divider 3: DSPI clock
- divided by auxiliary clock 0 divider 4: LINFlexD clock
- divided by auxiliary clock 0 divider 5: eTimer clock

See *Figure 217* 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 199. Auxiliary Clock 0 Select Control Register (CGM_AC0_SC)

Table 266. CGM_AC0_SC field descriptions

Field	Description
4:7 SELCTL	Auxiliary Clock 0 Source Selection Control Selects the source for auxiliary clock 0. 0000 16 MHz internal RC oscillator 0001 external crystal oscillator 0010 PLL_0 0011 reserved 0100 reserved 0101 reserved 0110 reserved 0111 reserved 1000 reserved 1001 reserved 1010 reserved 1011 reserved 1100 reserved 1101 reserved 1110 reserved 1111 reserved

26.2.1.20 Auxiliary Clock 0 Select Status Register (CGM_AC0_SS)

This register provides the current auxiliary clock 0 source selection.

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 200. Auxiliary Clock 0 Select Status Register (CGM_AC0_SS)

Table 267. CGM_AC0_SS field descriptions

Field	Description
4:7 SELSTAT	Auxiliary Clock 0 Source Selection Status This value indicates the current source for auxiliary clock 0. 0000 16 MHz internal RC oscillator 0001 external crystal oscillator 0010 PLL_0 0011 reserved 0100 reserved 0101 reserved 0110 reserved 0111 reserved 1000 reserved 1001 reserved 1010 reserved 1011 reserved 1100 reserved 1101 reserved 1110 reserved 1111 reserved

26.2.1.21 Auxiliary Clock 0 Divider 0 Configuration Register (CGM_AC0_DC0)

This register controls auxiliary clock 0 divider 0.

Note: Do not attempt a byte or an half-word write accesses to this register. Such an access does 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 201. Auxiliary Clock 0 Divider 0 Configuration Register (CGM_AC0_DC0)

Table 268. CGM_AC0_DC0 field descriptions

Field	Description
0 DE	Divider Enable 0 Disable auxiliary clock 0 divider 0 1 Enable auxiliary clock 0 divider 0
12:15 DIV	Divider Division Value The resultant PIT clock will have a period ‘DIV + 1’ times that of auxiliary clock 0. If DE is set to 0 (divider 0 is disabled), any write access to the DIV field is ignored and the PIT clock remains disabled.

26.2.1.22 Auxiliary Clock 0 Divider 1 Configuration Register (CGM_AC0_DC1)

This register controls auxiliary clock 0 divider 1.

Note: Do not attempt a byte or an half-word write accesses to this register. Such an access does 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 202. Auxiliary Clock 0 Divider 1 Configuration Register (CGM_AC0_DC1)

Table 269. CGM_AC0_DC1 field descriptions

Field	Description
0 DE	Divider Enable 0 Disable auxiliary clock 0 divider 1 1 Enable auxiliary clock 0 divider 1
9:15 DIV	Divider Division Value The resultant SAR ADC clock will have a period 'DIV + 1' times that of auxiliary clock 0. If DE is set to 0 (divider 1 is disabled), any write access to the DIV field is ignored and the SAR ADC clock remains disabled.

26.2.1.23 Auxiliary Clock 0 Divider 2 Configuration Register (CGM_AC0_DC2)

This register controls auxiliary clock 0 divider 2.

Note: Do not attempt a byte or an half-word write accesses to this register. Such an access does 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	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 203. Auxiliary Clock 0 Divider 2 Configuration Register (CGM_AC0_DC2)

Table 270. CGM_AC0_DC2 field descriptions

Field	Description
0 DE	Divider Enable 0 Disable auxiliary clock 0 divider 2 1 Enable auxiliary clock 0 divider 2
9:15 DIV	Divider Division Value The resultant CTU clock will have a period 'DIV + 1' times that of auxiliary clock 0. If DE is set to 0 (divider 2 is disabled), any write access to the DIV field is ignored and the CTU clock remains disabled.

26.2.1.24 Auxiliary Clock 0 Divider 3 Configuration Register (CGM_AC0_DC3)

This register controls auxiliary clock 0 divider 3.

Note: Do not attempt a byte or an half-word write accesses to this register. Such an access does not result in an exception; however, the new value is not loaded.

Address: 0x0114 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 204. Auxiliary Clock 0 Divider 3 Configuration Register (CGM_AC0_DC3)

Table 271. CGM_AC0_DC3 field descriptions

Field	Description
0 DE	Divider Enable 0 Disable auxiliary clock 0 divider 3 1 Enable auxiliary clock 0 divider 3
9:15 DIV	Divider Division Value The resultant DSPI clock will have a period ‘DIV + 1’ times that of auxiliary clock 0. If DE is set to 0 (divider 3 is disabled), any write access to the DIV field is ignored and the DSPI clock remains disabled.

26.2.1.25 Auxiliary Clock 0 Divider 4 Configuration Register (CGM_AC0_DC4)

This register controls auxiliary clock 0 divider 4.

Note: Do not attempt a byte or an half-word write accesses to this register. Such an access does not result in an exception; however, the new value is not loaded.

Address: 0x0118 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 205. Auxiliary Clock 0 Divider 4 Configuration Register (CGM_AC0_DC4)

Table 272. CGM_AC0_DC4 field descriptions

Field	Description
0 DE	Divider Enable 0 Disable auxiliary clock 0 divider 4 1 Enable auxiliary clock 0 divider 4
12:15 DIV	Divider Division Value The resultant LINFlexD clock will have a period 'DIV + 1' times that of auxiliary clock 0. If DE is set to 0 (divider 4 is disabled), any write access to the DIV field is ignored and the LINFlexD clock remains disabled.

26.2.1.26 Auxiliary Clock 0 Divider 5 Configuration Register (CGM_AC0_DC5)

This register controls auxiliary clock 0 divider 5.

Note: Do not attempt a byte or an half-word write accesses to this register. Such an access does not result in an exception; however, the new value is not loaded.

Address: 0x011C 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 206. Auxiliary Clock 0 Divider 5 Configuration Register (CGM_AC0_DC5)

Table 273. CGM_AC0_DC5 field descriptions

Field	Description
0 DE	Divider Enable 0 Disable auxiliary clock 0 divider 5 1 Enable auxiliary clock 0 divider 5
12:15 DIV	Divider Division Value The resultant eTimer clock will have a period 'DIV + 1' times that of auxiliary clock 0. If DE is set to 0 (divider 5 is disabled), any write access to the DIV field is ignored and the eTimer clock remains disabled.

26.2.1.27 Auxiliary Clock 1 Select Control Register (CGM_AC1_SC)

This register is used to select the current clock source for the following clocks:

- undivided: (unused)
- divided by auxiliary clock 1 divider 0: FlexCAN clock

See *Figure 218* 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	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	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 207. Auxiliary Clock 1 Select Control Register (CGM_AC1_SC)

Table 274. CGM_AC1_SC field descriptions

Field	Description
4:7 SELCTL	Auxiliary Clock 1 Source Selection Control This value selects the current source for auxiliary clock 1. 0000 reserved 0001 external crystal oscillator 0010 PLL_0 0011 reserved 0100 reserved 0101 reserved 0110 reserved 0111 reserved 1000 reserved 1001 reserved 1010 reserved 1011 reserved 1100 reserved 1101 reserved 1110 reserved 1111 reserved

26.2.1.28 Auxiliary Clock 1 Select Status Register (CGM_AC1_SS)

This register provides the current auxiliary clock 1 source selection.

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	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	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 208. Auxiliary Clock 1 Select Status Register (CGM_AC1_SS)

Table 275. CGM_AC1_SS field descriptions

Field	Description
4:7 SELSTAT	Auxiliary Clock 1 Source Selection Status This value indicates the current source for auxiliary clock 1. 0000 reserved 0001 external crystal oscillator 0010 PLL_0 0011 reserved 0100 reserved 0101 reserved 0110 reserved 0111 reserved 1000 reserved 1001 reserved 1010 reserved 1011 reserved 1100 reserved 1101 reserved 1110 reserved 1111 reserved

26.2.1.29 Auxiliary Clock 1 Divider 0 Configuration Register (CGM_AC1_DC0)

This register controls auxiliary clock 1 divider 0.

Note: Do not attempt a byte or an half-word write accesses to this register. Such an access does 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 209. Auxiliary Clock 1 Divider 0 Configuration Register (CGM_AC1_DC0)

Table 276. CGM_AC1_DC0 field descriptions

Field	Description
0 DE	Divider Enable 0 Disable auxiliary clock 1 divider 0 1 Enable auxiliary clock 1 divider 0
10:15 DIV	Divider Division Value The resultant FlexCAN clock will have a period ‘DIV + 1’ times that of auxiliary clock 1. If DE is set to 0 (divider 0 is disabled), any write access to the DIV field is ignored and the FlexCAN clock remains disabled.

26.2.1.30 Auxiliary Clock 2 Select Control Register (CGM_AC2_SC)

This register is used to select the current clock source for the PLL_0 Reference clock.

See *Figure 219* for details.

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 210. Auxiliary Clock 2Select Control Register (CGM_AC2_SC)

Table 277. CGM_AC2_SC field descriptions

Field	Description
4:7 SELCTL	Auxiliary Clock 2 Source Selection Control This value selects the current source for auxiliary clock 2. 0000 16 MHz internal RC osc. 0001 external crystal oscillator 0010 reserved 0011 reserved 0100 reserved 0101 reserved 0110 reserved 0111 reserved 1000 reserved 1001 reserved 1010 reserved 1011 reserved 1100 reserved 1101 reserved 1110 reserved 1111 reserved

26.2.1.31 Auxiliary Clock 2 Select Status Register (CGM_AC2_SS)

This register provides the current auxiliary clock 2 source selection.

Address: 0x0144 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 211. Auxiliary Clock 2 Select Status Register (CGM_AC2_SS)

Table 278. CGM_AC2_SS field descriptions

Field	Description
4:7 SELSTAT	Auxiliary Clock 2 Source Selection Status This value indicates the current source for auxiliary clock 2. 0000 16 MHz internal RC osc. 0001 external crystal oscillator 0010 reserved 0011 reserved 0100 reserved 0101 reserved 0110 reserved 0111 reserved 1000 reserved 1001 reserved 1010 reserved 1011 reserved 1100 reserved 1101 reserved 1110 reserved 1111 reserved

26.2.1.32 Auxiliary Clock 3 Select Control Register (CGM_AC3_SC)

This register is used to select the current clock source for the PLL_1 reference clock.

See *Figure 220* 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	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	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 212. Auxiliary Clock 3 Select Control Register (CGM_AC3_SC)

Table 279. CGM_AC3_SC field descriptions

Field	Description
4:7 SELCTL	Auxiliary Clock 3 Source Selection Control This value selects the current source for auxiliary clock 3. 0000 reserved 0001 external crystal oscillator 0010 reserved 0011 PLL_0 0100 reserved 0101 reserved 0110 reserved 0111 reserved 1000 reserved 1001 reserved 1010 reserved 1011 reserved 1100 reserved 1101 reserved 1110 reserved 1111 reserved

26.2.1.33 Auxiliary Clock 3 Select Status Register (CGM_AC3_SS)

This register provides the current auxiliary clock 3 source selection.

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	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	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 213. Auxiliary Clock 3Select Status Register (CGM_AC3_SS)

Table 280. CGM_AC3_SS field descriptions

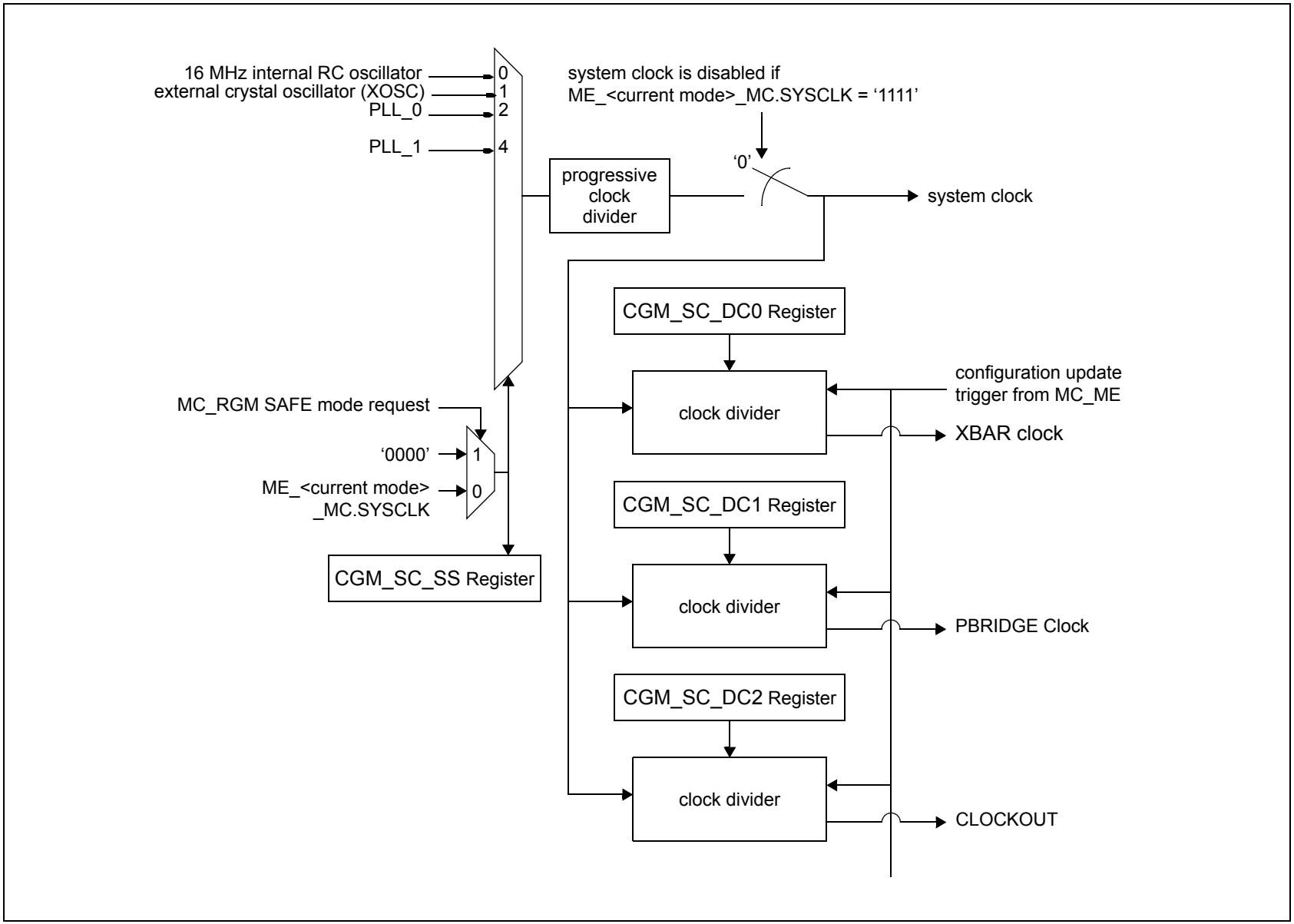
Field	Description
4:7 SELSTAT	Auxiliary Clock 3 Source Selection Status This value indicates the current source for auxiliary clock 3. 0000 reserved 0001 external crystal oscillator 0010 reserved 0011 PLL_0 0100 reserved 0101 reserved 0110 reserved 0111 reserved 1000 reserved 1001 reserved 1010 reserved 1011 reserved 1100 reserved 1101 reserved 1110 reserved 1111 reserved

26.3 Functional description

26.3.1 System clock generation

Figure 214 shows the block diagram of the system clock generation logic. The MC_ME provides the system clock select and switch mask (see *Chapter 48: Mode Entry Module (MC_ME)* for more details), and the MC_RGM provides the safe clock request (see *Chapter 43: Reset Generation Module (MC_RGM)* for more details). The safe clock request forces the selector to select the 16 MHz internal RC oscillator as the system clock and to ignore the system clock select.

Figure 214. MC_CGM system clock generation overview



26.3.1.1 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

26.3.1.2 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 down and/or up 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.

26.3.1.2.1 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 25*.

Equation 25

$$a_{max} = \frac{f_{chg}}{f_{src}}$$

26.3.1.2.2 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 16 MHz internal RC oscillator. After the expiration of this time, the module changes the clock divider value which changes the frequency of the system clock.

26.3.1.2.3 Configuration of CGM_PCS_DIVC*n*[RATE]

For a given maximum allowed frequency change rate amax the value d to be programmed into the PCS_DIVCn[RATE] register is given in *Table 281*.

Table 281. MC_CGM CGM_PCS_DIVC*n*[RATE] values

a_{max}	d = CGM_PCS_DIVC*n*[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

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 26*.

Equation 26

$$k = \left\lceil 0.5 + \sqrt{0.25 - \frac{2\left(1 - \frac{f_{src}}{16MHz}\right)}{d}} \right\rceil$$

26.3.1.2.4 Clock ramp-down

The clock ramp-down starts with the divider value 1 and with the given divider increment value PCD_DIVCn[RATE] and ends with the given divider value PCS_DIVEn[DIVE].

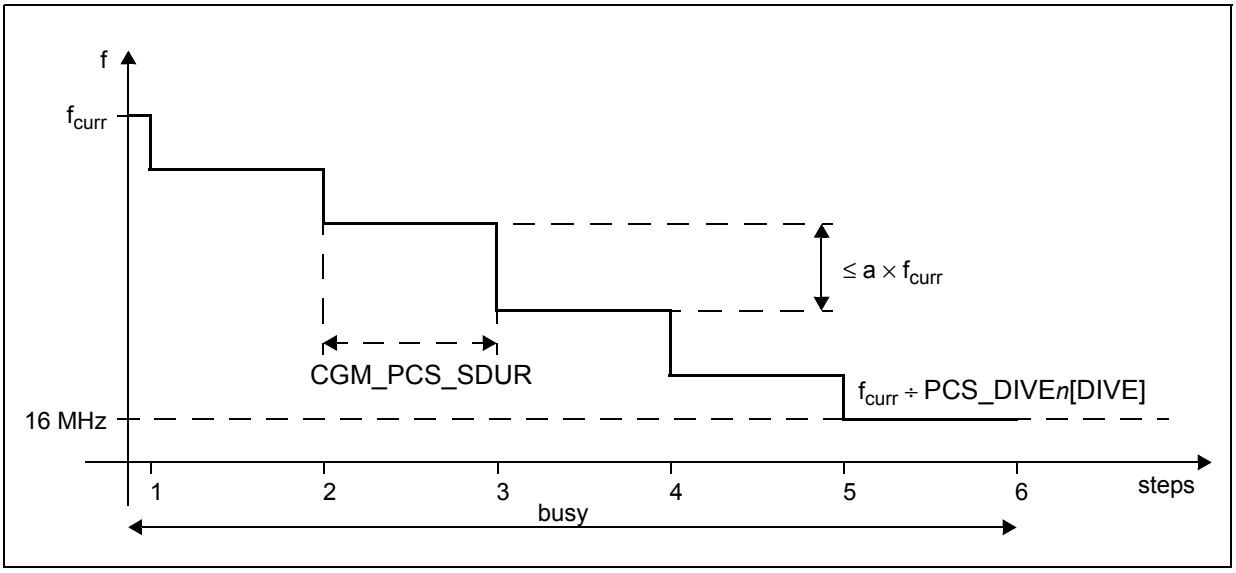
The divider end value for clock ramp-down is given in *Equation 27*.

Equation 27

$$\textbf{PCS_DIVEn[DIVE]} = \frac{f_{src}}{16MHz} \times 1000 - 1$$

Where f_{curr} is the frequency of the currently selected system clock source.

Figure 215. MC_CGM System Clock Ramp-Down Timing (k = 6 example)



26.3.1.2.5 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 28*.

Equation 28

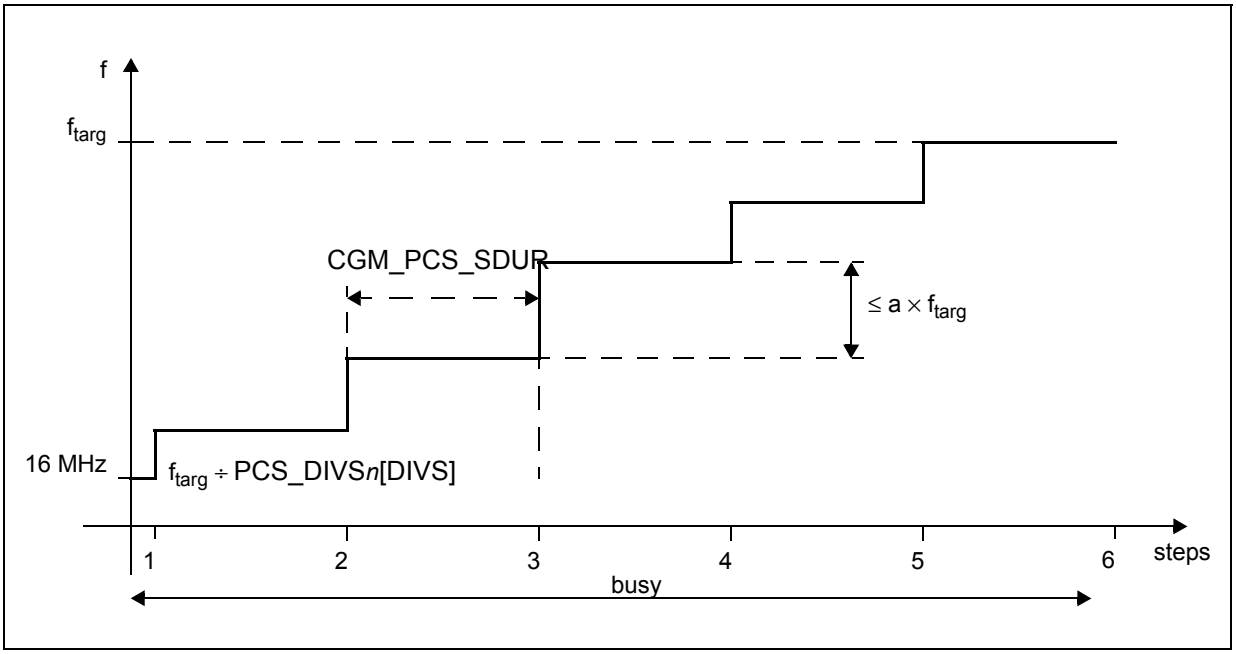
$$\textbf{PCS_DIVCn[INIT]} = d \times k \times 1000 - 1$$

Where k is calculated from *Equation 26* using the target system clock source frequency f_{targ}. The divider start value for clock ramp-up is given in *Equation 29*.

Equation 29

$$\textbf{PCS_DIVSn[DIVS]} = \left(1 + d\frac{k(k+1)}{2}\right) \times 1000 - 1$$

Figure 216. MC_CGM System Clock Ramp-Up Timing (k = 6 example)



26.3.1.3 System clock disable

During the TEST mode, the system clock can be disabled by the MC_ME.

26.3.1.4 System clock dividers

MC_CGM generates divider enables for the following clocks:

- XBAR clock: controlled by the CGM_SC_DC0 register
- PBRIDGE Clock: controlled by the CGM_SC_DC1 register
- CLOCKOUT: controlled by the CGM_SC_DC2 register

26.3.1.4.1 System clock divider synchronization

The system clock dividers are synchronized to each other such that the rising edges of the lower frequency clocks are aligned with those of the higher frequency clocks. This, however, imposes limitations on the division factors which can be used. In order for the synchronization to work properly, each division factor must be selected such that its value is an integer multiple of each division factor that has a lower value.

Table 282 shows an example of a correct divider configuration.

Example 1 Correct system clock divider configuration:

Table 282. MC_CGM Example System Clock Division Values Compatible with Divider Synchronization

Divider	Division factor	Register value
0	1	CGM_SC_DC0[DIV] = 0
1	2	CGM_SC_DC1[DIV] = 1
2	6	CGM_SC_DC2[DIV] = 5

In this case, system clock divider 1 has a division factor greater than that of system clock divider 0. Since 2 is an integer multiple of 1, its configuration is correct.

System clock divider 2 has a division factor greater than dividers 0 and 1; 6 is an integer multiple of 1 and 2.

Caution: Configuring the system clock dividers incorrectly will cause 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.
The dividers should be programmed as such to achieve ratio between XBAR and PBRIDGE to be 1:1 or 2:1. PBRIDGE frequency cannot exceed 40MHZ. Any ratio greater than 2:1 is not supported.

Note: During configuration (that is moving from one dividers configuration to another) is allowed to have a ratio greater than 2:1 with the constraint that in the meanwhile no registers access has to be performed on DMA module.

26.3.1.4.2 System clock divider update triggering

As opposed to the auxiliary clock dividers, which are updated immediately after a configuration change, the system clock dividers are updated only after the next software triggered mode change. This procedure ensures that the system clock dividers are updated simultaneously and that no intermediate inconsistent configurations can occur. It also ensures that the divider configurations are aligned with the target mode's system clock selection.

The system clock divider configuration update sequence is as follows:

1. Configure the target mode in the MC_ME to select the desired system clock source
2. Configure CGM_SC_DC0 to CGM_SC_DC2 registers as desired and aligned with the target system clock source frequency making sure that the dividers are consistently configured as described in *Section 26.3.1.4.1: System clock divider synchronization*
3. Request a mode change in the MC_ME to the desired target mode
4. When both mode changing and system clock switching are successfully completed, the new system clock divider configuration takes effect

See *Chapter 48: Mode Entry Module (MC_ME)*, for details on how to configure modes and how to request mode change.

26.3.2 Auxiliary clock generation

Figure 217 through *Figure 220* show the block diagrams of the generation logic for the various auxiliary clocks. See the following sections for auxiliary clock selection control:

- *Section 26.2.1.19: Auxiliary Clock 0 Select Control Register (CGM_AC0_SC)*
- *Section 26.2.1.27: Auxiliary Clock 1 Select Control Register (CGM_AC1_SC)*
- *Section 26.2.1.30: Auxiliary Clock 2 Select Control Register (CGM_AC2_SC)*
- *Section 26.2.1.32: Auxiliary Clock 3 Select Control Register (CGM_AC3_SC)*

Figure 217. MC_CGM auxiliary clock 0 generation overview

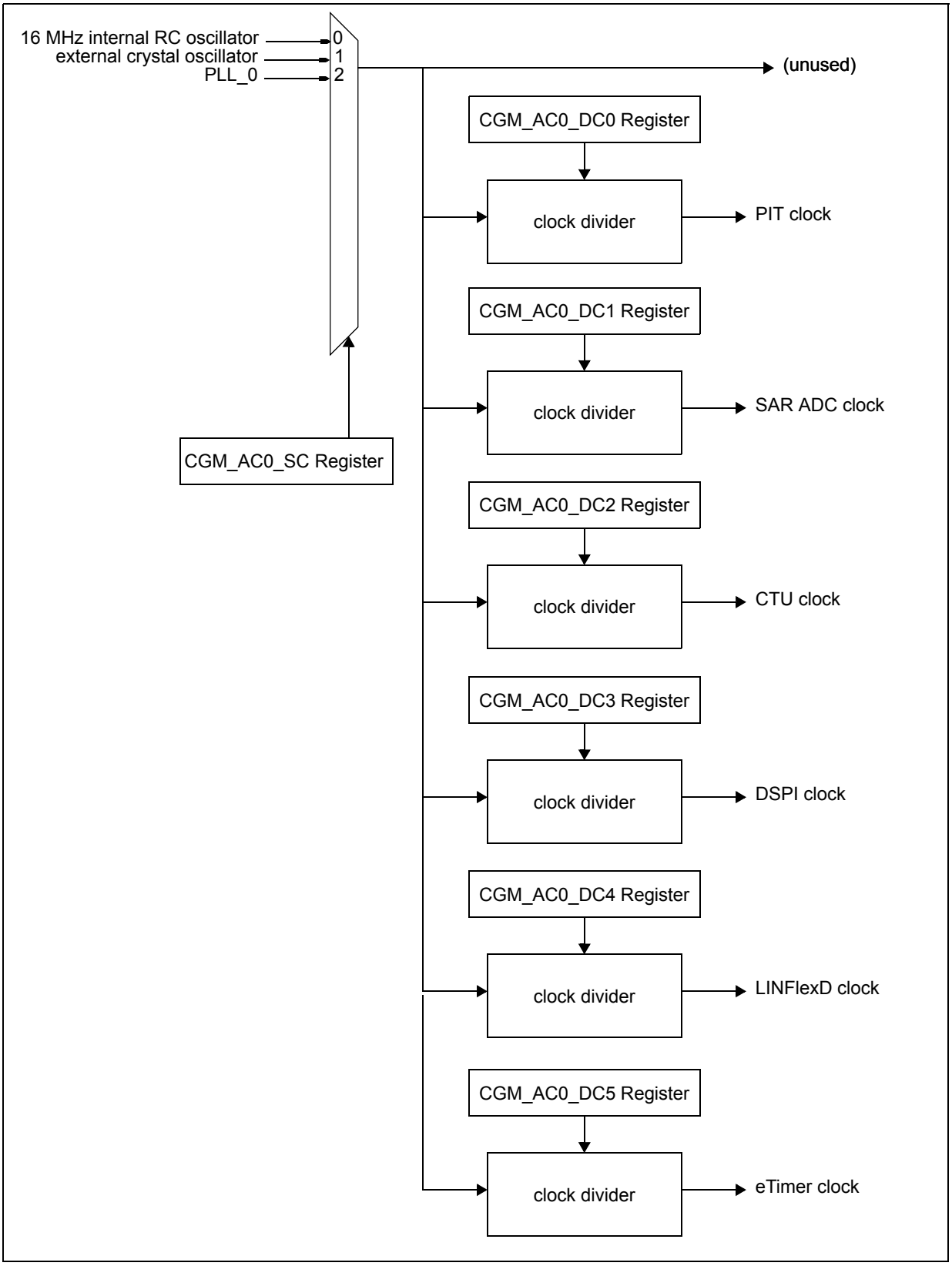


Figure 218. MC_CGM auxiliary clock 1 generation overview

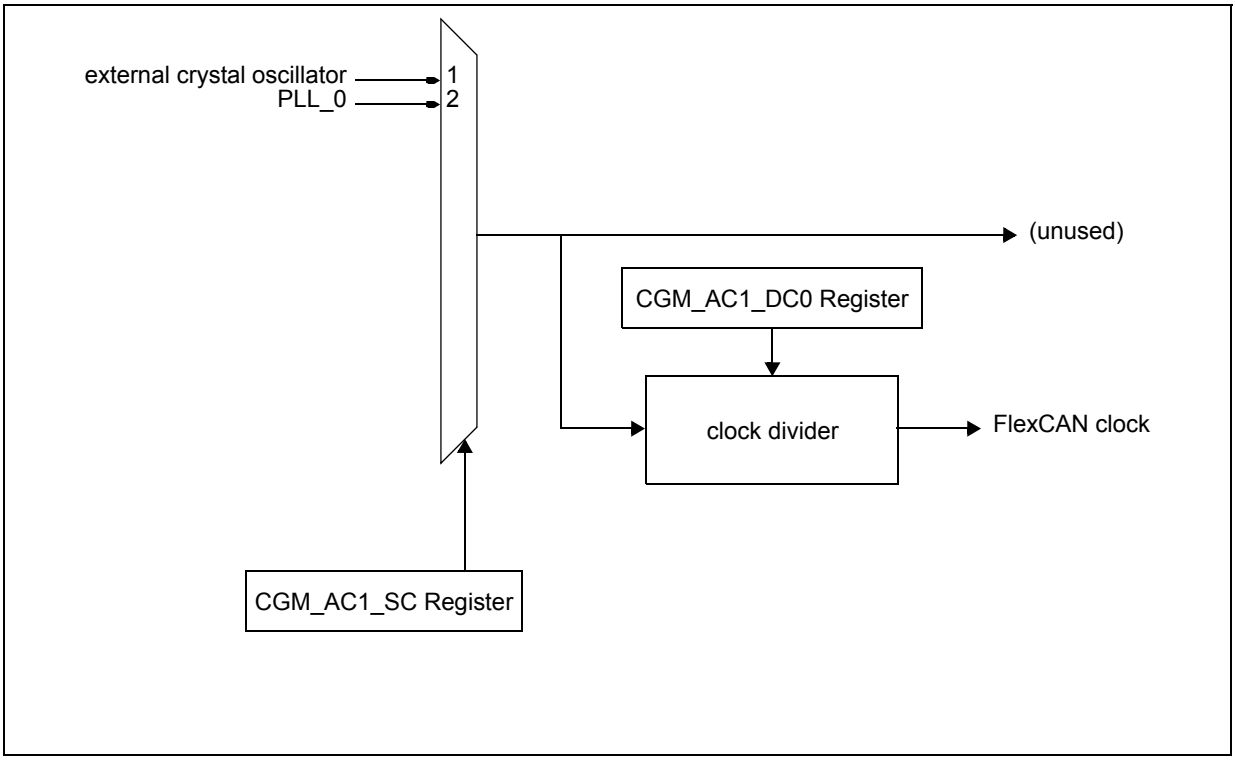


Figure 219. MC_CGM auxiliary clock 2 generation overview

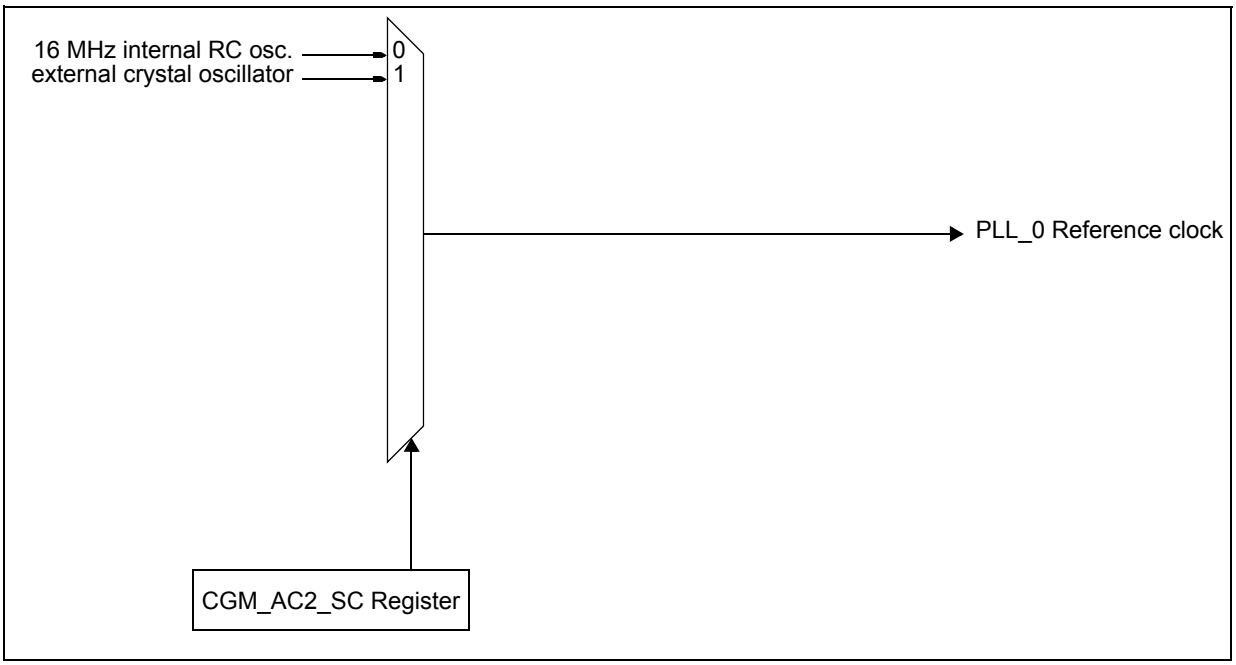
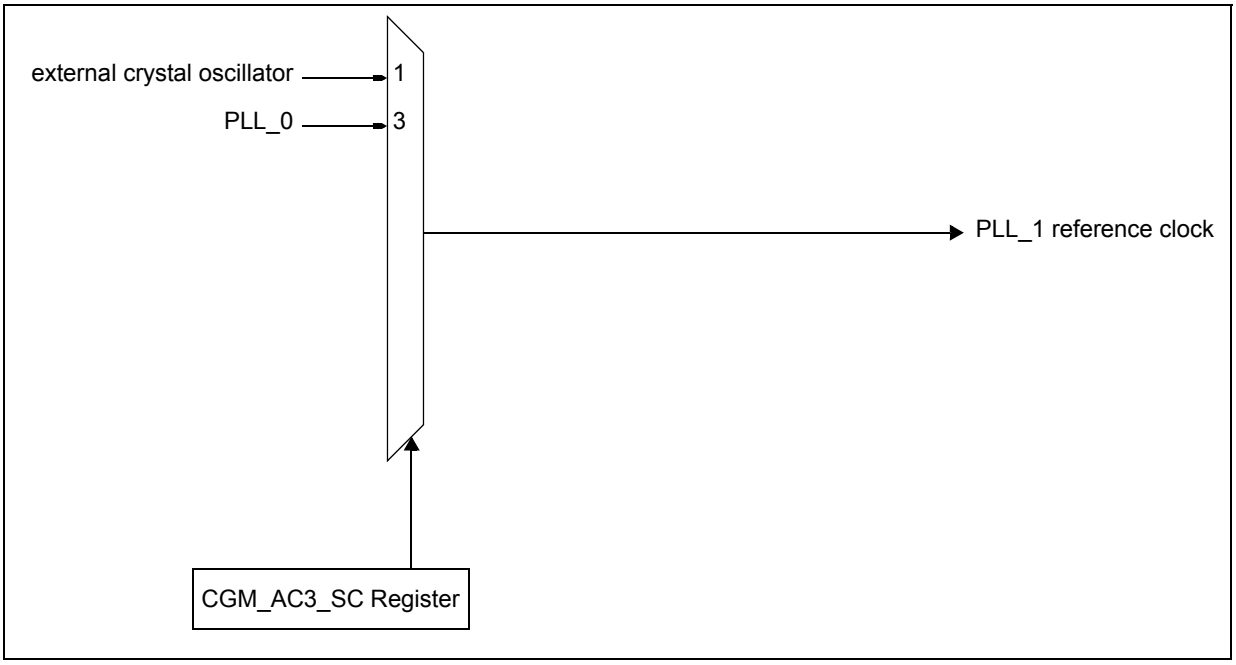


Figure 220. MC_CGM auxiliary clock 3 generation overview



26.3.2.1 Auxiliary clock dividers

The MC_CGM generates the following derived clocks:

- PIT clock - controlled by the CGM_AC0_DC0 register
- SAR ADC clock - controlled by the CGM_AC0_DC1 register
- CTU clock - controlled by the CGM_AC0_DC2 register
- DSPI clock - controlled by the CGM_AC0_DC3 register
- LINFlexD clock- controlled by the CGM_AC0_DC4 register
- eTimer clock- controlled by the CGM_AC0_DC5 register
- FlexCAN clock - controlled by the CGM_AC1_DC0 register

26.3.2.2 Divider update triggering

There are two methods for updating the dividers associated with the system clock and each auxiliary clock: immediate and register triggered. Which method is used for which set of dividers is controlled by the CGM_DIV_UPD_TYPE register.

Regardless of which method is selected for a given group of dividers, once an update of one of the dividers of that group has been triggered, the associated bit in the CGM_UPD_STAT register is set. This bit is then cleared when all the dividers of that group have completed the update process and are running with the current configuration.

26.3.2.2.1 Immediate divider update

If a divider group has been configured to be updated immediately (i.e., the corresponding CGM_DIV_UPD_TYPE register bit = ‘0’), the configuration of a divider in that group will be applied immediately on the write to that divider’s configuration register. The actual update will take some time to actually reach the divider output due to the clock domain crossing and edge alignment mechanisms.

26.3.2.2.2 Software-triggered divider update

If a divider group has been configured to be updated via a software trigger (i.e., the corresponding CGM_DIV_UPD_TYPE register bit = ‘1’), the configuration of a divider in that group will not be applied immediately on the write to that divider’s configuration register. Instead, the configuration in the corresponding configuration register will be applied only when software writes any value to the CGM_DIV_UPD_TRIG register. The actual update will take some time to actually reach the divider output due to the clock domain crossing and edge alignment mechanisms.

In the case of the system clock dividers, if the SYS_DIV_RATIO_CHNG bit in the CGM_SC_DIV_RC register = ‘1’, the crossbar switch is requested to halt its ongoing transactions, and only after the crossbar switch has acknowledged the request does the update of the system clock dividers take place. Once these updates have completed, the halt request to the crossbar is deasserted.

Note: If halting the crossbar can cause a bus master to starve and thus lead to lost data or other problems, it is recommended to vary the system clock divider configurations without changing the ratios and to set CGM_SC_DIV_RC[SYS_DIV_RATIO_CHNG] to ‘0’.

In order to ensure a clean clock divider configuration update, software must check that all ongoing updates have already completed (i.e., all relevant bits in the CGM_DIV_UPD_STAT register are ‘0’) before triggering a new update via the CGM_DIV_UPD_TRIG register.

27 OSC Digital Interface (XOSC)

27.1 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

27.2 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 283 shows the possible configurations of the XOSC.

Table 283. 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

27.2.1 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].

The XOSC can also be enabled by standalone control in the XOSC_CTL register. The XOSC_CTL[OSCON] bit controls power-down while the S_OSC bit provides the oscillator clock available status.

27.2.2 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 98: 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.

27.2.3 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).

27.2.4 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.

27.3 Memory map and register definition

Table 284 shows the memory map of the XOSC registers.

Table 284. XOSC memory map

Offset	Register	Location
0x00	XOSC Control Register (XOSC_CTL)	*Section 27.3.1.1*

27.3.1 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.

27.3.1.1 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	S_OSC	OSCON
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 221. XOSC_CTL register

Table 285. 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

Table 285. XOSC_CTL register field descriptions(Continued)

Field	Description
30 S_OSC	Crystal oscillator status 0 Crystal oscillator output clock is not stable 1 Crystal oscillator is providing a stable clock
31 OSCON	Crystal oscillator power-down control 0 Crystal oscillator is switched off 1 Crystal oscillator is switched on

28 IRCOSC digital interface

28.1 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)

It provides access to trimming information used for safety implementations.

28.2 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

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.

28.3 Memory map and register definition

Table 286 shows the memory map of the IRCOSC registers.

Table 286. 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 28.3.1.1*
0x04	IRCOSC Native Trimming Register (IRCOSC_NT)	32	R	0xXX00_0XXX(1)	*Section 28.3.1.2*

1. See register section for specific reset values.

28.3.1 Register descriptions

28.3.1.1 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 222. IRCOSC control register (IRCOSC_CTL)

Table 287. 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 288* shows how the IRCOSC_CTL[USER_TRIM] field is used to trim the IRCOSC frequency.

Table 288. 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 Data Sheet for df_{var_SW} value and min/max frequency variations.

28.3.1.2 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	Reserved(2)						0	0	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	RCTRIM							
W																
Reset	0	0	0	0	0	0	0	0	—(1)	—(1)	—(1)	—(1)	—(1)	—(1)	—(1)	—(1)

1. Reset value is coming from factory calibration (TEST Flash)
2. Always read ‘0’

Figure 223. IRCOSC Native Trimming register (IRCOSC_NT)

Table 289. IRCOSC_NT field descriptions

Field	Description
24:31 RCTRIM	IRCOSC trimming value for safety check

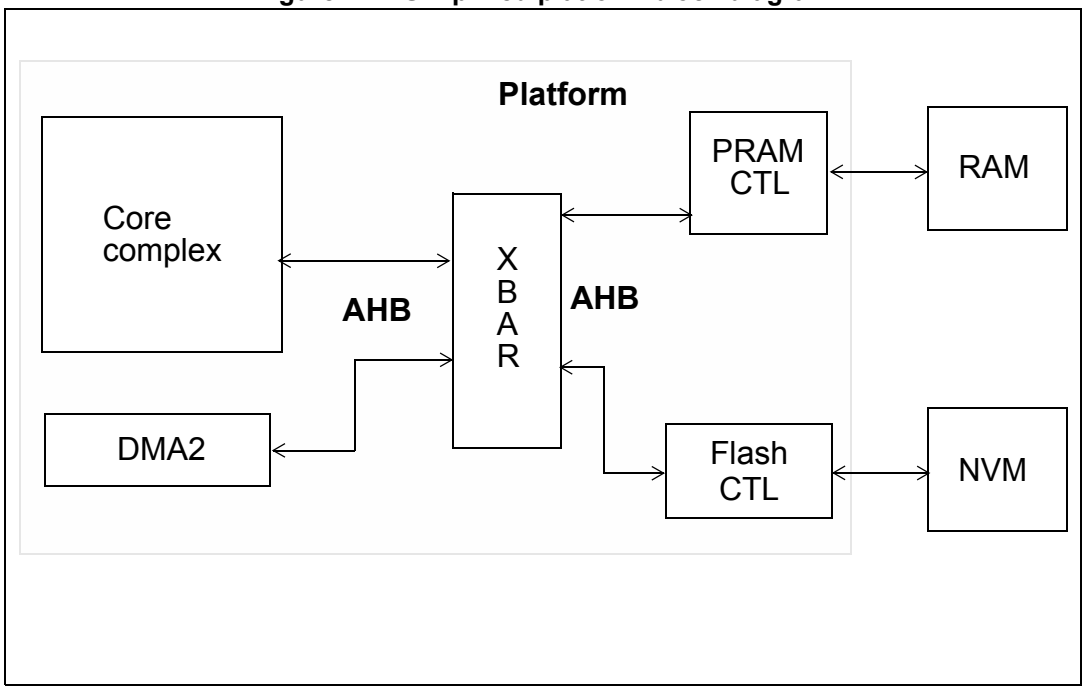
29 Platform RAM controller (PRAM)

29.1 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 32-bit AHB and one 32-bit RAM array interfaces. The first AHB port provides a connection to the platform crossbar for direct RAM access from the various crossbar masters. *Figure 224* is a simplified block diagram depicting the position of the PRAM controller within a typical platform architecture:

Figure 224. Simplified platform block diagram



The following list summarizes the key features of the PRAM controller:

- System bus supports 32-bit data and 8-bit ECC AHB interfaces
- Array interface supports a 32-bit data and 8-bit ECC interface
- Late-write support via 32-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 a single word

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 path contains a multiplexer that chooses the source of the read data to be presented on the system bus on a read request. In the event that a read request matches the contents of the late-write buffer, a RAM access is not required and the late-write buffer content is moved onto the read data bus. Otherwise, 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 32 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.

29.2 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 290. PRAM controller memory map

Address offset (hex)	Register name	Reset value (hex)	Location
0x00	Platform RAM Configuration Register 1 (PRCR1)	0x0000_0200	*Section 29.2.1*

29.2.1 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			P1_BO_DIS	P0_BO_DIS	0	0	0	0	0	
W							PRI1	PRI0								FT_DIS
Reset	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0

Figure 225. Platform RAM configuration register 1 (PRCR1)

Table 291. 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 32-bit WRP4 read bursts are optimized such that the controller returns a 2-1-1-1 response(1) when PRCR[FT_DIS]=1 1 32-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

1. The number of cycles taken for a RAM access can vary ±1 clock cycle depending on the RAM speed relative to the PRAM controller clock frequency. If system RAM is running at the same frequency as the PRAMC controller a random initial access takes 2 clock cycles. If system RAM is running at a slower frequency a random initial access may take 3 clock cycles. Subsequent burst beats take either 1 or 2 cycles depending on RAM speed relative to PRAMC controller clock frequency.

29.3 Functional description

29.3.1 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.

29.3.1.1 Reads

Read transfers, of any size, can be configured to complete with a zero or one additional wait state response on the AHB.

29.3.1.2 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

29.3.2 Writes

29.3.2.1 32-bit writes

Aligned **32**-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 32-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.

29.3.2.2 Less than 32-bit writes

Writes of size less than 32 bits incur a read-modify-write action as a consequence of the ECC coding scheme's 32-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 word. The newly formed word and its associated checkbits are subsequently written to the RAM. For details on the ECC coding scheme, refer to *Section 29.5: 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 32 bits require the insertion of one wait state before the data phase can be completed.

29.3.2.3 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.

29.4 Initialization/application information

Prior to reading, the ECC must be initialized by writing a known 32-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 32 bits or an unaligned write is requested. Reading from an uninitialized address is likely to generate an uncorrectable ECC event.

29.5 Safety considerations

29.5.1 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 32-bit (4-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 292*. A ‘*’ in *Table 292* 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 292. 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		*			*	*	*				*			*	*	*		*				*				*		*	*		*	

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.

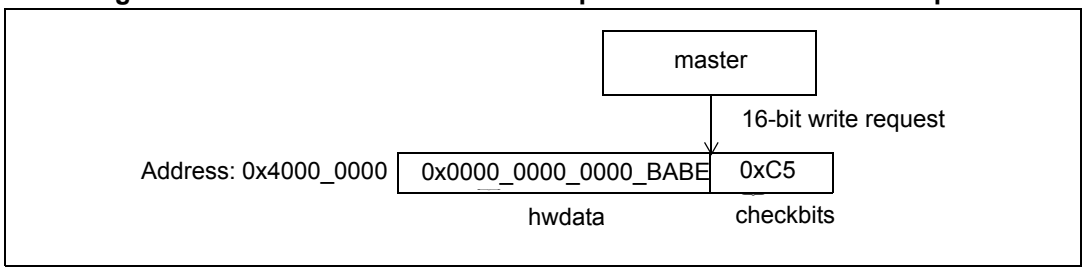
29.5.1.1 E2EECC considerations on less-than-32-bit write transactions

Recall that writes of less than 32-bits incur a read-modify-write action as a consequence of the ECC coding scheme 32-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, or 16-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:

Note: The below example is shown for a 64-bit RAM array access. The same mechanism should be extrapolated for 32-bit RAM array access.

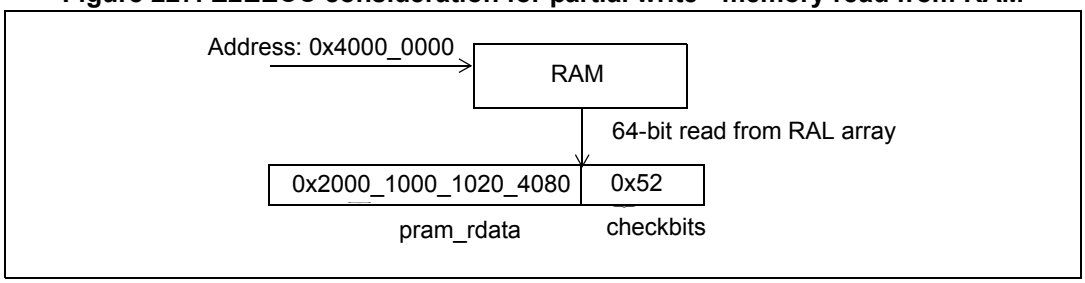
Figure 226. E2EECC consideration for partial write - 16-bit write request



- 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).

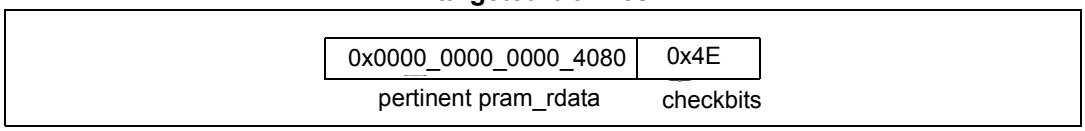
  If a non correctable ECC event is detected, the AHB request is terminated with error on the system bus. In addition, the ECC event is reported to the MEMU.

Figure 227. E2EECC consideration for partial write - memory read from RAM



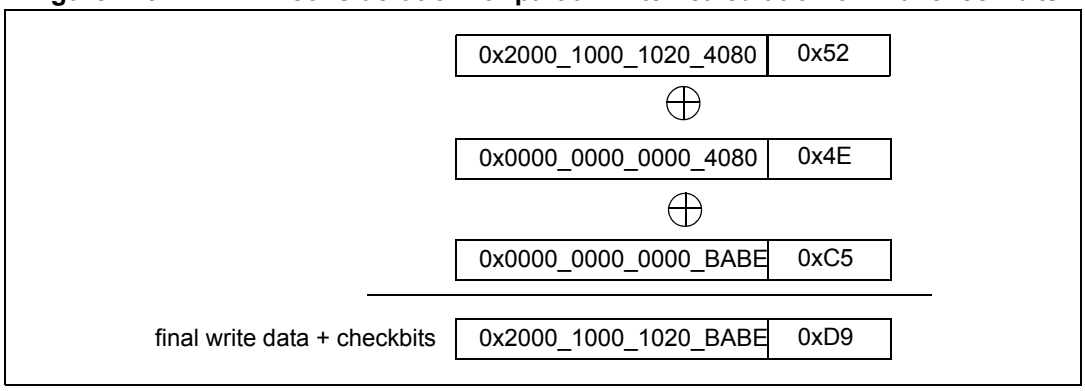
- The PRAM controller isolates the pertinent byte lanes of the read data by zero-padding the non pertinent byte lanes and calculating the checkbit contribution associated with the read data in the byte lanes to be updated as a result of the write request.

Figure 228. 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 229. 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.

29.5.2 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

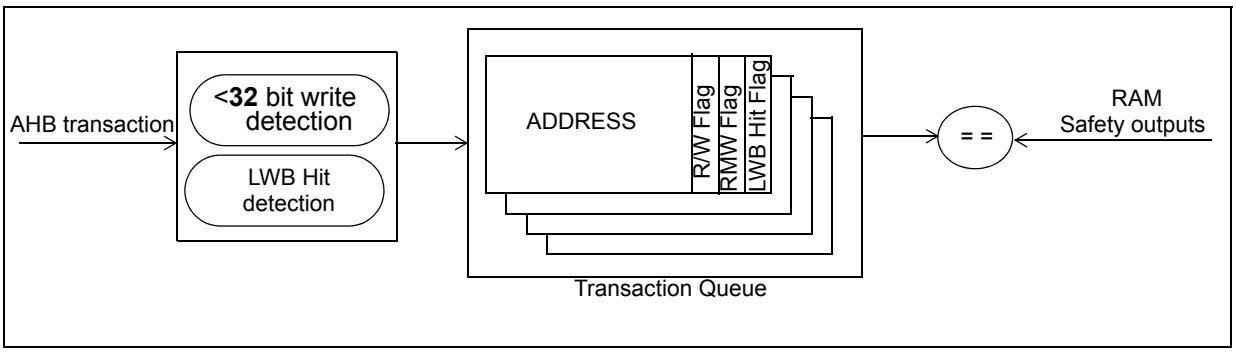
The function of the transaction monitor relies on a feedback path between the RAM controller and RAM array, wherein the RAM provides latched address and control feedback outputs as an indication of received inputs when a RAM access is initiated.

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.

As a countermeasure against false misses to the late-write buffer on reads, the transaction monitor must also verify the expected source of data returned on a read as supplied either from the RAM or from the late-write buffer.

Figure 230. RAM controller transaction monitor block diagram



30 Flash memory controller (PFLASH Controller)

30.1 Introduction

The flash memory controller has following function(s):

- It acts as an interface between the system bus (AHB-Lite 2.v6) and the flash array.

The flash memory controller supports two32-bit AHB buses and a 128-bit read data interface to the flash memory array. Each AHB port contains a 4-entry, 4-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 128 bits in size, matching the code flash array page size and providing 128 bytes of high-speed local storage.

30.2 Features

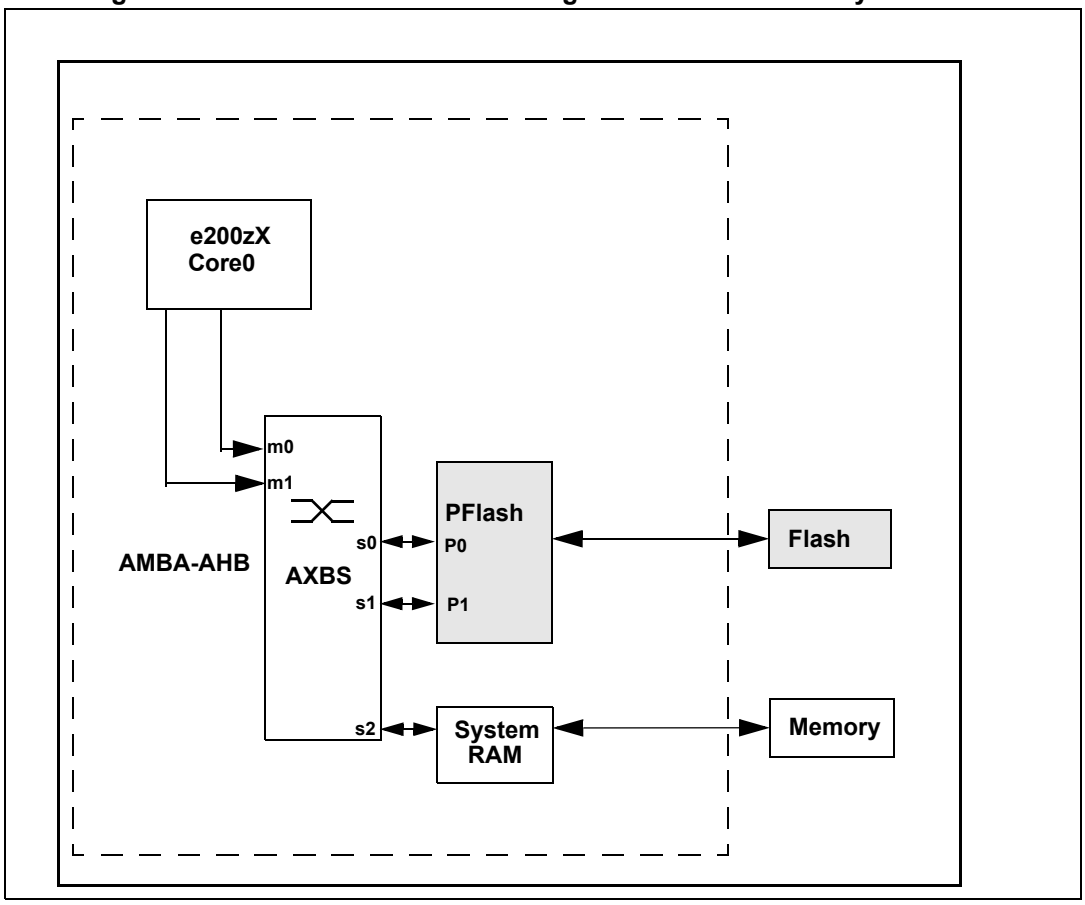
The following list summarizes the key features of the flash memory controller:

- Two32-bit AHB ports (p0, p1)
- 128-bit read data bus + 64-bit write data bus
- configurable read buffering and line prefetching support via 4-entry, 4-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

30.3 Block diagram

Figure 231 provides a block diagram showing the flash memory controller and the attached flash array.

Figure 231. Platform-centric block diagram with flash memory controller



30.4 Memory map and register definitions

30.4.1 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 293. Flash memory controller memory map

Address offset	Register	Location
0x000	Platform Flash Configuration Register 1 (PFCR1)	*Section 30.4.1.1*
0x004	Platform Flash Configuration Register 2 (PFCR2)	*Section 30.4.1.2*
0x008	Platform Flash Configuration Register 3 (PFCR3)	*Section 30.4.1.3*
0x00C	Platform Flash Access Protection Register (PFAPR)	*Section 30.4.1.4*
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.

30.4.1.1 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 232. Platform Flash Configuration Register 1 (PFCR1)

Table 294. 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 294. 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
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 294. 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.

30.4.1.2 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 233. Platform Flash Configuration Register 2 (PFCR2)

Table 295. 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 295. 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.

30.4.1.3 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 234. Platform Flash Configuration Register 3 (PFCR3)

Table 296. 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 296. 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

30.4.1.4 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 235. Platform Flash Access Protection Register (PFAPR)

Table 297. 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 297. 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 297. 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

30.5 Functional description

As shown in *Figure 231* 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 (128-bit page size) may be buffered in each of four 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.

30.5.1 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.

30.5.2 Access protections

30.5.2.1 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 30.4.1.4: 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.

30.5.2.2 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].

30.5.3 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.

30.5.4 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.

30.5.5 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 30.4.1.4: 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 30.4.1.3: 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 30.5.6: Flash error response operation*).

30.5.6 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.

30.5.7 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

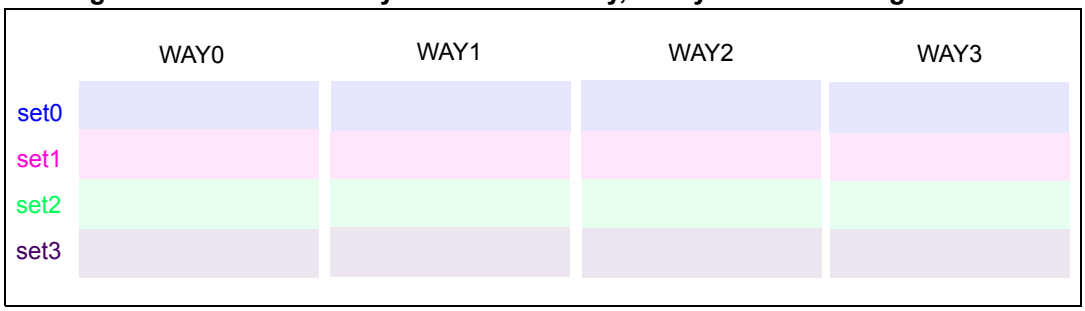
Each section can be independently censored, and the flash memory controller provides independent censorship control inputs for each region on a read vs. write basis.

30.5.8 Line read buffers and prefetch operation

Each AHB port of the flash memory controller contains a four-way set-associative mini cache, where each way contains four page buffers which are used to hold data read from the flash arrays. Each 128-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 30.4.1.1: 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 16-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 236. Flash memory controller 4-entry, 4-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 30.4.1.1: 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 30.4.1.1: Platform Flash Configuration Register 1 (PFCR1)* for a description of these controls.

30.5.9 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 30.4.1.1: 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 30.4.1.2: Platform Flash Configuration Register 2 (PFCR2)* for a description of these controls. Prefetches are never triggered by write cycles.

30.5.10 Per-Master prefetch triggering

Prefetch triggering may be controlled for individual bus masters. Refer to *Section 30.4.1.1: Platform Flash Configuration Register 1 (PFCR1)* for a description of these controls.

30.5.11 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 30.4.1.3: 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.

30.5.12 Safety considerations

30.5.12.1 e2eEDC (End-to-End EDC)

30.5.12.1.1 e2eEDC and flash accesses

There are multiple complications associated with the use of the standard e2eEDC 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 e2eEDC 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 e2eEDC 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 e2eEDC codeword:

addr = 0x00345678, rdata = 0xFFFFFFFF_FFFFFFFF, rchkbit = 0xC4 (0x00 ^ 0xC4)

2. Program address 0x00345678 with data = 0xBABEFACE_DEADBEEF. The initiating bus master calculates the e2eEDC 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 e2eEDC 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.

30.5.12.1.2 ECC 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 are returned to the requesting master, and the single-bit correction event is suppressed from being reported.

In the event of a non-correctable error detection, a fixed value, FFFF, is returned to the requesting master as determined by the requesting address and the non-correctable error event is suppressed from being reported.

30.5.12.2 Flash address generation check

Functional safety coverage of the address and data are handled by ECC performed within the flash. 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.

30.5.13 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.

31 Embedded Flash Memory

31.1 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**. which 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**. Containing the registers and logic which control the operation of the Flash core and contains 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.

31.1.1 Overview

The 560 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.

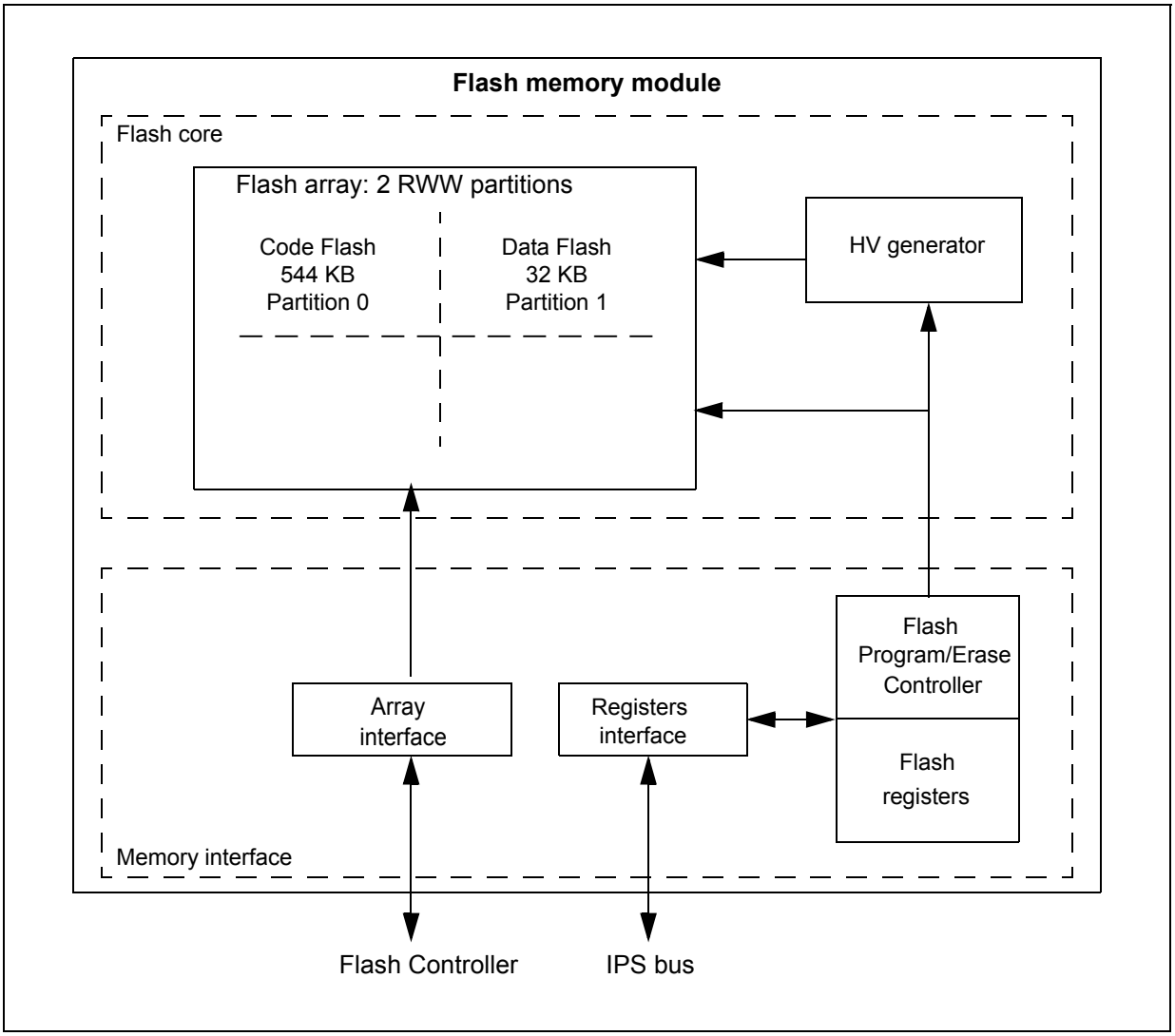
EEPROM emulation is managed in one block of the module, referred to as Data Flash.

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 128 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 237. 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 half-page (128 bits).
- For reading data partitions – by half-page (128 bits).

Multiple-word 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 128 bits, although Read buffering may be performed by the Bus Interface Unit (BIU). The address for each word retrieved within a half-page differs from the other addresses in the half-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.

31.1.2 Features

- 45 ns code/data array access time
- Code Partition High Read parallelism: 128 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.

31.1.3 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 31.4: Functional description*.

31.2 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 128K Address Space:

- Low Address Space must always be present and be up to 256 KB in size and may be organized in a maximum of 14 blocks (16 KB[n], 32 KB and 64 KB blocks).
- Mid Address Space 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 can be present and be up to 512 KB may be organized in a maximum of 16 small blocks (8 KB[n], 16 KB, 32 KB and 64 KB blocks).
- 128K Address Space can be present and be up to 3.75 MB in size and may be organized in a maximum of 31 blocks (128 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. The 128 KB blocks can be mapped in the 128K address space. 8 KB[n] 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.

n. 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.

31.2.1 Flash array memory map

31.2.1.1 Code Flash

Partition 0 accounts for 544 KB of the total Flash Memory and is divided into 10 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 128 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 560 KB Flash Memory module is shown in *Table 298*.

Table 298. Flash memory partition 0 memory map

Block	Absolute addresses	Size	Address space	LOCK	SEL
B0BF	0x0040_4000 to 0x0040_5FFF	8 KB	Low	LLK0	—
B0F1	0x00FC_0000 to 0x00FC_3FFF	16 KB	Low	LLK1	LSL1
B0F2	0x00FC_4000 to 0x00FC_7FFF	16 KB	Low	LLK2	LSL2
B0F3	0x00FC_8000 to 0x00FC_BFFF	16 KB	Low	LLK3	LSL3
B0F4	0x00FC_C000 to 0x00FC_FFFF	16 KB	Low	LLK4	LSL4
B0F5	0x00FD_0000 to 0x00FD_7FFF	32 KB	Low	LLK5	LSL5
B0F6	0x00FD_8000 to 0x00FD_FFFF	32 KB	Low	LLK6	LSL6
B0F7	0x00FE_0000 to 0x00FE_FFFF	64 KB	Low	LLK7	LSL7
B0F8	0x00FF_0000 to 0x00FF_FFFF	64 KB	Low	LLK8	LSL8
B0F9	0x0100_0000 to 0x0101_FFFF	128 KB	128 K	128KLK0	128KSL0
B0F10	0x0102_0000 to 0x0103_FFFF	128 KB	128 K	128KLK1	128KSL1
—	—	—	—	—	—
—	—	—	—	—	—
—	—	—	—	—	—
B0UT	0x0040_0000 to 0x0040_1FFF	8 KB	Low	TSLOCK	—
B0TF	0x0040_8000 to 0x0040_BFFF	16 KB	Reserved	—	—

31.2.1.2 Data Flash

Data Flash is allocated in partition 1 of the Flash Memory module in 4 × 8 blocks destined for data storage.

The Data Flash sectorization of the Flash Memory module is listed below.

Table 299. Flash Memory Partition 1 memory map

Block	Absolute addresses	Size	Address Space	LOCK	SEL
B1F0	0x0080_0000 to 0x0080_1FFF	8 KB	High	HLK0	HSL0
B1F1	0x0080_2000 to 0x0080_3FFF	8 KB	High	HLK1	HSL1
B1F2	0x0080_4000 to 0x0080_5FFF	8 KB	High	HLK2	HSL2

Table 299. Flash Memory Partition 1 memory map(Continued)

Block	Absolute addresses	Size	Address Space	LOCK	SEL
B1F3	0x0080_6000 to 0x0080_7FFF	8 KB	High	HLK3	HSL3
—	—	—	—	—	—

31.2.1.3 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 300*.

Table 300. TestFlash block memory map

Address Offset	Description	Size	Access
UTEST (Base Address 0x0040_0000 to 0x0040_1FFF)			
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	4096 bytes	R/P
BAF (Base Address 0x0040_4000 to 0x0040_5FFF)			
0x0000	User OTP – BAF Area	8 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 (8 KB) and BAF (8 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 stand-alone 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 stand-alone 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).

31.3 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.

31.3.1 Register memory maps

This section presents the memory mapping of all the Flash memory registers.

31.3.1.1 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 301*.

Table 301. Flash User Registers memory map

Offset address	Description	Access(1)	Reset value(2)	Location
0x0000	Module Configuration Register (MCR)	R/W	0x0000_0600	*Figure 238*
0x0004	Reserved	—	—	—
0x0008	Extended Module Configuration Register (MCRE)	R	0x4200_002A(3)	*Figure 239*
0x000C	Reserved	—	—	—
0x0010	Locking 0 register (LOCK0)	R/W	0xBFFF_FFFF	*Figure 240*
0x0014	Locking 1 register (LOCK1)	R/W	0x0000_FFFF	*Figure 241*
0x0018	Locking 2 register (LOCK2)	R/W	0x7FFF_FFFF	*Figure 242*
0x001C	Reserved	—	—	—
0x0020	Reserved	—	—	—
0x0024	Reserved	—	—	—
0x0028	Reserved	—	—	—
0x002C	Reserved	—	—	—
0x0030	Reserved	—	—	—
0x0034	Reserved	—	—	—
0x0038	Select 0 register (SEL0)	R/W	0x0000_0000	*Figure 243*
0x003C	Select 1 register (SEL1)	R/W	0x0000_0000	*Figure 244*
0x0040	Select 2 register (SEL2)	R/W	0x0000_0000	*Figure 245*
0x0044	Reserved	—	—	—
0x0048	Reserved	—	—	—
0x004C	Reserved	—	—	—
0x0050	Address Register (ADR)	R	0x0000_0000	*Figure 246*
0x0054	User Test register 0 (UT0)	R/W	0x0000_0001	*Figure 247*
0x0058	User Multiple Input Signature Register 0 (UM0)	R/W	0x0000_0000	*Figure 248*
0x005C	User Multiple Input Signature Register 1 (UM1)	R/W	0x0000_0000	*Figure 249*
0x0060	User Multiple Input Signature Register 2 (UM2)	R/W	0x0000_0000	*Figure 250*

Table 301. Flash User Registers memory map(Continued)

Offset address	Description	Access(1)	Reset value(2)	Location
0x0064	User Multiple Input Signature Register 3 (UM3)	R/W	0x0000_0000	*Figure 251*
0x0068	Reserved	R/W	—	—
0x006C	Reserved	R/W	—	—
0x0070	Reserved	R/W	—	—
0x0074	Reserved	R/W	—	—
0x0078	User Multiple Input Signature Register 8 (UM8)	R/W	0x0000_0000	*Figure 252*
0x007C	User Multiple Input Signature Register 9 (UM9)	R/W	0x0000_0000	*Figure 253*
0x0080	Over Program Protection 0 (OPP0)	R	0x0000_0000	*Figure 254*
0x0084	Over Program Protection 1 (OPP1)	R	0x0000_0000	*Figure 255*
0x0088	Over Program Protection 2 (OPP2)	R	0x0000_0000	*Figure 256*
0x008C	Reserved	—	—	—
0x0090	Test Mode Disable Password Check (TMD)	W	—	*Figure 257*

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.

31.3.2 User register descriptions

This section consists of user register descriptions in address order.

31.3.2.1 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	0	0	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 238. Module Configuration Register (MCR)

Table 302. 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.
6:7 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.
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.

Table 302. MCR field descriptions(Continued)

Field	Description
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 (i.e., 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.
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.

Table 302. MCR field descriptions(Continued)

Field	Description
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.
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.

Table 302. MCR field descriptions(Continued)

Field	Description
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.
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.

Table 302. MCR field descriptions(Continued)

Field	Description
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 302. 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 303*.

Table 303. 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.

31.3.2.2 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		N128K					N64KH			N32KH		N16KH		
W																
Reset	0	1	0	0	0	0	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	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 239. Extended Module Configuration Register (MCRE)

Table 304. 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 N128K	Number of 128 KB blocks. The value of the N128K field is dependent upon the size of the embedded Flash memory. N128K is read only. N128K value represents in the number of blocks effectively available. 00000 Zero 128 KB blocks. 00001 One 128 KB block. 00010 Two 128 KB blocks. 00011 Three 128 KB blocks. 00100 Four 128 KB blocks. 00101 Five 128 KB blocks. 00110 Six 128 KB blocks. 00111 Seven 128 KB blocks 01000 Eight 128 KB blocks. 01001 Nine 128 KB blocks. 01010 Ten 128 KB blocks. … 11100 28 128 KB blocks. 11101 29 128 KB blocks. 11110 30 128 KB blocks. 11111 31 128 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 304. 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 304. 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.

31.3.2.3 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’.

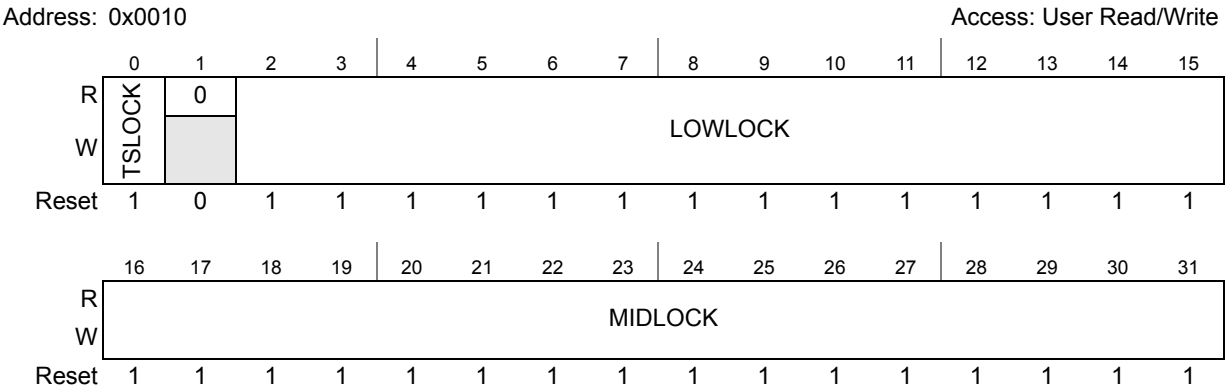


Figure 240. Lock 0 register (LOCK0)

Table 305. 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.

31.3.2.4 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 241. Lock 1 register (LOCK1)

Table 306. 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.

31.3.2.5 Lock 2 Register (LOCK2)

The LOCK2 (128K Address Space Block Locking) register provides a means of protecting blocks from being modified. The A128KLOCK bits corresponding to nonexistent blocks are read as ‘1’.

To determine the effective status of the A128KLOCK 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’.

Address: 0x0018 Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	A128KLOCK														
W																
Reset	0	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	A128KLOCK															
W																
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Figure 242. Lock 2 Register (LOCK2)

Table 307. LOCK2 field descriptions

Field	Description
1:31 A128KLOCK	128K 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 128 KB blocks starts with A128KLOCK[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 A128KLOCK bits is locked. When blocks are not present (due to configuration or total memory size), the A128KLOCK bits default to locked and are not writable. The reset value is always ‘1’ and register writes have no effect. 0 A128K Address Space Block is unlocked and can be modified. 1 A128K Address Space Block is locked and cannot be modified.

31.3.2.6 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 31.4.5.5: 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

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	LOWSEL													
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	MIDSEL															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 243. Select 0 register (SEL0)

Table 308. 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.

31.3.2.7 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 31.4.5.5: 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 244. Select 1 register (SEL1)

Table 309. 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.

31.3.2.8 Select 2 register (SEL2)

The SEL2 (128K 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 31.4.5.5: 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	A128KSEL														
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	A128KSEL															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 245. Select 2 register (SEL2)

Table 310. SEL2 field descriptions

Field	Description
1:31 A128KSEL	128K address space block select 30-0 The block numbering for the 128 KB blocks starts with A128KSEL[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 A128KSEL bits default to unselected and are not writable. The reset value is always ‘0’, and register writes have no effect. 0 A128K Address Space Block is unselected for Erase. 1 A128K Address Space Block is selected for Erase.

31.3.2.9 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 311* according to the priority rules in *Table 312*.

Table 311. ADR update mode list

Mode	Update on error	Note
Read	ECC Double Error, ECC Single Error (if UT0.SBCE=1)	—
Modify	FPEC error	—

Table 311. ADR update mode list(Continued)

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 312. 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 246. Address register (ADR)

Table 313. 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 314*, *Table 315*).

Table 314. Flash memory partition 0 addresses mapping

Block	Flash format addresses	Absolute addresses
B0BF	0x0080_6000 to 0x0080_7FFF	0x0040_4000 to 0x0040_5FFF
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 0x0005_FFFF	0x0100_0000 to 0x0101_FFFF
B0F10	0x0006_0000 to 0x0007_FFFF	0x0102_0000 to 0x0103_FFFF
—	—	—
—	—	—
—	—	—
B0UT	0x0080_4000 to 0x0080_5FFF	0x0040_0000 to 0x0040_1FFF
B0TF	0x0080_8000 to 0x0080_BFFF	0x0040_8000 to 0x0040_BFFF

Table 315. Flash Memory Partition 1 addresses mapping

Block	Flash format addresses	Absolute addresses
B1F0	0x0008_0000 to 0x0008_1FFF	0x0080_0000 to 0x0080_1FFF
B1F1	0x0008_2000 to 0x0008_3FFF	0x0080_2000 to 0x0080_3FFF
B1F2	0x0008_4000 to 0x0008_5FFF	0x0080_4000 to 0x0080_5FFF
B1F3	0x0008_6000 to 0x0008_7FFF	0x0080_6000 to 0x0080_7FFF
—	—	—

31.3.2.10 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 247. User Test 0 register (UT0)

Table 316. 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 316. 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 mode 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 writeable 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 316. 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 and 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 on-going. 1 Array Integrity Check is done.

31.3.2.11 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 (i.e., 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.

31.3.2.11.1 User Multiple Input Signature 0 register (UM0)

UM0 represents bits 31–0 of the whole 144-bit word (two 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 248. User Multiple Input Signature 0 register (UM0)

Table 317. 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.

31.3.2.11.2 User Multiple Input Signature 1 register (UM1)

UM1 represents bits 63–32 of the whole 144-bit word (two 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 249. User Multiple Input Signature 1 register (UM1)

Table 318. 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.

31.3.2.11.3 User Multiple Input Signature 2 register (UM2)

UM2 represents bits 95–64 of the whole 144-bit word (two 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 250. User Multiple Input Signature 2 register (UM2)

Table 319. 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.

31.3.2.11.4 User Multiple Input Signature 3 register (UM3)

UM3 represents bits 127–96 of the whole 144-bit word (two 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 251. User Multiple Input Signature 3 register (UM3)

Table 320. 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.

31.3.2.11.5 User Multiple Input Signature 8 register (UM8)

UM8 represents the ECC parity bits of the whole 144-bits word (8 parity bits for each of the two doublewords).

Address: 0x0078 Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	MISR287	MISR286	MISR285	MISR284	MISR283	MISR282	MISR281	MISR280	MISR279	MISR278	MISR277	MISR276	MISR275	MISR274	MISR273	MISR272
Reset	0	0	0	0	0	0	0	0	0	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	MISR271	MISR270	MISR269	MISR268	MISR267	MISR266	MISR265	MISR264	MISR263	MISR262	MISR261	MISR260	MISR259	MISR258	MISR257	MISR256
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 252. User Multiple Input Signature 8 register (UM8)

Table 321. 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.

31.3.2.11.6 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 253. User Multiple Input Signature 9 register (UM9)

Table 322. UM9 field descriptions

Field	Description
0:31 MISR[288]	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.

31.3.2.12 Over-program protection registers

Over-program protection prevents blocks from being over-programmed. These registers show the over-program protection status. See *Section 31.4.5.3: Over-programming protection (OPP) enable*.

31.3.2.12.1 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 254. Over-program protection 0 register (OPP0)

Table 323. 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.

31.3.2.12.2 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 255. Over-program protection 1 register (OPP1)

Table 324. 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.

31.3.2.12.3 Over-program protection 2 register (OPP2)

The 128K 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	A128KOPP														
W																
Reset(1)	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	A128KOPP															
W																
Reset(1)	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—

1. A reset value of "—" indicates that the value is set at the factory and can vary among different applications

Figure 256. Over-program protection 2 register (OPP2)

Table 325. OPP2 field descriptions

Field	Description
1:31 A128KOPP	128K High address space Over Program Protection 1-31 The block numbering for the 128 KB Blocks starts at A128KOPP[0] and continues until all blocks are accounted for. The A128KOPP field is not writable, and is status only. The default value of the A128KOPP bits is not over-programming protected. When blocks are not present (due to configuration or total memory size), the A128KOPP 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.

31.3.2.13 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 (Test Mode Disable Override Passcode at address offset 0x3DC8) 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 257. Test Mode Disable Password Check register (TMD)

Table 326. 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 31.4.6.2: 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.

31.4 Functional description

This chapter describes the details of all the modes of operation available to the Flash Memory module.

31.4.1 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.

31.4.2 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.

31.4.3 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.

31.4.4 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 128K) 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 128 (1 half-page = two 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.

31.4.4.1 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.

31.4.4.1.1 EEPROM emulation

The chosen ECC algorithm allows some bit manipulation so that a doubleword can be re-written 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 327 shows a set of doublewords sharing the same ECC value:

Table 327. 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
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.

31.4.4.2 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.

31.4.5 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 328*.

Table 328. 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.

31.4.5.1 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 un-selected 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 258. 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 */
```

31.4.5.2 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 259. 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 260. 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}.

31.4.5.3 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 128 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.

31.4.5.4 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.

31.4.5.5 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 128K 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 261. 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 */
```

31.4.5.5.1 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 262. 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 263. 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}.

31.4.5.5.2 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).

31.4.5.6 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).

31.4.5.6.1 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 128-bit data, the respective ECC parity data, and the double ECC (ERR) errors of the four doublewords are therefore captured by the MISR through two different Read accesses at the same location.

The whole check is done through two 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.

The 128-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 264. 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 A128KSL2-1 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 */
```

31.4.5.6.2 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 128-bit data, the corresponding ECC parity data and the double ECC (ERR) errors of the four doublewords are therefore captured by the MISR through two different Read accesses at the same location. The whole check is accomplished through two 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.

The 128-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 265. 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 */
```

31.4.6 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.

31.4.6.1 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 128K Address Space Block.

There are also Address Space Block Locking registers provided to independently lock/unlock each block of Low, Mid, High and 128K 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 128K 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).

31.4.6.2 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 329. 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 128K space	Data[78:48]

32 Error Injection Module (EIM)

32.1 Introduction

32.1.1 Overview

The Error Injection Module (EIM) provides support for inducing single-bit and multi-bit inversions on CPU write data path. Injecting errors on memory accesses can be used to exercise the SEC/DED ECC/EDC function of the related system. The EIM is mapped into the IP-Skyblue space.

32.1.2 Features

The EIM includes these features:

- Supports injection by recalculating ECC parity bits leaving the original write data unchanges
- Protection against accidental enable and reconfiguration error injection function via two-stage enable mechanism

32.2 Memory map and register definition

The EIM 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 generates an IPS error termination. The EIM allows access to the programming model by all system bus masters.

Attempted updates to the programming model while the EIM module is in the middle of an operation results in a non-deterministic behavior.

32.2.1 Memory map

Table 330. EIM memory map

Address Offset	Register	Location
0x0000	EIM Control Register (EIM_CTRL)	*Section 32.2.2.1*
0x0004	EIM Data Register (EIM_DATA)	*Section 32.2.2.2*
0x0008	Reserved(1)	
0x000C–3FFF	Reserved	

1. Access does not terminate in error response.

32.2.2 Register descriptions

Following sections provide description of the registers of this module.

32.2.2.1 EIM Control Register (EIM_CTRL)

The EIM Control Register is used to globally enable/disable the error injection function.

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	0	EIMEN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 266. EIM Control Register (EIM_CTRL)

Table 331. EIM_CTRL field descriptions

Field	Description
31 EIMEN	EIM Error Injection Enable. This bit enables or disables the error injection function of the EIM. 0 Error injection is not performed on any CPU write operation. 1 Error injection is performed on CPU write operation as configured in EIM Data Register

32.2.2.2 EIM Data Register (EIM_DATA)

The EIM Data Register (EIM_DATA) defines the 32-bit mask (DATA_MASK).

Each bit of DATA_MASK is used to enable or disable the error injection on individual data bits and provide exclusive control for single/double/multi bit error injection during the next CPU write operation. Note that there is a module error injection enable (EIM_CTRL[EIMEN]) also which must be asserted in conjunction with the EIM_DATA bits to enable an error injection for the next CPU write operation.

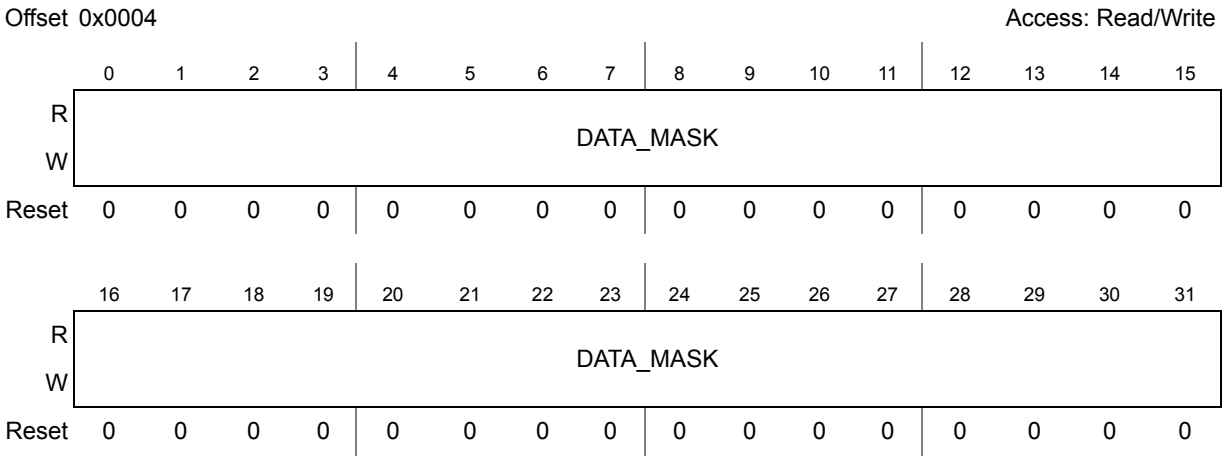


Figure 267. EIM Data Register (EIM_DATA)

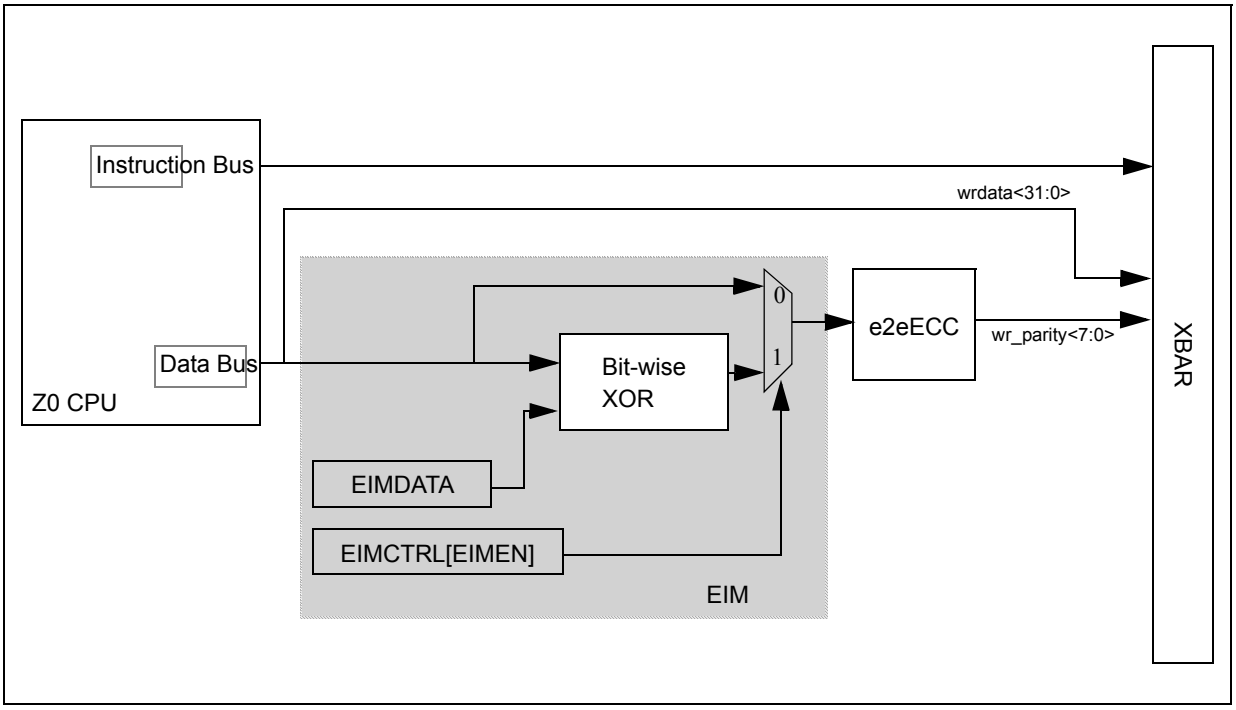
Table 332. EIM_DATA field descriptions

Field	Description
0–31 DATA_MASK	Data Mask. This field defines a bit-mapped mask that specifies if the corresponding bit of the 32-bit word on the write data bus from the CPU should be inverted or left unmodified. 0 The corresponding bit of the 32-bit word on the CPU write data bus is left unmodified. 1 The corresponding bit of the 32-bit word on the CPU write data bus is inverted.

32.3 Functional description

The error injection is performed by injecting errors in the parity bits during write operation of CPU (Figure 268). The incoming write-data bus from CPU is inverted if EIM is enabled and corresponding DATA_MASK bit is programmed.

Figure 268. EIM functional description

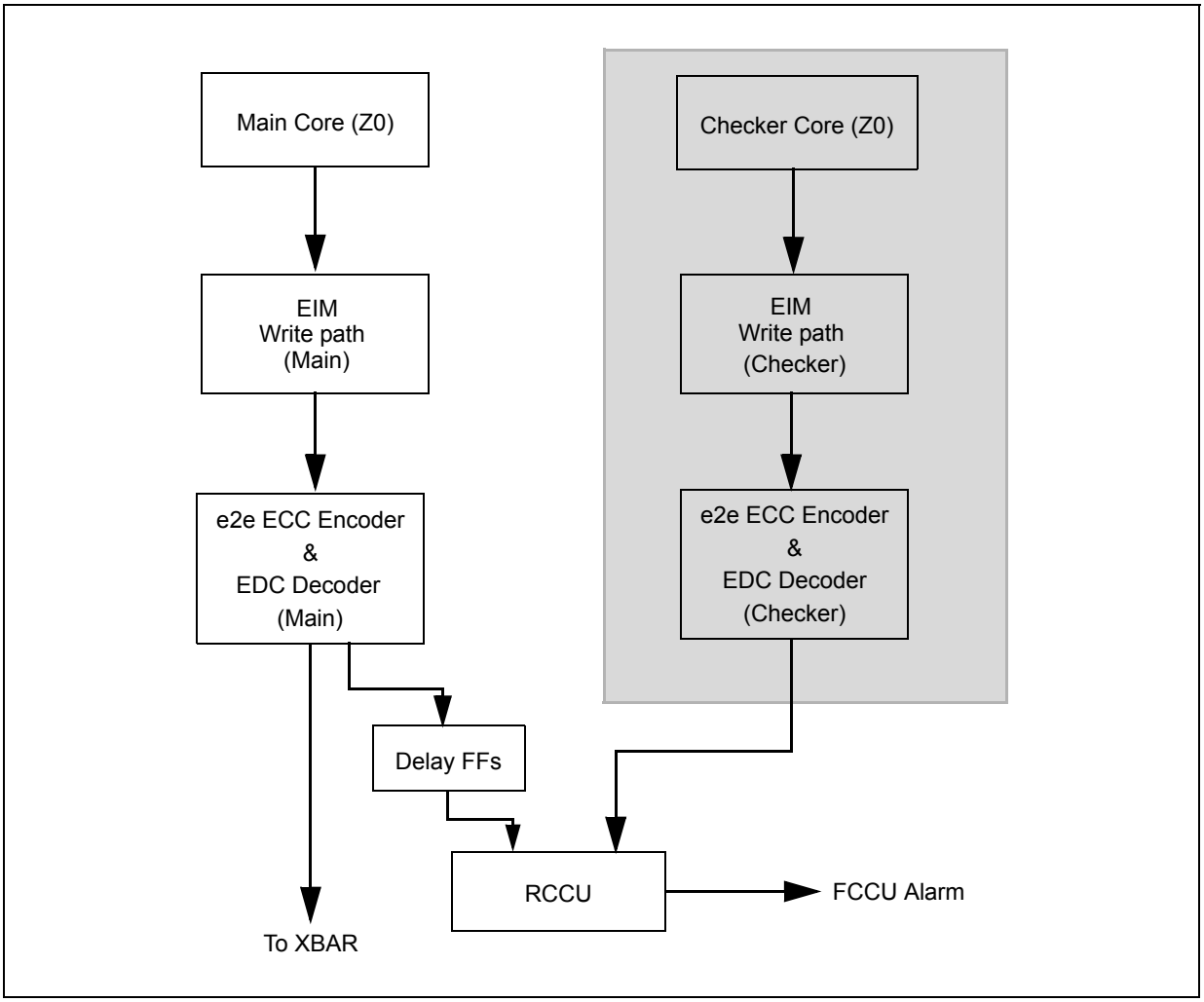


EIM is used to invert the write data bits which then is used to calculate parity bits. Thus the updated parity bits will be generated as per the inverted bits (single/multi).The XBAR receives the original write data bus along-with the EIM controlled parity bits which then is used to write into any slave location. When the written slave location is read back, it results in desired ECC error (SEC/DED).

Note: The EIM_CTRL[EIMEN] bit does not get reset automatically, CPU then needs to perform a dedicated write on EIMCTRL to reset it. As this write operation will happen when EIMEN is set, it will result in a Single/Multi-bit ecc error by the PBRIDGE EDC module which performs a check on all write operation for the correct parity. The user should take care of this error accordingly in FCCU/MEMU.

The EIM module is replicated in checker shell as well for the correct lockstep operation. The implementation is done as below:

Figure 269. EIM replication with lockstep operation



The replicated EIM provides a safety coverage of the functional EIM module.

32.4 Application Information

The following example provides the step sequence to inject a single bit error in a RAM location:

1. Write 32-bit (W32) EIM_DATA = 0x0000_0001 (single bit flip)
2. Write 32-bit (W32) EIM_CTRL = 0x0000_0001 (EIMEN = 1)
3. Write 32-bit at location 0x4000_0000 = 0xFFFF_FFFF (writing to a RAM location with correct data but incorrect parity)
4. Read 32-bit at location 0x4000_0000. Single-bit ecc error gets generated by CPU.
5. Write 32-bit (W32) EIM_CTRL = 0x0000_0000 (EIMEN = 0). It modifies again the parity with single bit error injection.
6. EIMEN gets reset but a single ecc error would be generated by PBRIDGE EDC due to EIM write performed with EIMEN set. This ecc error should be ignored by the user.

The following example provides the step sequence to inject a double bit error in a RAM location:

1. Write 32-bit (W32) EIM_DATA = 0x0000_0003 (double bits flip)
2. Write 32-bit (W32) EIM_CTRL = 0x0000_0001 (EIMEN = 1)
3. Write 32-bit at location 0x4000_0000 = 0xFFFF_FFFF (writing to a RAM location)
4. Read 32-bit at location 0x4000_0000. Multi-bit ecc error gets generated by CPU.
5. Write 32-bit (W32) EIM_CTRL = 0x0000_0000 (EIMEN = 0). It modifies again the parity with double bit error injection.
6. EIMEN gets reset but a multi ecc error would be generated by PBRIDGE EDC due to EIM write performed with EIMEN set. This ecc error should be ignored by the user.

33 Analog-to-digital converter (ADC) configuration

33.1 Overview

This chapter is organized into the following sections:

- *Section 33.1: Overview*
  - *Section 33.1.1: ADC subsystem block diagram*
  - *Section 33.1.2: Analog input pin multiplexing*
- *Section 33.2: Configuration*

SPC570Sx implements two 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 (16:1 input mux) and SARADC_0 (maximum 16:1 input mux).

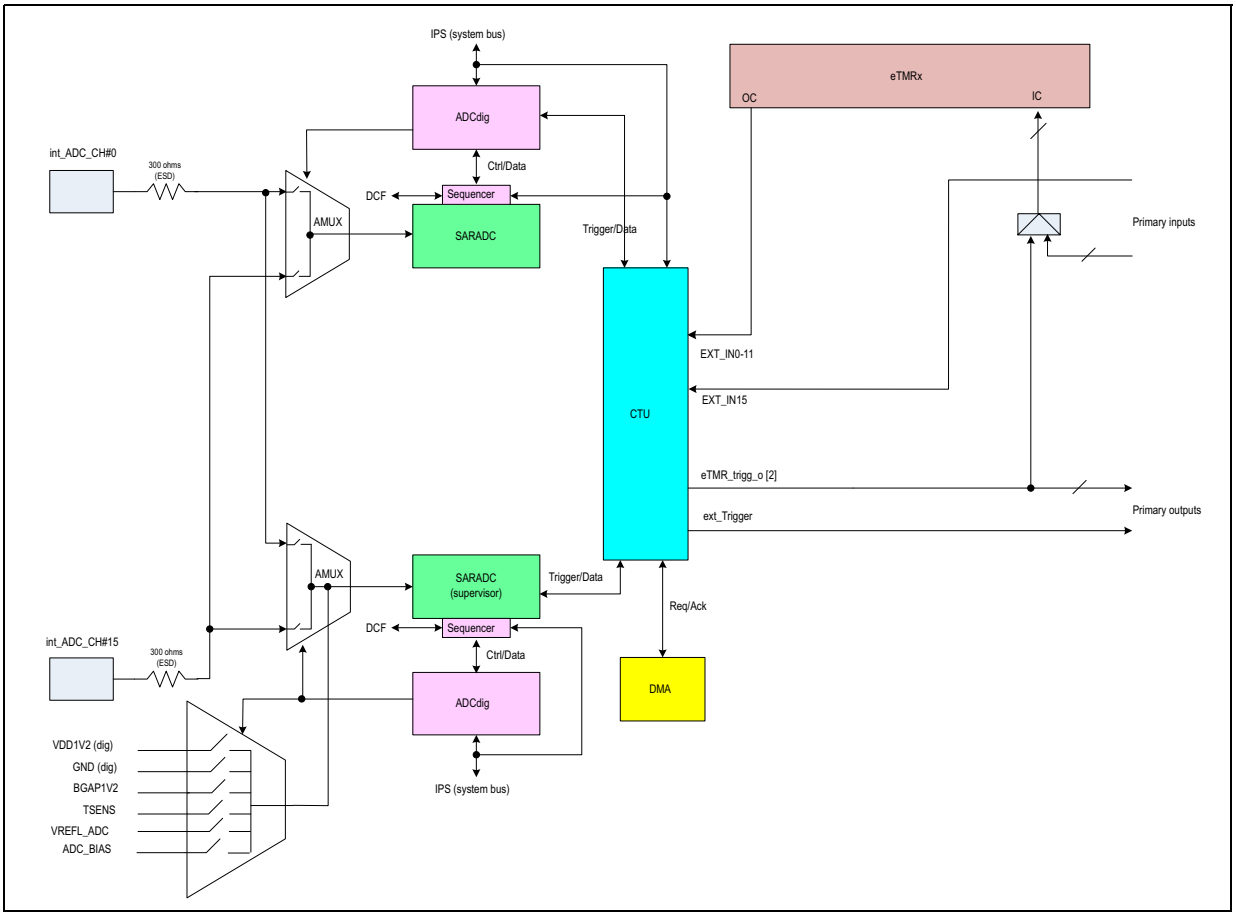
Each ADC instance has independent analog input pin multiplexing, register controls, trigger and clock inputs, independent interrupt and DMA capability. Each SARADC has a dedicated peripheral bridge (PBRIDGE) interface for CPU access to the registers.

The SARADCs share the same supply (VDD_HV_ADC_TSENS and VSS_HV_ADC_TSENS) and reference pins. This is true for both packages (eTQFP64 and eTQFP100).

33.1.1 ADC subsystem block diagram

Figure 270 gives the block diagram for the high-level integration of the ADC subsystem for SPC570Sx. The diagram illustrates how each ADC has independent register interfaces, interrupt/DMA requests, and trigger source selection.

Figure 270. ADC subsystem block diagram



33.1.2 Analog input pin multiplexing

The two SARADCs (functional and supervisor) are equivalent and fully share the 16 analog input channels. On the 64-pin package, only a subset of the total 16 analog input channels is available. *Table 333* gives the analog input pin assignment for both SPC570Sx 100-pin and 64-pin packages.

Table 333. SPC570Sx analog input pin multiplexing

ADC instance	eTQFP64		eTQFP100	
	Analog input pins	Mux input	Analog input pins	Mux input
SARADC_B	—	—	AN[15] (pad 18, pin 22)	AN[15] (pad 18, pin 22)
	AN[9] (pad 19, pin 14)	AN[9] (pad 19, pin 14)	AN[9] (pad 19, pin 23)	AN[9] (pad 19, pin 23)
	AN[8] (pad 20, pin 15)	AN[8] (pad 20, pin 15)	AN[8] (pad 20, pin 24)	AN[8] (pad 20, pin 24)
	AN[7] (pad 21, pin 16)	AN[7] (pad 21, pin 16)	AN[7] (pad 21, pin 25)	AN[7] (pad 21, pin 25)
	AN[6] (pad 22, pin 17)	AN[6] (pad 22, pin 17)	AN[6] (pad 22, pin 26)	AN[6] (pad 22, pin 26)
	AN[5] (pad 23, pin 18)	AN[5] (pad 23, pin 18)	AN[5] (pad 23, pin 27)	AN[5] (pad 23, pin 27)
	—	—	AN[14] (pad 24, pin 28)	AN[14] (pad 24, pin 28)
	—	—	AN[13] (pad 25, pin 30)	AN[13] (pad 25, pin 30)
	AN[4] (pad 26, pin 20)	AN[4] (pad 26, pin 20)	AN[4] (pad 26, pin 31)	AN[4] (pad 26, pin 31)
	AN[3] (pad 27, pin 21)	AN[3] (pad 27, pin 21)	AN[3] (pad 27, pin 32)	AN[3] (pad 27, pin 32)
	—	—	AN[12] (pad 28, pin 33)	AN[12] (pad 28, pin 33)
	—	—	AN[11] (pad 29, pin 34)	AN[11] (pad 29, pin 34)
	AN[2] (pad 30, pin 22)	AN[2] (pad 30, pin 22)	AN[2] (pad 30, pin 35)	AN[2] (pad 30, pin 35)
	AN[1] (pad 31, pin 23)	AN[1] (pad 31, pin 23)	AN[1] (pad 31, pin 36)	AN[1] (pad 31, pin 36)
	—	—	AN[10] (pad 32, pin 37)	AN[10] (pad 32, pin 37)
	AN[0] (pad 33, pin 24)	AN[0] (pad 33, pin 24)	AN[0] (pad 33, pin 38)	AN[0] (pad 33, pin 38)
SARADC_0	—	—	AN[15] (pad 18, pin 22)	AN[15] (pad 18, pin 22)
	AN[9] (pad 19, pin 14)	AN[9] (pad 19, pin 14)	AN[9] (pad 19, pin 23)	AN[9] (pad 19, pin 23)
	AN[8] (pad 20, pin 15)	AN[8] (pad 20, pin 15)	AN[8] (pad 20, pin 24)	AN[8] (pad 20, pin 24)
	AN[7] (pad 21, pin 16)	AN[7] (pad 21, pin 16)	AN[7] (pad 21, pin 25)	AN[7] (pad 21, pin 25)
	AN[6] (pad 22, pin 17)	AN[6] (pad 22, pin 17)	AN[6] (pad 22, pin 26)	AN[6] (pad 22, pin 26)
	AN[5] (pad 23, pin 18)	AN[5] (pad 23, pin 18)	AN[5] (pad 23, pin 27)	AN[5] (pad 23, pin 27)
	—	—	AN[14] (pad 24, pin 28)	AN[14] (pad 24, pin 28)
	—	—	AN[13] (pad 25, pin 30)	AN[13] (pad 25, pin 30)
	AN[4] (pad 26, pin 20)	AN[4] (pad 26, pin 20)	AN[4] (pad 26, pin 31)	AN[4] (pad 26, pin 31)
	AN[3] (pad 27, pin 21)	AN[3] (pad 27, pin 21)	AN[3] (pad 27, pin 32)	AN[3] (pad 27, pin 32)
	—	—	AN[12] (pad 28, pin 33)	AN[12] (pad 28, pin 33)
	—	—	AN[11] (pad 29, pin 34)	AN[11] (pad 29, pin 34)
	AN[2] (pad 30, pin 22)	AN[2] (pad 30, pin 22)	AN[2] (pad 30, pin 35)	AN[2] (pad 30, pin 35)
	AN[1] (pad 31, pin 23)	AN[1] (pad 31, pin 23)	AN[1] (pad 31, pin 36)	AN[1] (pad 31, pin 36)
	—	—	AN[10] (pad 32, pin 37)	AN[10] (pad 32, pin 37)
	AN[0] (pad 33, pin 24)	AN[0] (pad 33, pin 24)	AN[0] (pad 33, pin 38)	AN[0] (pad 33, pin 38)

33.2 Configuration

The following sections describe configuration details:

- *Section 33.2.1: External signal description*
- *Section 33.2.2: Integration diagram*
- *Section 33.1.2: Analog input pin multiplexing*
- *Section 33.2.3: Clock sources*
- *Section 33.2.4: Triggering*
- *Section 33.2.5: SAR absolute voltage reference*
- *Section 33.2.6: Diagnostics*
- *Section 33.2.7: Channel assignment*
- *Section 33.2.8: Register definitions*

For detailed module information, refer to *Chapter 34: Successive Approximation Register Analog-to-Digital Converter (SARADC) Digital Interface*.

33.2.1 External signal description

The following table defines the external package pins connected to the SARADC modules on the device.

Table 334. SARADC external signal description

Signal name	Type	Voltage	Description
AN[0:15](1)	Analog input	0–5 V	Single-ended analog input with pull up/down
VDD_HV_ADC_TSENS	Analog supply	3.3 V/5 V	SARADC power supply
VSS_HV_ADC_TSENS	Analog ground	0 V	SARADC ground supply
VREFH_ADC	Analog reference	2–5 V	SARADC reference
VREFL_ADC	Analog reference	0 V	SARADC ground reference

1. Not all signals listed here are implemented on all packages—see also *Table 333*.

33.2.2 Integration diagram

SARADC_B implements a bias generator converting the following reference points through the internal 20 kΩ source impedance:

- $3/3 \times (V_{REFH_ADC} - V_{REFL_ADC})$
- $2/3 \times (V_{REFH_ADC} - V_{REFL_ADC})$
- $1/3 \times (V_{REFH_ADC} - V_{REFL_ADC})$
- V_{REFL_ADC}

See the SARADC block diagram in *Chapter 34: Successive Approximation Register Analog-to-Digital Converter (SARADC) Digital Interface*.

33.2.3 Clock sources

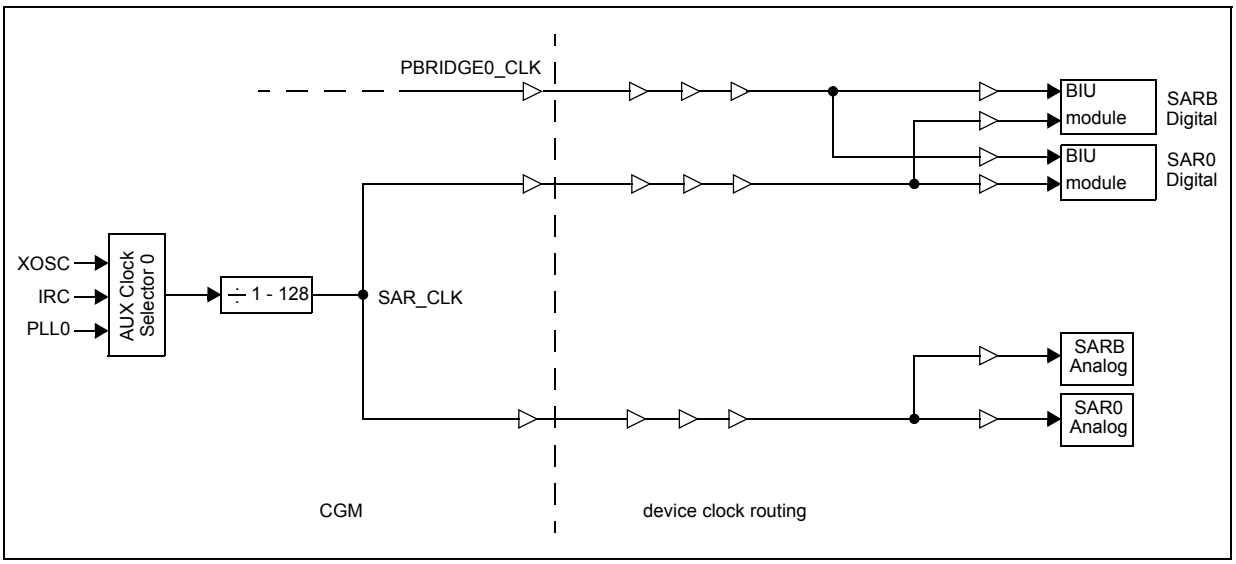
Each SARADC digital interface on SPC570Sx has a register clock input, which is separate from the module clock input. The register interface is clocked by the PBRIDGE0_CLK. The

module clock is sourced by the ADC clock (SAR_CLK). See Section 23.2, Clock generation for more detail on the clock sources.

The input clock for the SARADC analog blocks is clocked by the SAR_CLK. *Figure 271* shows a block diagram of the SARADC clock architecture. The maximum frequency for the clock input to the converter is defined in the device datasheet.

To avoid crosstalk issues, the two SARADCs run at the same clock frequency. The SAR digital interface is clocked from the PBRIDGE0_CLK clock. The SAR analog blocks are clocked from the output of a single divider of the SAR_CLK at the device level.

Figure 271. SARADC clock architecture



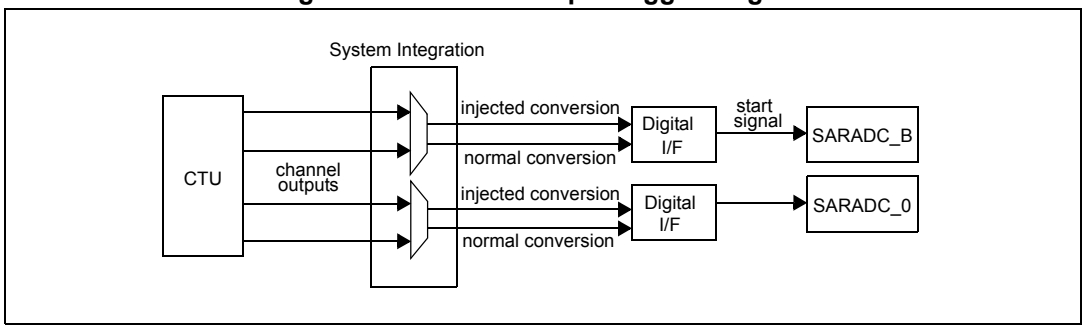
33.2.4 Triggering

Each SARADC has two independent trigger inputs for starting conversion on an analog input: one is related to injected conversion and the other is related to normal conversion. Each SARADC on the device has an independent set of trigger inputs, but the same sources can be used by each SARADC and the trigger inputs are configurable by SIUL2_MSCRs associated to the SARADC.

The trigger sources for each SARADC on the device are CTU channels.

The SPC570Sx SoC diagram for the SARADC trigger inputs is given in *Figure 272*.

Figure 272. SARADC input trigger diagram



Trigger source selection for each SARADC is given in *Chapter 13: System Integration Unit Lite2 (SIUL2)*.

33.2.5 SAR absolute voltage reference

SPC570Sx provides an internal voltage which decreases with temperature (CTAT voltage). This voltage after converting through ADC is used to generate a reference code which remains constant with temperature (See *Chapter 22: Temperature sensor (TSENS)*: *Section 22.6: Equations for converting CTAT and PTAT voltages into constant reference code* for details). This voltage can be read by SARADC_B. The ADC conversion channel number for this CTAT voltage is given in *Table 336*.

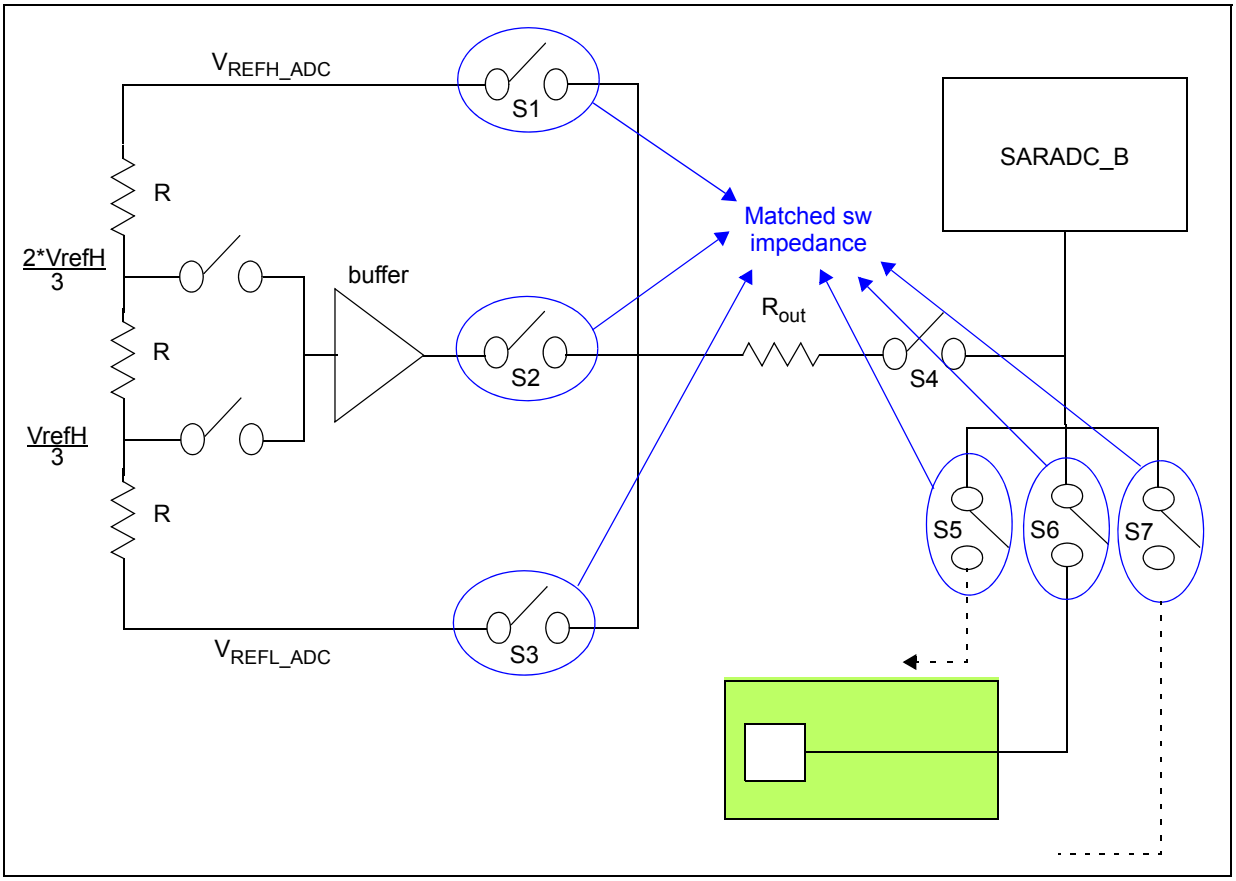
33.2.6 Diagnostics

33.2.6.1 Self Test features

Following are the Self Test features:

- The ADCBIAS provides four different voltage levels defined as VREFL_ADC, VREFH_ADC/3, 2 * VREFH_ADC/3 and VREFH_ADC. The self test internal reference voltage precision is provided in the SARn ADC electrical specification table of the device datasheet.
- The impedance requirement has been satisfied putting a physical poly resistor R_{out} = 20 kΩ typical characterized by a process and temperature variation of ±30%.
- The impedance of the switches that are connecting the different references to the ADC input has not been considered in the impedance calculation.
- The impedance of the switches from the reference generation to the ADC input has been matched with the one of the switches that are connecting the device pin to the ADC input.
- The sampling time of SARADC_B needs to be extended to 2.5 µs due to long settling time induced by the high impedance of the source (20 kΩ).
- The SARADC Digital Interface provides the necessary control to select the required switches S1–S7 (shown in *Figure 273: Self Test implementation*) to perform the Self Test.

Figure 273. Self Test implementation



33.2.6.2 Internal reference

SARADC_B provides the ability to sample and convert four internal voltages through a 20 kΩ source impedance. These voltages are as follows:

- V_{REFH_ADC}
- 1/3 × V_{REFH_ADC}
- 2/3 × V_{REFH_ADC}
- V_{REFL_ADC}

Conversion of an internal reference voltage works the same as that for a normal conversion, with the exception of the conversion time. Due to the 20 kΩ source impedance, the sampling time is much higher than the normal time.

Enabling the sampling and selection of an internal reference is independent of input channel selection for SARADC_B.

33.2.6.3 Analog input pin programmable pull up/down

An open or short circuit condition is detectable within the device for analog input pins. There is a programmable pull up/down pad cell at the input pad. The pull up/down is included inside the analog input pad cell.

Control of the pull up/down is independently done in the SIUL2_MSCRs for each analog input pin.

Only the analog input pins specified in *Table 334* have the pull up/down feature.

33.2.7 Channel assignment

The table below provides the input multiplexer channel assignments for the SARADCs.

Table 335. SARADC analog input channel assignment

Analog input pin	SARADC_B input channel(1)	SARADC		Description
		Module	Channel	
AN[15:0]	15–0	SARADC_B	15–0	Analog input pin
AN[15:0]	15–0	SARADC_0	15–0	Analog input pin

1. Not all SARADC_B input channels are implemented on all packages—see also *Table 333*.

33.2.7.1 Internal voltage monitor channel assignments

Each internal supply monitor and other test signals have a unique test channel assignment, and are treated by software like any other A/D conversion. Access to the internal test channels is provided on SARADC_B. The internal test channel assignments for SARADC_B are given in *Table 336*.

Table 336. SARADC_B analog test channel assignment

SARADC_B input channel	Description
96	Reserved
97	Reserved
98	Reserved
99	Reserved
100	Reserved
101	Reserved
102	Reserved
103	Reserved
104	VDD_LV
105	Ground
106	Reserved
107	Reserved
108	Reserved
109	VREFL_ADC (low voltage reference)
110	Reserved
111	Reserved
112	Reserved
113	Reserved

Table 336. SARADC_B analog test channel assignment(Continued)

SARADC_B input channel	Description
114	Reserved
115	Reserved
116	Reserved
117	Reserved
118	Reserved
119	Reserved
120	Temperature Sensor (PTAT)
121	ADC Bandgap Reference (CTAT)
122	Reserved
123	Reserved
124	SAR BIAS 0—V_{REFL_ADC} through 20 kΩ source impedance
125	SAR BIAS 1—1/3 × (V_{REFH_ADC} – V_{REFL_ADC}) through 20 kΩ source impedance
126	SAR BIAS 2—2/3 × (V_{REFH_ADC} – V_{REFL_ADC}) through 20 kΩ source impedance
127	SAR BIAS 3—3/3 × (V_{REFH_ADC} – V_{REFL_ADC}) through 20 kΩ source impedance

33.2.8 Register definitions

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

- *Section 33.2.8.1: SARADC_B register definitions*
- *Section 33.2.8.2: SARADC_0 register definitions*

Table 337 provides a key for the register definition tables. Test channels are only used for SARADC_B.

Table 337. 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

Table 337. SARADC register descriptions(Continued)

Name	Description	User access
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
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 338. SARADC test channel register descriptions (SARADC_B only)

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

33.2.8.1 SARADC_B register definitions

Table 339. SARADC_B 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	NSTART	NTRGEN	NEDGESEL		JSTART	JTRGEN	JEDGESEL		JTRGSEQ	0	0	0
	W																
	R	0	0	0	0	JTRGSEL				ABORT	ABORT	0	FRZ	0	0	0	PWDN
	W																

Table 339. SARADC_B 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	JABORT	0	0
	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	0	0	JEOC	JECH	NEOC	NECH
	W													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_CH[15]	EOC_CH[14]	EOC_CH[13]	EOC_CH[12]	EOC_CH[11]	EOC_CH[10]	EOC_CH[9]	EOC_CH[8]	EOC_CH[7]	EOC_CH[6]	EOC_CH[5]	EOC_CH[4]	EOC_CH[3]	EOC_CH[2]	EOC_CH[1]	EOC_CH[0]
	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	0	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/W	IM_CH[15]	IM_CH[14]	IM_CH[13]	IM_CH[12]	IM_CH[11]	IM_CH[10]	IM_CH[9]	IM_CH[8]	IM_CH[7]	IM_CH[6]	IM_CH[5]	IM_CH[4]	IM_CH[3]	IM_CH[2]	IM_CH[1]	IM_CH[0]
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 339. SARADC_B 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	DS_CH[15]	DS_CH[14]	DS_CH[13]	DS_CH[12]	DS_CH[11]	DS_CH[10]	DS_CH[9]	DS_CH[8]	DS_CH[7]	DS_CH[6]	DS_CH[5]	DS_CH[4]	DS_CH[3]	DS_CH[2]	DS_CH[1]	DS_CH[0]
	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_CH[15]	NCE_CH[14]	NCE_CH[13]	NCE_CH[12]	NCE_CH[11]	NCE_CH[10]	NCE_CH[9]	NCE_CH[8]	NCE_CH[7]	NCE_CH[6]	NCE_CH[5]	NCE_CH[4]	NCE_CH[3]	NCE_CH[2]	NCE_CH[1]	NCE_CH[0]
	W																
ICJCMR0	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	JCE_CH[15]	JCE_CH[14]	JCE_CH[13]	JCE_CH[12]	JCE_CH[11]	JCE_CH[10]	JCE_CH[9]	JCE_CH[8]	JCE_CH[7]	JCE_CH[6]	JCE_CH[5]	JCE_CH[4]	JCE_CH[3]	JCE_CH[2]	JCE_CH[1]	JCE_CH[0]
	W																

Table 339. SARADC_B 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	0	0	0	PCE	0	CTSEL		0	0	0	0	VALID	OVERW	RESULT	
	W																
	R	0	0	0	0	CDATA											
	W																
ICWSELR0	R	0	WSEL_CH7			0	WSEL_CH6			0	WSEL_CH5			0	WSEL_CH4		
	W																
	R	0	WSEL_CH3			0	WSEL_CH2			0	WSEL_CH1			0	WSEL_CH0		
	W																
ICWSELR1	R	0	WSEL_CH15			0	WSEL_CH14			0	WSEL_CH13			0	WSEL_CH12		
	W																
	R	0	WSEL_CH11			0	WSEL_CH10			0	WSEL_CH9			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_CH[15]	WEN_CH[14]	WEN_CH[13]	WEN_CH[12]	WEN_CH[11]	WEN_CH[10]	WEN_CH[9]	WEN_CH[8]	WEN_CH[7]	WEN_CH[6]	WEN_CH[5]	WEN_CH[4]	WEN_CH[3]	WEN_CH[2]	WEN_CH[1]	WEN_CH[0]
	W																
ICAWORR0	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	AWOR_CH[15]	AWOR_CH[14]	AWOR_CH[13]	AWOR_CH[12]	AWOR_CH[11]	AWOR_CH[10]	AWOR_CH[9]	AWOR_CH[8]	AWOR_CH[7]	AWOR_CH[6]	AWOR_CH[5]	AWOR_CH[4]	AWOR_CH[3]	AWOR_CH[2]	AWOR_CH[1]	AWOR_CH[0]
	W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c

Table 340. SARADC_B 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_CH[127]	EOC_CH[126]	EOC_CH[125]	EOC_CH[124]	0	0	EOC_CH[121]	EOC_CH[120]	0	0	0	0	0	0	0	0
	W	w1c	w1c	w1c	w1c			w1c	w1c								
	R	0	0	EOC_CH[109]	0	0	0	EOC_CH[105]	EOC_CH[104]	0	0	0	0	0	0	0	0
	W			w1c				w1c	w1c								
TCIMR	R	IM_CH[127]	IM_CH[126]	IM_CH[125]	IM_CH[124]	0	0	IM_CH[121]	IM_CH[120]	0	0	0	0	0	0	0	0
	W																
	R	0	0	IM_CH[109]	0	0	0	IM_CH[105]	IM_CH[104]	0	0	0	0	0	0	0	0
	W																
TCDSR	R	DS_CH[127]	DS_CH[126]	DS_CH[125]	DS_CH[124]	0	0	DS_CH[121]	DS_CH[120]	0	0	0	0	0	0	0	0
	W																
	R	0	0	DS_CH[109]	0	0	0	DS_CH[105]	DS_CH[104]	0	0	0	0	0	0	0	0
	W																
TCNCMR	R	NCE_CH[127]	NCE_CH[126]	NCE_CH[125]	NCE_CH[124]	0	0	NCE_CH[121]	NCE_CH[120]	0	0	0	0	0	0	0	0
	W																
	R	0	0	NCE_CH[109]	0	0	0	NCE_CH[105]	NCE_CH[104]	0	0	0	0	0	0	0	0
	W																

Table 340. SARADC_B 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_CH[127]	JCE_CH[126]	JCE_CH[125]	JCE_CH[124]	0	0	JCE_CH[121]	JCE_CH[120]	0	0	0	0	0	0	0	0
	W																
	R	0	0	JCE_CH[109]	0	0	0	JCE_CH[105]	JCE_CH[104]	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	0	WSEL_CH126		0	0	WSEL_CH125		0	0	WSEL_CH124	
	W																
	R	0	0	0	0	0	0	0	0	0	0	WSEL_CH121		0	0	WSEL_CH120	
	W																
TCWENR	R	WEN_CH[127]	WEN_CH[126]	WEN_CH[125]	WEN_CH[124]	0	0	WEN_CH[121]	WEN_CH[120]	0	0	0	0	0	0	0	0
	W																
	R	0	0	WEN_CH[109]	0	0	0	WEN_CH[105]	WEN_CH[104]	0	0	0	0	0	0	0	0
	W																

Table 340. SARADC_B 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_CH[127]	AWOR_CH[126]	AWOR_CH[125]	AWOR_CH[124]	0	0	AWOR_CH[121]	AWOR_CH[120]	0	0	0	0	0	0	0	0
	W	w1c	w1c	w1c	w1c			w1c	w1c								
	R	0	0	AWOR_CH[109]	0	0	0	AWOR_CH[105]	AWOR_CH[104]	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																

33.2.8.2 SARADC_0 register definitions

Table 341. 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	NSTART	NTRGEN	NEDGESEL		JSTART	JTRGEN	JEDGESEL		JTRGSEQ	0	0	0
	W																
	R	0	0	0	0	JTRGSEL				ABORT	ABORT	0	FRZ	0	0	0	PWDN
	W																
MSR	R	0	0	0	0	NSTART	0	0	0	JSTART	0	0	0	0	JABORT	0	0
	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	0	0	JEOC	JECH	NEOC	NECH
	W													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_CH[15]	EOC_CH[14]	EOC_CH[13]	EOC_CH[12]	EOC_CH[11]	EOC_CH[10]	EOC_CH[9]	EOC_CH[8]	EOC_CH[7]	EOC_CH[6]	EOC_CH[5]	EOC_CH[4]	EOC_CH[3]	EOC_CH[2]	EOC_CH[1]	EOC_CH[0]
	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	0	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_CH[15]	IM_CH[14]	IM_CH[13]	IM_CH[12]	IM_CH[11]	IM_CH[10]	IM_CH[9]	IM_CH[8]	IM_CH[7]	IM_CH[6]	IM_CH[5]	IM_CH[4]	IM_CH[3]	IM_CH[2]	IM_CH[1]	IM_CH[0]
	W																

Table 341. 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
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_CH[15]	DS_CH[14]	DS_CH[13]	DS_CH[12]	DS_CH[11]	DS_CH[10]	DS_CH[5]	DS_CH[8]	DS_CH[7]	DS_CH[6]	DS_CH[5]	DS_CH[4]	DS_CH[3]	DS_CH[2]	DS_CH[1]	DS_CH[0]
	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_CH[15]	NCE_CH[14]	NCE_CH[13]	NCE_CH[12]	NCE_CH[11]	NCE_CH[10]	NCE_CH[9]	NCE_CH[8]	NCE_CH[7]	NCE_CH[6]	NCE_CH[5]	NCE_CH[4]	NCE_CH[3]	NCE_CH[2]	NCE_CH[1]	NCE_CH[0]
	W																

Table 341. 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
ICJCMR0	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	JCE_CH[15]	JCE_CH[14]	JCE_CH[13]	JCE_CH[12]	JCE_CH[11]	JCE_CH[10]	JCE_CH[9]	JCE_CH[8]	JCE_CH[7]	JCE_CH[6]	JCE_CH[5]	JCE_CH[4]	JCE_CH[3]	JCE_CH[2]	JCE_CH[1]	JCE_CH[0]
	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	0	0	0	PCE	0	CTSEL		0	0	0	0	VALID	OVERW	RESULT	
	W																
	R	0	0	0	0	CDATA											
	W																
ICWSELR0	R	0	WSEL_CH7			0	WSEL_CH6			0	WSEL_CH5			0	WSEL_CH4		
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
ICWSELR1	R	0	0	0	0	0	0	0	0	0	WSEL_CH13			0	WSEL_CH12		
	W																
	R	0	WSEL_CH11			0	WSEL_CH10			0	0	0	0	0	WSEL_CH8		
ICWENR0	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	WEN_CH[15]	WEN_CH[14]	WEN_CH[13]	WEN_CH[12]	WEN_CH[11]	WEN_CH[10]	WEN_CH[9]	WEN_CH[8]	WEN_CH[7]	WEN_CH[6]	WEN_CH[5]	WEN_CH[4]	WEN_CH[3]	WEN_CH[2]	WEN_CH[1]	WEN_CH[0]
	W																

Table 341. 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	AWOR_CH[15]	AWOR_CH[14]	AWOR_CH[13]	AWOR_CH[12]	AWOR_CH[11]	AWOR_CH[10]	AWOR_CH[9]	AWOR_CH[8]	AWOR_CH[7]	AWOR_CH[6]	AWOR_CH[5]	AWOR_CH[4]	AWOR_CH[3]	AWOR_CH[2]	AWOR_CH[1]	AWOR_CH[0]
	W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c

34 Successive Approximation Register Analog-to-Digital Converter (SARADC) Digital Interface

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 SoC 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.

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.

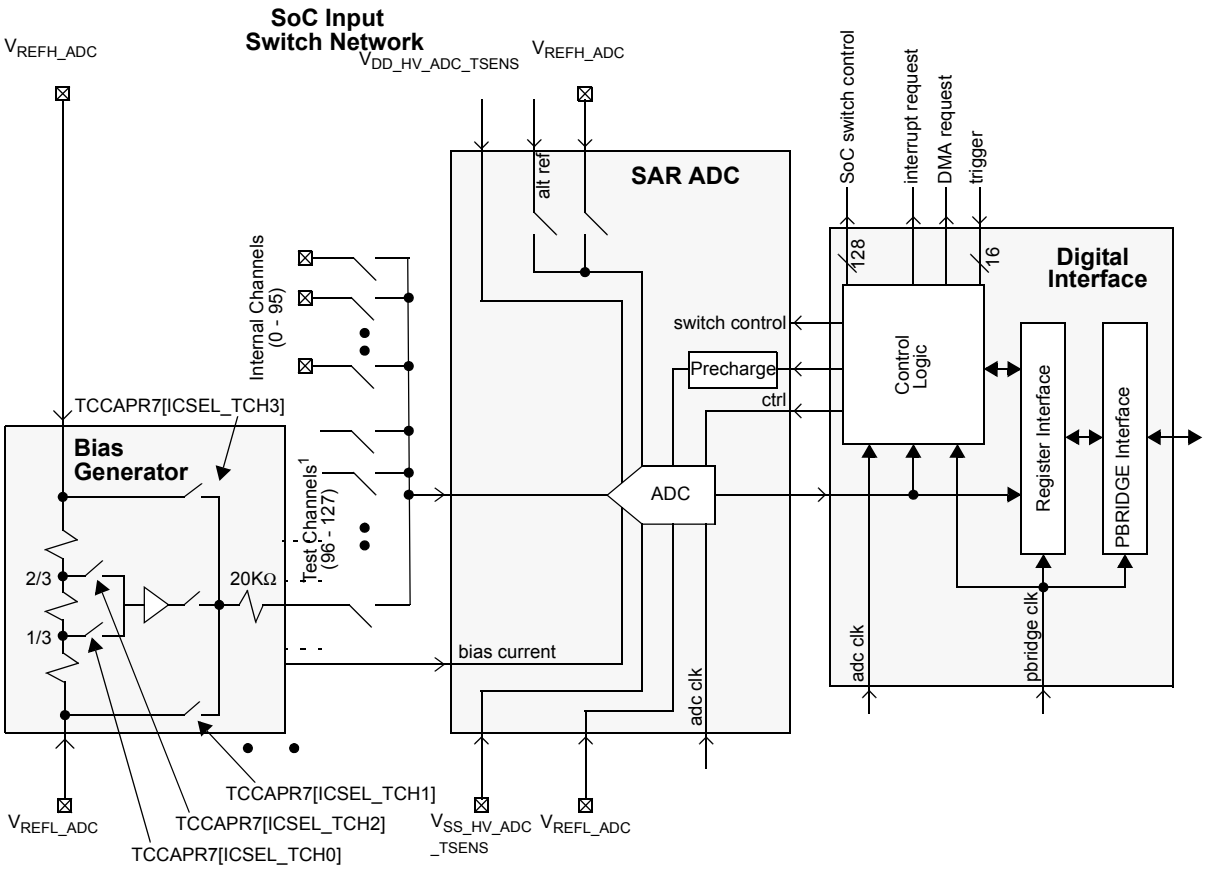
Note: The number of input channels and their mapping depends on the device configuration. See the device-specific Analog-to-Digital Converters (ADC) Configuration section (analog input pin multiplexing and SAR ADC channel assignment) for a channel mapping table.

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 274 shows the SARADC block diagram.

Figure 274. SARADC block diagram



Note:

1. 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
- 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
- Normal conversion with dedicated trigger input
- Injected conversion with dedicated trigger input
- 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:
  - 10-bit or 12-bit conversion result in one half-word
  - 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) controlled by the ctsen parameter
- 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

34.4 Functional description

Two main conversion types are available within the SARADC digital interface:

- Normal conversion
- Injected conversion

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. The normal trigger enable MCR[NTRGEN] bit should be 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.

- By normal trigger
  - A normal trigger valid edge or level is detected to start the conversion.The normal trigger is enabled by setting the NTRGEN bit in MCR and two options are available. A programmed event (rising, falling or both edges depending on NEDGESEL bitfield of MCR) on the normal trigger input starts the normal conversion. If the trigger mode is enabled (MCR[NTRGEN] bit is set) and a conversion is in progress, no conversion request should be given by programming MCR[NSTART] bit.
  - An appropriate gap should be maintained between any two consecutive triggers such that next trigger is received only after the completion of the current conversion. This gap value can calculated as a sum of total conversion time as programmed using CTR0-3 registers for each channel in the chain.

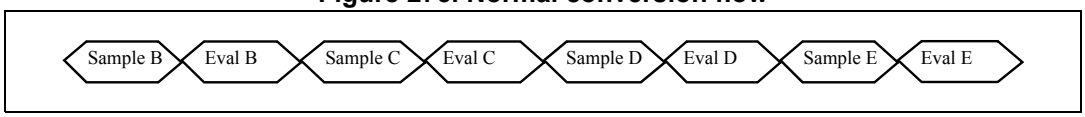
34.4.1.2 Normal conversion operating modes

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 275*.

Figure 275. 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.

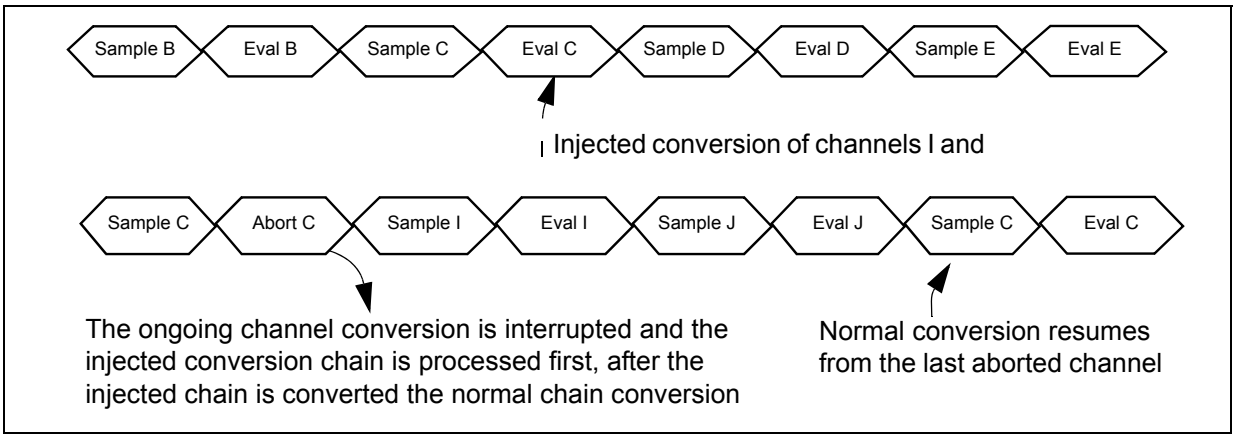
For Example, channels A-B-C-D-E-F-G-H are present in the device where channels B-D-E are to be converted in the *Scan Mode*. MODE = 1 is set for Scan Mode. Conversion starts from the channel B followed by conversion of the channels D-E. At the end of conversion of channel E the scanning of channel B starts followed by conversion of the channels D-E. This sequence repeats itself until the MCR[NSTART] bit is reset by software.

In both the 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, *Figure 276*.

Figure 276. 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.

- By injection trigger
  - The injection trigger is enabled by setting the JTRGEN bit in MCR. A programmed event (rising, falling or both edges depending on JEDGESEL bitfield of MCR) on the injection external trigger starts the injected conversion and also sets the MSR[JSTART] bit. At the end of the chain, the MSR[JSTART] bit is reset and the normal chain conversion is resumed.
  - When JTRGSEQ bit is set, the injection trigger sequence feature is enabled. In this mode, each valid edge of injection trigger input will convert one analog channel. First valid edge will convert the first analog channel enabled through ICJCMR0–2, TCJCMR, ECJCMR0–3 registers. The next valid edge converts the next enabled analog input and so on. Once all the enabled channels are converted, the channel selection loops back to the first enabled channel for next trigger edge. If the JTRGSEQ is reset, the channel selection is reset again to first channel enabled through ICJCMR0–2, TCJCMR, ECJCMR0–3 registers. If JTRGSEQ is reset before completing the injected chain, the remaining channels are completed in one-shot after receiving next valid trigger edge.An appropriate gap should be maintained between any two consecutive injected triggers such that next injected trigger is received only after the completion of ongoing injected conversion. This gap value can calculated as a sum of total conversion time as programmed using CTR0-3 registers for each channel in the injected chain.

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, both injected chain and Normal conversion chain are aborted (both the NSTART and JSTART bits are also reset).

Note: ABORT/ABORTCHAIN functions are not supported during CTU initiated conversions.

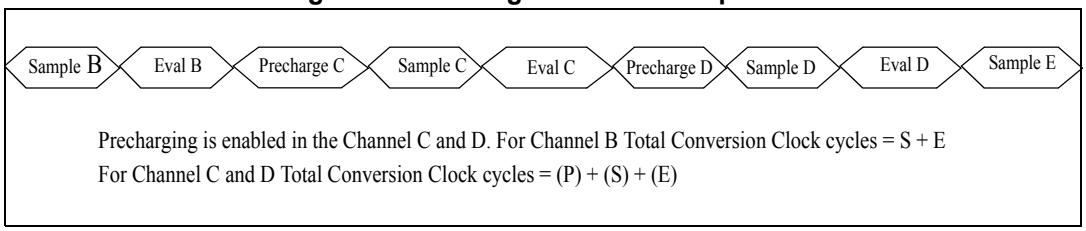
34.4.4 Analog conversion timings and reference selection

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 277*. Please refer to device datasheet for the minimum precharge and sample durations required.

Figure 277. 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'. 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} = 12 * t_{ck}

t_{eval} = 10 * t_{ck} (for 10-bit conversion)

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} refers to reciprocal of f_{ck}, where f_{ck} = SARADC peripheral clock. Total conversion time is < 5µs.

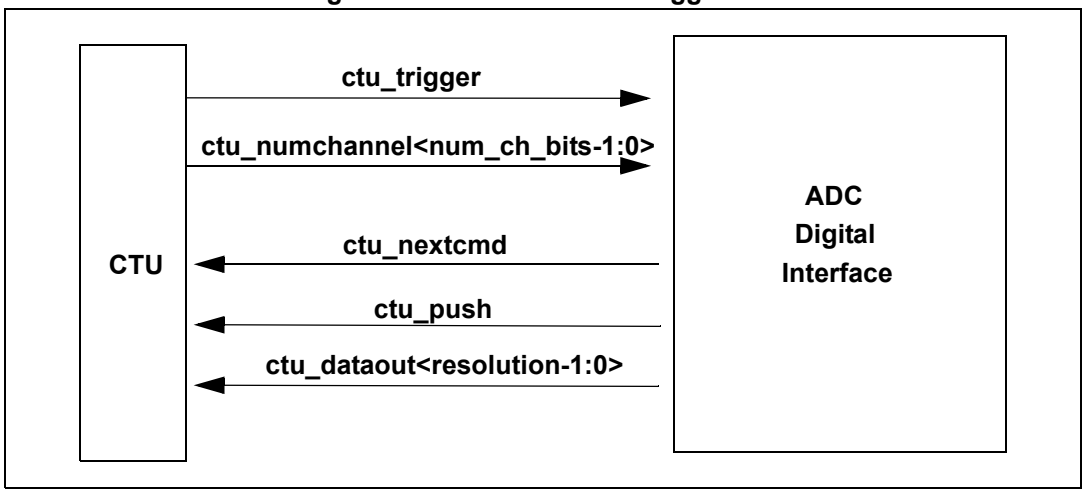
Note: The current implementation of ADCDIG does not support hardware restriction over values of PRECHG and INPSAMP fields. As minimum values for these fields are product specific and need to be managed through software.

34.4.5 CrossTriggering Unit (CTU)

The CrossTriggering 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 278* 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 278. SARADC crosstrigger unit



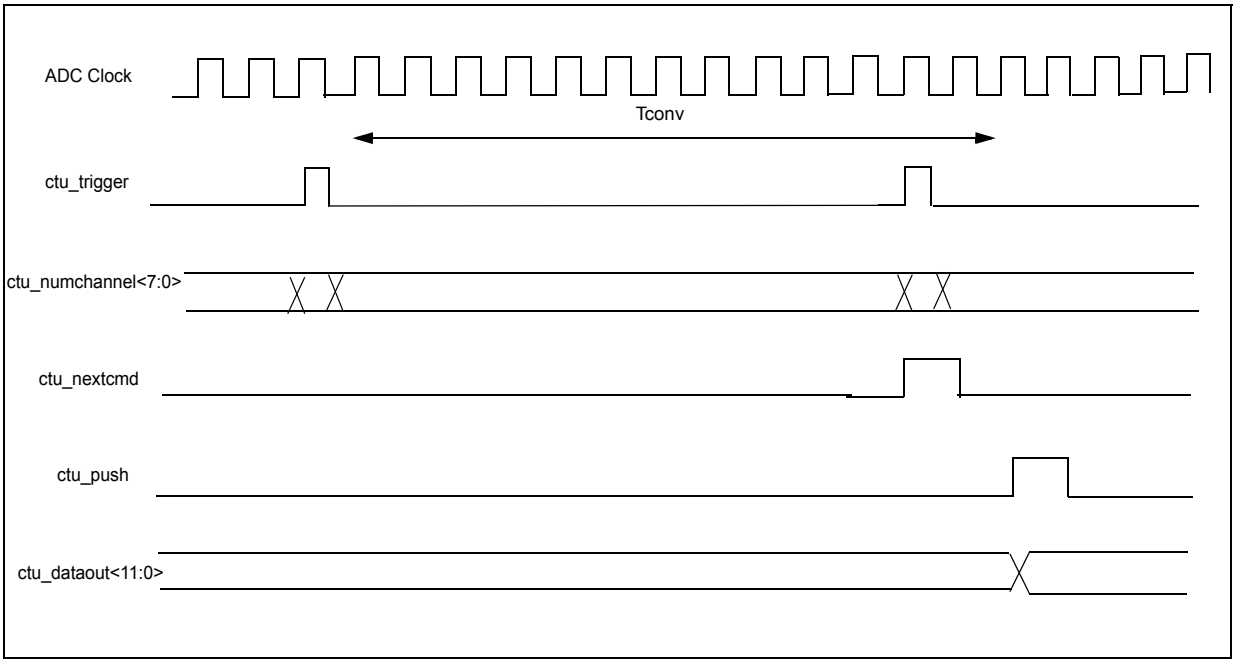
Different CTU modes can be selected programming the CTUEN and CTU_MODE fields of the MCR and configuring the CTSEN parameter as described in the following table.

Table 342. CTU modes

CTSEN	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

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 279*.

Figure 279. 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 JABORT 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. Refer ADC configuration for detailed mapping/control description of Self Test feature.

It is also possible to perform test channel conversions with and without shorting the normal analog channel pin with the internal test voltage. The test channel to internal channel mapping is enabled through top level synthesis parameter testch_short_intch vector. If shorting of a particular test channel has to be disabled then the parameter should be set to ‘0’ for the corresponding test channel. This will result in removal of corresponding TCCAPR registers (see ADCDig IG for more details on testch_short_intch). For implemented TCCAPR registers to enable the short between the test channel and any internal analog channel, ESIC_TCHx bit of TCCAPR0–7 registers should be set. The mapping of thetest channel to any internal analog channel is determined by the ICSEL_TCHx bitfield of TCCAPR0–7 registers.

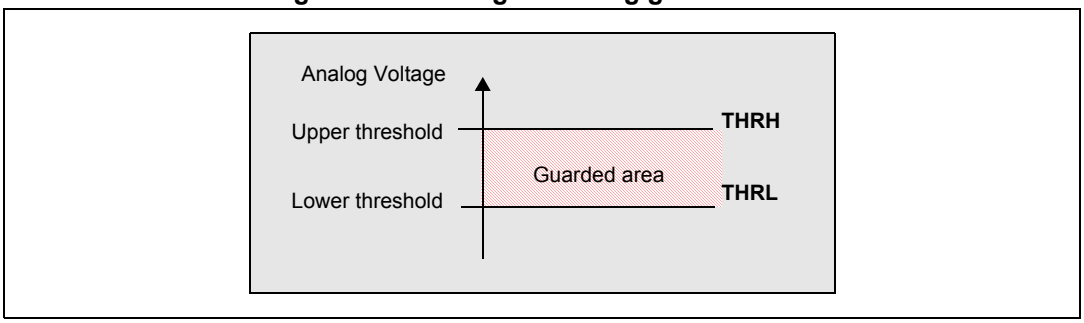
For analog test channel assignment, refer to the device configuration or ADC configuration section.

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 280*) bounded by upper and a lower threshold values named THRH and THRL respectively.

Figure 280. 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 343*. Depending on the MSKWDGxL and MSKWDGxH mask bits in the WTIMR, an interrupt is generated on threshold violation.

Table 343. 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 at any time by setting the PWDN bit in the MCR. If a conversion is ongoing, the SARADC cannot enter power-down mode until the conversion completes. Otherwise, the ongoing operation should be aborted manually by resetting the NSTART bit and using the ABORTCHAIN bit.

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 powerdown mode the application has to wait for the end of current CTU conversion (CTUSTART bit automatically reset when conversion finishes). Additionally, when SARADC is in CTU control Mode the application need to exit CTU control mode(by resetting the CTUEN bit) before requesting 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 request while ctu triggered conversions are ongoing.

34.5 Memory map and registers

This section provides memory map for the SARADC digital interface.

Note: The actual availability and number of registers and their configuration are chip-specific. For this information, see "Device Configuration" chapter.

Table 344. SARADC digital interface register map

Offset	Register	Register classification (1)	Location
0x000	Main Configuration Register (MCR)	Saf-Relv	*34.5.1.1 on page 675*
0x004	Main Status Register (MSR)	No-Saf	*34.5.1.2 on page 678*
0x008–0x00C	Reserved		
0x010	Interrupt Status Register (ISR)	No-Saf	*34.5.1.3 on page 679*
0x014–0x01C	Internal channel Interrupt Pending Registers 0–2 (ICIPR0–ICIPR2)	Saf-Relv	*34.5.1.4 on page 680*
0x020	Interrupt Mask Register (IMR)	Saf-Relv	*34.5.1.5 on page 681*
0x024–0x02C	Internal Channel Interrupt Mask Registers 0–2 (ICIMR0–ICIMR2)	Saf-Relv	*34.5.1.6 on page 682*
0x030	Watchdog Threshold Interrupt Status Register (WTISR)	Saf-Relv	*34.5.1.7 on page 683*
0x034	Watchdog Threshold Interrupt Mask Register (WTIMR)	Saf-Relv	*34.5.1.8 on page 684*
0x038–0x03C	Reserved		
0x040	DMA Enable Register (DMAE)	Saf-Relv	*34.5.1.9 on page 685*
0x044–0x04C	Internal Channel DMA Select Registers 0–2 (ICDSR0–ICDSR2)	Saf-Relv	*34.5.1.10 on page 686*
0x050–0x05C	Reserved		
0x060–0x06C	Watchdog Threshold Registers 0–3 (WTHRHLR0–WTHRHLR3)	Saf-Relv	*34.5.1.11 on page 687*
0x070–0x090	Reserved		
0x094–0x0A0	Conversion Timing Registers 0–3 (CTR0–3)	Saf-Relv	*34.5.1.12 on page 688*
0x0A4–0x0AC	Internal Channel Normal Conversion Mask Registers 0–2 (ICNCMR0–ICNCMR2)	Saf-Relv	*34.5.1.13 on page 688*
0x0B0	Reserved		
0x0B4–0x0BC	Internal Channel Injected Conversion Mask Registers 0–2 (ICJCMR0–ICJCMR2)	Saf-Relv	*34.5.1.14 on page 689*
0x0C0–0x0C4	Reserved		
0x0C8	Power Down Exit Delay Register (PDEDR)	Saf-Relv	*34.5.1.15 on page 691*
0x0CC–0x0FC	Reserved		
0x100–0x27C	Internal Channel Data Registers 0–95 (ICDR0–ICDR95)	Saf-Relv	*34.5.1.16 on page 691*

Table 344. SARADC digital interface register map(Continued)

Offset	Register	Register classification (1)	Location
0x280–0x2AC	Reserved		
0x2B0–0x2DC	Internal Channel Watchdog Selection Registers 0–11 (ICWSELR0–ICWSELR11)	Saf-Relv	*34.5.1.17 on page 693*
0x2E0–0x2E8	Internal Channel Watchdog Enable Registers 0–2 (ICWENR0–ICWENR2)	Saf-Relv	*34.5.1.18 on page 694*
0x2EC	Reserved		
0x2F0–0x2F8	Internal Channel Analog Watchdog Out of Range Registers 0–2 (ICAWORR0–ICAWORR2)	No-Saf	*34.5.1.19 on page 695*
0x2FC–0x3FC	Reserved		
0x400	Test Channel Interrupt Pending Register (TCIPR)	Saf-Relv	*34.5.1.20 on page 696*
0x404	Test Channel Interrupt Mask Register (TCIMR)	Saf-Lat	*34.5.1.21 on page 697*
0x408	Test Channel DMA Select Register (TCDSR)	Saf-Lat	*34.5.1.22 on page 698*
0x40C	Test Channel Normal Conversion Mask Register (TCNCMR)	Saf-Lat	*34.5.1.23 on page 698*
0x410	Test Channel Injected Conversion Mask Register (TCJCMR)	Saf-Lat	*34.5.1.24 on page 699*
0x414–0x420	Test Channel Watchdog Selection Registers 0–3 (TCWSELR0–TCWSELR3)	Saf-Lat	*34.5.1.25 on page 700*
0x424	Test Channel Watchdog Enable Register (TCWENR)	Saf-Relv	*34.5.1.26 on page 701*
0x428	Test Channel Analog Watchdog Out of Range Register (TCAWORR)	Saf-Relv	*34.5.1.27 on page 702*
0x42C	Reserved		
0x430–0x44C	Test Channel Connection with Analog Pin Registers 0–7 (TCCAPR0–TCCAPR7)	Saf-Relv	*34.5.1.28 on page 703*
0x450–0x4CC	Test Channel Data Registers 96-127 (TCDR96-TCDR127)	Saf-Relv	*34.5.1.29 on page 705*
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	XSTRTEN	NSTART	NTRGEN(2)	NEDGESEL(2)		JSTART	JTRGEN(2)	JEDGESEL(2)		JTRGSEQ(2)	0	CTUEN	CTU_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	0	0	JTRGSEL				ABORT CHAIN	ABORT	0	FRZ	0	0	0	PWDN
W																
Reset	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

2. These bits may be written, but will have no effect on the device.

Figure 281. Main Configuration Register (MCR)

Table 345. 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.
3	Reserved
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
5 NTRGEN	Normal trigger enable 0 Normal trigger disabled to start a normal conversion. 1 Normal trigger enabled to start a normal conversion.

Table 345. MCR field descriptions(Continued)

Field	Description
6–7 NEDGESEL[1:0]	Normal trigger edge selection 00 Falling edge of normal trigger is selected. 01 Rising edge of normal trigger is selected. 1*x* Both rising and falling edges of normal trigger are selected. (*x* = 0,1)
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
9 JTRGEN	Injection trigger enable 0 Injection trigger disabled for channel injection (injected conversion cannot be started using an injection trigger) 1 Injection trigger enabled for channel injection
10–11 JEDGESEL[1:0]	Injection trigger edge selection 00 Falling edge of injection trigger is selected. 01 Rising edge of injection trigger is selected. 1*x* Both rising and falling edges of injection trigger are selected. (*x* = 0,1)
12 JTRGSEQ	Injection trigger sequence enable 0 Injection trigger sequence mode is disabled. 1 Injection trigger sequence mode is enabled.
13	Reserved Write of any value has no effect; read value is always 0.
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. 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.
16–19	Reserved Write of any value has no effect; read value is always 0.
20–23 JTRGSEL[3:0]	Injection Trigger Input Selection This bitfield selects which trigger input to be selected to start injected conversion. 0000 Trigger input 0 is selected 0001 Trigger input 1 is selected ... 1111 Trigger input 15 is selected.

Table 345. MCR field descriptions(Continued)

Field	Description
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.
26	Reserved
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, SDADC 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.
28–30	Reserved
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	CTU START(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 282. Main Status Register (MSR)

Table 346. MSR field descriptions

Field	Description
0–3	Reserved Write of any value has no effect; read value is always 0.
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.
5–7	Reserved Write of any value has no effect; read value is always 0.
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.
9–12	Reserved Write of any value has no effect; read value is always 0.
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.
14	Reserved Write of any value has no effect; read value is always 0.

Table 346. MSR field descriptions(Continued)

Field	Description
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[7:0]	Channel under measure address This status bit is used to signal which channel is under measure.
24–28	Reserved
29–31 ADCSTATUS[2:0]	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 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	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 283. Interrupt Status Register (ISR)

Table 347. ISR field descriptions

Field	Description
0–25	Reserved Write of any value has no effect; read value is always 0.
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.
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 Registers 0–2 (ICIPR0–ICIPR2)

The interrupt channel register to channel association is described in *Table 349*.

Offset 0x014–0x01C Access: User Read Only

	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 284. Internal Channel Interrupt Pending Registers (ICIPR0–ICIPR2)

Table 348. ICIPR0–ICIPR2 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 349. ICIPR0–2 Registers to channel association

Register	Register bits 31:0
ICIPR0	EOC_CH[31:0]
ICIPR1	EOC_CH[63:32]
ICIPR2	EOC_CH[95:64]

Parameter num_intch vector has positional inference for registers ICIPR0–2, such that:

- num_intch(i) (i = 0–31) corresponds to ICIPR0 bit[0:31].
- num_intch(i) (i = 32–63) corresponds to ICIPR1 bit[0:31].
- num_intch(i) (i = 64–95) corresponds to ICIPR2 bit[0:31].

If any num_intch(i) value is ‘0’, corresponding bit is not implemented and read access returns ‘0’ on that bit location. Thus:

- If num_intch[31:0] is all 0’s, register ICIPR0 is not implemented.
- If num_intch[63:32] is all 0’s, register ICIPR1 is not implemented.
- If num_intch[95:64] is all 0’s, register ICIPR2 is not implemented.

In any or all of the above circumstances, the corresponding registers are treated as reserved space 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(1)	MSKEOCTU(2)	MSKJEOC	MSKJECH	MSKNEOC	MSKNECH
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. Not funtionally used, but this bit can be written by software.
2. Only available with cross-triggering unit (CTU) feature.

Figure 285. Interrupt Mask Register (IMR)

Table 350. IMR field descriptions

Field	Description
0–26	Reserved Write of any value has no effect; read value is always 0.
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 Registers 0–2 (ICIMR0–ICIMR2)

The interrupt mask register to channel association is described in *Table 352*.

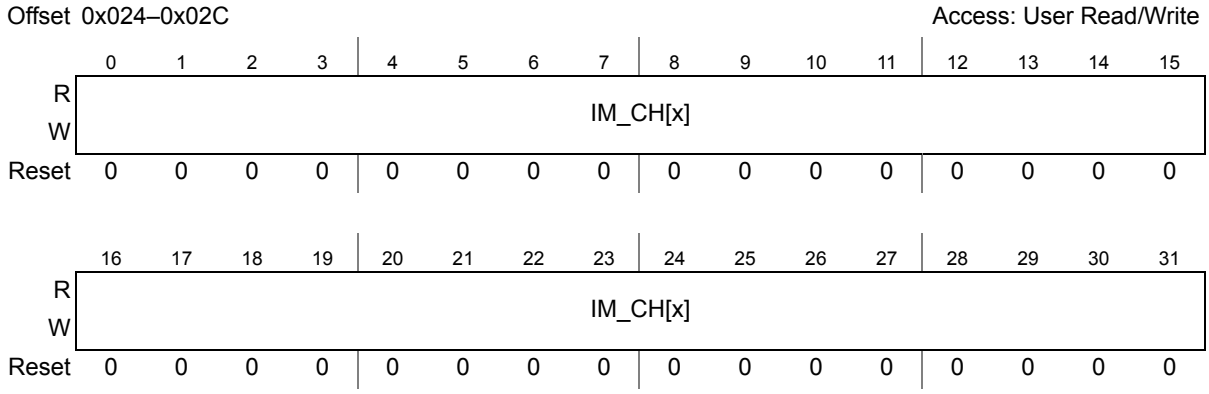


Figure 286. Internal Channel Interrupt Mask Registers (ICIMR0–ICIMR2)

Table 351. ICIMR0–ICIMR2 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 352. Interrupt Mask Registers to Channel Association

Register	Register bits 31:0
ICIMR0	IM_CH[31:0]
ICIMR1	IM_CH[63:32]
ICIMR2	IM_CH[95:64]

Parameter num_intch vector has positional inference for registers ICIMR0–2, such that:

- num_intch(i) (i = 0–31) corresponds to ICIMR0 bit[0:31].
- num_intch(i) (i = 32–63) corresponds to ICIMR1 bit[0:31].
- num_intch(i) (i = 64–95) corresponds to ICIMR2 bit[0:31].

If any of the num_intch(i) value is ‘0’ corresponding bit is not implemented and read access would return ‘0’ on that bit location. Thus:

- If num_intch[31:0] is all 0’s, register ICIMR0 is not implemented.
- If num_intch[63:32] is all 0’s, register ICIMR1 is not implemented.
- If num_intch[95:64] is all 0’s, register ICIMR2 is not implemented.

In any or all of the above circumstances, the corresponding registers are treated as reserved space 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 Only

	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 287. Watchdog Threshold Interrupt Status Register (WTISR)

Table 353. WTISR field descriptions

Field	Description
2x+1 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.
Bit 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.

Note: If num_watchdog generic parameter is zero, this register is not implemented.

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 W	MSKWDG15H	MSKWDG15L	MSKWDG14H	MSKWDG14L	MSKWDG13H	MSKWDG13L	MSKWDG12H	MSKWDG12L	MSKWDG11H	MSKWDG11L	MSKWDG10H	MSKWDG10L	MSKWDG9H	MSKWDG9L	MSKWDG8H	MSKWDG8L
Reset	0	0	0	0	0	0	0	0	0	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	MSKWDG7H	MSKWDG7L	MSKWDG6H	MSKWDG6L	MSKWDG5H	MSKWDG5L	MSKWDG4H	MSKWDG4L	MSKWDG3H	MSKWDG3L	MSKWDG2H	MSKWDG2L	MSKWDG1H	MSKWDG1L	MSKWDG0H	MSKWDG0L
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 288. Watchdog Threshold Interrupt Mask Register (WTIMR)

Table 354. WTIMR field descriptions

Field	Description
2x+1 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.
Bit 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.

Note: If num_watchdog generic parameter is zero, this register is not implemented.

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 289. DMA Enable Register (DMAE)

Table 355. DMAE register field descriptions

Field	Description
0–29	Reserved Write of any value has no effect; read value is always 0.
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 Registers 0–2 (ICDSR0–ICDSR2)

The DMA select registers to channel association is described in *Table 357*.

Offset 0x044–0x04C Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DS_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	DS_CH[x]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 290. Internal Channel DMA Select Register (ICDSR0–ICDSR2)

Table 356. ICDSR0–ICDSR2 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 357. Internal Channel DMA Select Registers to Channel Association

Register	Register bits 31:0
ICDSR0	DS_CH[31:0]
ICDSR1	DS_CH[63:32]
ICDSR2	DS_CH[95:64]

34.5.1.11 Watchdog Threshold Registers 0–3 (WTHRHLR0–WTHRHLR3)

Offset 0x060–0x06C — 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 291. Watchdog Threshold Registers 0–3 (WTHRHLR0–WTHRHLR3)

Table 358. WTHRHLR0–WTHRHLR3 field descriptions

Field	Description
0–3	Reserved Write of any value has no effect; read value is always 0.
4–15 THRH	High threshold value for channel x
16–19	Reserved Write of any value has no effect; read value is always 0.
20–31 THRL	Low threshold value for channel x

The number of WTHRHLR registers depends on num_watchdog generic parameter configuration. If num_watchdog is zero, these registers are not implemented.

The length of the threshold field depends on the generic resolution.

34.5.1.12 Conversion Timing Registers 0–3 (CTR0–CTRL3)

Offset 0x094–0x0A0 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 292. Conversion Timing Registers 0–3 (CTR0–CTRL3)

Table 359. CTR0–CTRL3 field descriptions

Field	Description
0 CRES	Conversion Resolution select This bit selects the conversion resolution for the conversion. 0 12-bit resolution 1 10-bit resolution
1–19	Reserved Write of any value has no effect; read value is always 0.
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 Registers 0–2 (ICNCMR0–ICNCMR2)

The normal conversion mask registers to channel association is described in *Table 361*.

Offset 0x0A4–0x0AC 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 293. Internal Channel Normal Conversion Mask Registers (ICNCMR0–ICNCMR2)

Table 360. ICNCMR0–ICNCMR2 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 361. Internal Channel Normal Conversion Mask Registers to Channel Association

Register	Register bits 31:0
ICNCMR0	NCE_CH[31:0]
ICNCMR1	NCE_CH[63:32]
ICNCMR2	NCE_CH[95:64]

Parameter num_intch vector has positional inference for registers ICNCMR0–2, such that:

- num_intch(i) (i = 0–31) corresponds to ICNCMR0 bit[0:31]
- num_intch(i) (i = 32–63) corresponds to ICNCMR1 bit[0:31]
- num_intch(i) (i = 64–95) corresponds to ICNCMR2 bit[0:31].

If any of the num_intch(i) value is ‘0’ corresponding bit is not implemented and read access would return ‘0’ on that bit location. Thus:

- If num_intch[31:0] is all 0’s, register ICNCMR0 is not implemented.
- If num_intch[63:32] is all 0’s, register ICNCMR1 is not implemented.
- If num_intch[95:64] is all 0’s, register ICNCMR2 is not implemented.

In any or all of the above circumstances, the corresponding registers are treated as reserved space and any attempt to access this space generates a transfer error.

34.5.1.14 Internal Channel Injected Conversion Mask Registers 0–2 (ICJCMR0–ICJCMR2)

The injected conversion mask registers to channel association is described in *Table 363*.

Offset 0x0B4–0x0BC 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 294. Internal Channel Injected Conversion Mask Registers (ICJCMR0–ICJCMR2)

Table 362. ICJCMR0–ICJCMR2 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 363. Internal Channel Normal Conversion Mask Registers to Channel Association

Register	Register bits 31:0
ICJCMR0	JCE_CH[31:0]
ICJCMR1	JCE_CH[63:32]
ICJCMR2	JCE_CH[95:64]

Parameter num_intch vector has positional inference for registers ICJCMR0–2, such that:

- num_intch(i) (i = 0–31) corresponds to ICJCMR0 bit[0:31].
- num_intch(i) (i = 32–63) corresponds to ICJCMR1 bit[0:31].
- num_intch(i) (i = 64–95) corresponds to ICJCMR2 bit[0:31].

If any of the num_intch(i) value is ‘0’ corresponding bit is not implemented and read access returns ‘0’ on that bit location. Thus:

- If num_intch[31:0] is all 0’s, register ICJCMR0 is not implemented
- If num_intch[63:32] is all 0’s, register ICJCMR1 is not implemented
- if num_intch[95:64] is all 0’s, register ICJCMR2 is not implemented.

In any or all of the above circumstances, the corresponding registers are treated as reserved space 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 295. Power Down Exit Delay Register (PDEDR)

Table 364. PDEDR field descriptions

Field	Description
0–23	Reserved Write of any value has no effect; read value is always 0.
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 Registers 0–95 (ICDR0–ICDR95)

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 and some control information to select the required reference voltage, timing parameter selection for each channel.

Offset 0x100–0x27C 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[15:0]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 296. Internal Channel Data Registers (ICDR0–ICDR95)

Table 365. ICDR0–ICDR95 field descriptions

Field	Description
0-3	Reserved
4 PCE	Precharge Enable This bit enables the precharging phase during channel conversion. 0 Precharge phase is disabled 1 Precharge phase is enabled
5	Reserved. Write of any value has no effect; read value is always zero.
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
8–10	Reserved Write of any value has no effect; read value is always 0.
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.
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

Table 365. ICDR0–ICDR95 field descriptions(Continued)

Field	Description
14–15 RESULT[1:0]	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.

The length of the useful converted data out of 16 bits of CDATA field also depends on the generic *resolution* and *conv_mode_select*.

As each of ICDRx corresponds to an internal channel inferred through generic num_intch. If any of the num_intch(i) values is ‘0’, the corresponding ICDRx is not implemented and any attempt to access this space generates a transfer error.

This register should be accessed via 32-bit R/W.

34.5.1.17 Internal Channel Watchdog Select Registers (ICWSELR0–ICWSELR11)

These registers contain WSEL_CHx[3:0] 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 367*.

Offset 0x2B0–0x2DC — Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	WSEL_CH7				WSEL_CH6				WSEL_CH5				WSEL_CH4			
Reset	0	0	0	0	0	0	0	0	0	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_CH3				WSEL_CH2				WSEL_CH1				WSEL_CH0			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 297. Internal Channel Watchdog Select Registers (ICWSELR0–ICWSELR11)

Table 366. ICWSELR0–ICWSELR11 field descriptions

Field	Description
4*x+3 : 4*x WSEL_CH[x]	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 367. Internal Channel Watchdog Enable Registers to Channel Association

Register	Register bits 31:0
ICWSELRx (x = 0 to 11)	WSEL_CH[(8*x)+7] .. WSEL_CH[8*x]

Each such nibble (WSEL_CHx) has positional correspondence to num_intch parameter (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 covering all 96 internal channels.

If the value of any num_intch(i) is ‘0’, the corresponding WSEL_CHx nibble is not implmented. For the implemented nibbles, width of WSEL_CHx field within the nibble is dependent on num_wdg parameter For e.g. if num_wdg 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 Registers 0–2 (ICWENR0–ICWENR2)

The Watchdog enable registers to channel association is described in *Table 369*.

Offset “0x2E0–0x2E8 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 298. Internal Channel Watchdog Enable Registers (ICWENR0–ICWENR2)

Table 368. ICWENR0–ICWENR2 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 369. Internal Channel Watchdog Enable Registers to Channel Association

Register	Register bits 31:0
ICWER0	WEN_CH[31:0]
ICWER1	WEN_CH[63:32]
ICWER2	WEN_CH[95:64]

Parameter num_intch vector has positional inference for registers ICWER0–2, such that:

- num_intch(i) (i = 0–31) corresponds to ICWER0 bit[0:31]
- num_intch(i) (i = 32–63) corresponds to ICWER1 bit[0:31]
- num_intch(i) (i = 64–95) corresponds to ICWER2 bit[0:31]

If any num_intch(i) value is '0', the corresponding bit is not implemented and read access returns '0' on that bit location. Thus:

- if num_intch[31:0] is all 0's, register ICWER0 is not implemented.
- if num_intch[63:32] is all 0's, register ICWER1 is not implemented.
- if num_intch[95:64] is all 0's, register ICWER2 is not implemented.

In any or all of the above circumstances, the corresponding registers are treated as reserved space and any attempt to access this space generates a transfer error.

34.5.1.19 Internal Channel Analog Watchdog Out of Range Registers 0–2 (ICAWORR0–ICAWORR2)

The analog watchdog registers to channel association is described in *Table 371*.

Offset 0x2F0–0x2F8 Access: User Read Only

	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 299. Internal Channel Analog Watchdog Out of Range Registers (ICAWORR0–ICAWORR2)

Table 370. ICAWORR0–ICAWORR2 field descriptions

Field	Description
0–31 AWOR_CH[x]	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.

Note: In case either num_watchdog is zero, these registers are not implemented.

Table 371. Internal Channel Analog Watchdog Registers to Channel Association

Register	Register bits 31:0
ICAWORR0	AWOR_CH[31:0]
ICAWORR1	AWOR_CH[63:32]
ICAWORR2	AWOR_CH[95:64]

Parameter num_intch vector has positional inference for registers ICAWORR0–2, such that:

- num_intch(i) (i = 0–31) corresponds to ICAWORR0 bit [0:31].
- num_intch(i) (i = 32–63) corresponds to ICAWORR1 bit [0:31].
- num_intch(i) (i = 64–95) corresponds to ICAWORR2 bit [0:31].

If any of the num_intch(i) value is ‘0’, the corresponding bit is not implemented and read access returns ‘0’ on that bit location. Thus:

- If num_intch[31:0] is all 0’s, register ICAWORR0 is not implemented.
- If num_intch[63:32] is all 0’s, register ICAWORR1 is not implemented.
- If num_intch[95:64] is all 0’s, register ICAWORR2 is not implemented.

In any or all of the above circumstances, the corresponding registers are treated as reserved space 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 373*.

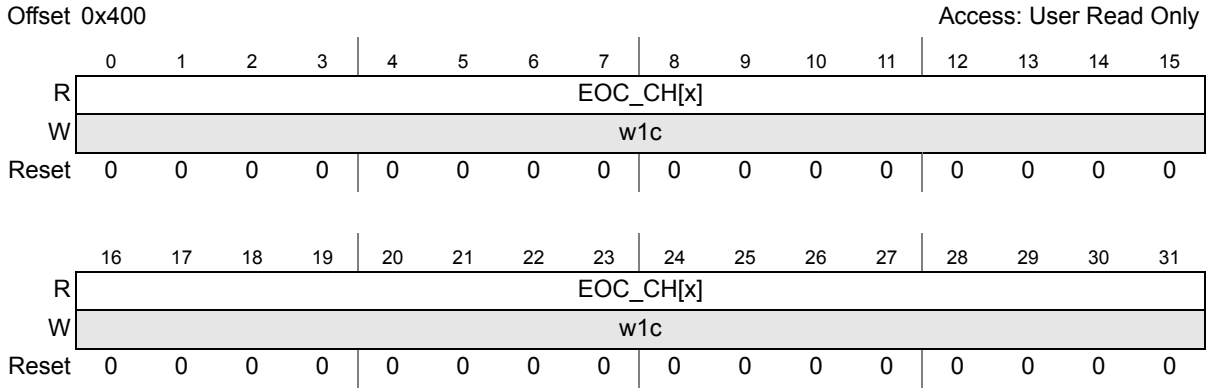


Figure 300. Test Channel Interrupt Pending Register (TCIPR)

Table 372. 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 373. TCIPR Register to Channel Association

Register	Register bits 31:0
TCIPR	EOC_CH[127:96]

Parameter num_testch vector has positional inference for TCIPR such that num_testch(i) (i = 0–31) corresponds to TCIPR bit[0:31]. If any of the num_testch(i) value is ‘0’, the corresponding bit is not implemented and read access returns ‘0’ on that bit location.

If num_testch is all 0’s, register TCIPR is not implemented and treated as reserved space. 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 375*.

Offset 0x404 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 301. Test Channel Interrupt Mask Register (TCIMR)

Table 374. TCIMR 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 375. Test Channel Interrupt Mask Register to Channel Association

Register	Register bits 31:0
TCIMR	IM_CH[127:96]

Parameter num_testch vector has positional inference for TCIMR such that num_testch(i) (i = 0–31) corresponds to TCIMR bit[0:31]. If any of the num_testch(i) value is ‘0’, the corresponding bit is not implemented and read access returns ‘0’ on that bit location.

If the complete num_testch is all 0’s, TCIMR is not implemented and treated as reserved space. 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 377*.

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 302. Test Channel DMA Select Register (TCDSR)

Table 376. 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 377. Test Channel DMA Select Registers to Channel Association

Register	Register bits 31:0
TCDSR	DS_CH[127:96]

Parameter num_testch vector has positional inference for TCDSR such that num_testch(i) (i = 0–31) corresponds to TCDSR bit[0:31]. If any of the num_testch(i) value is ‘0’, the corresponding bit is not implemented and read access returns ‘0’ on that bit location.

If the complete num_testch is all 0’s, TCDSR is not implemented and treated as reserved space. 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 379*.

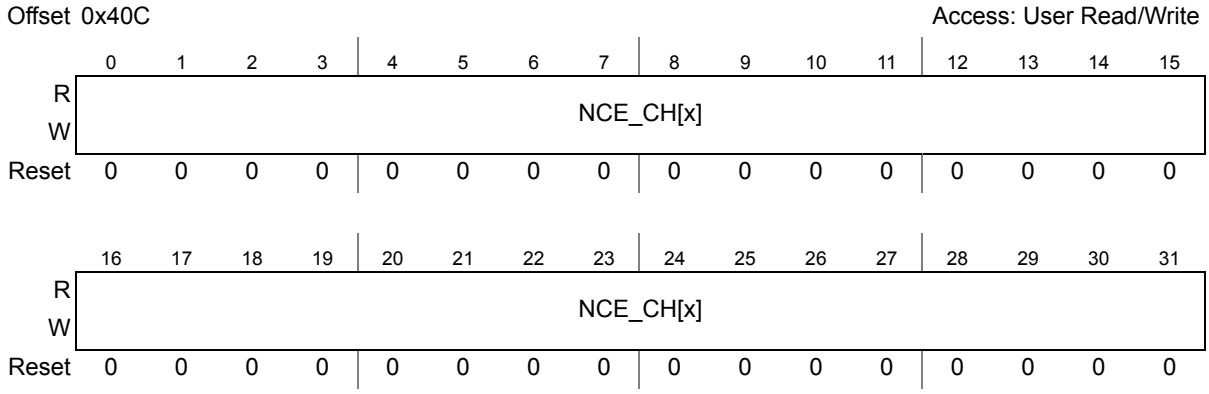


Figure 303. Test Channel Normal Conversion Mask Register (TCNCMR)

Table 378. 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 379. Test Channel Normal Conversion Mask Register to Channel Association

Register	Register bits 31:0
TCNCMR	NCE_CH[127:96]

Parameter num_testch vector has positional inference for TCNCMR such that num_testch(i) (i = 0–31) corresponds to TCNCMR bit[0:31]. If any of the num_testch(i) value is ‘0’, the corresponding bit is not implemented and read access returns ‘0’ on that bit location.

If the complete num_testch is all 0’s, TCNCMR is not implemented and treated as reserved space. 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 381*.

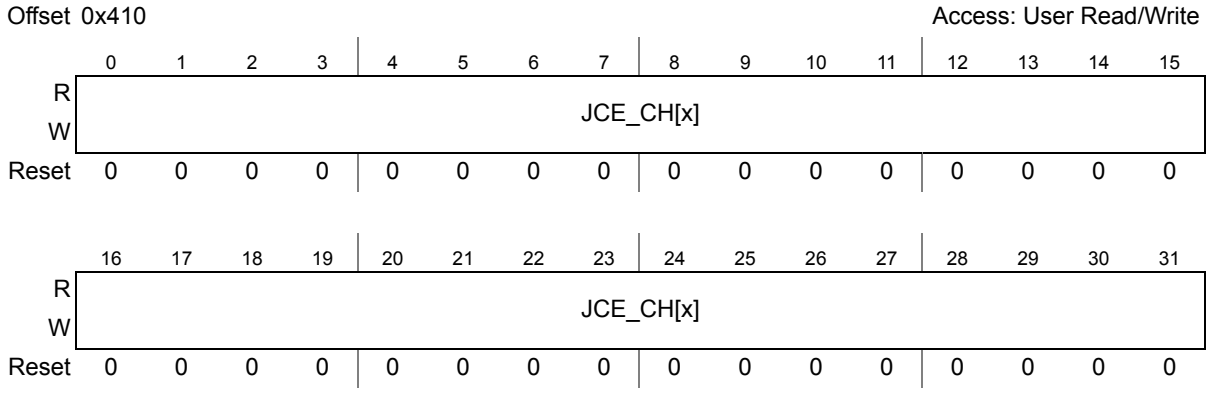


Figure 304. Test Channel Injected Conversion Mask Register (TCJCMR)

Table 380. 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 381. Test Channel Normal Conversion Mask Register to Channel Association

Register	Register bits 31:0
TCJCMR	JCE_CH[127:96]

Parameter num_testch vector has positional inference for registers such that num_testch(i) (i = 0–31) corresponds to TCJCMR bit[0:31]. If any of the num_testch(i) value is ‘0’, the corresponding bit is not implemented and read access returns ‘0’ on that bit location.

If the complete num_testch is all 0’s, TCJCMR is not implemented and treated as reserved space. Any attempt to access this space generates a transfer error.

34.5.1.25 Test Channel Watchdog Select Registers 0–3 (TCWSELR0–TCWSELR3)

This register contains WSEL_CHx[3:0] 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 383*.

Offset 0x414–0x420 Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	WSEL103				WSEL_CH102				WSEL_CH101				WSEL_CH100			
Reset	0	0	0	0	0	0	0	0	0	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_CH99				WSEL_CH98				WSEL_CH97				WSEL_CH96			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 305. Test Channels Watchdog Select Registers 0–3 (TCWSELR0–TCWSELR3)

Table 382. TCWSELR0–TCWSELR3 field descriptions

Field	Description
4*x+3 : 4*x WSEL_CH[x]	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 383. Test Channel Watchdog Select Registers to Channel Association

Register	Register bits 31:0
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 num_watchdog parameter:

- If num_watchdog > 8, the number of bits implemented is 4 (WSEL_CHx[3:0])
- If 4 < num_watchdog <= 8, the number of bits implemented is 3 (WSEL_CHx[3] is reserved).
- If 1 < num_watchdog <= 4, the number of bits implemented is 2 (WSEL_CHx[3:2] is reserved).
- If num_watchdog = 1, only 1 bit is present.

Each such nibble (WSEL_CHx) has positional correspondence to num_testch parameter (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 value of any num_testch(i) is ‘0’, the corresponding WSEL_CHx nibble is not implemented. For the implemented nibbles, width of WSEL_CHx field within the nibble is dependent on num_wdg parameter For e.g. if num_wdg 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 385*.

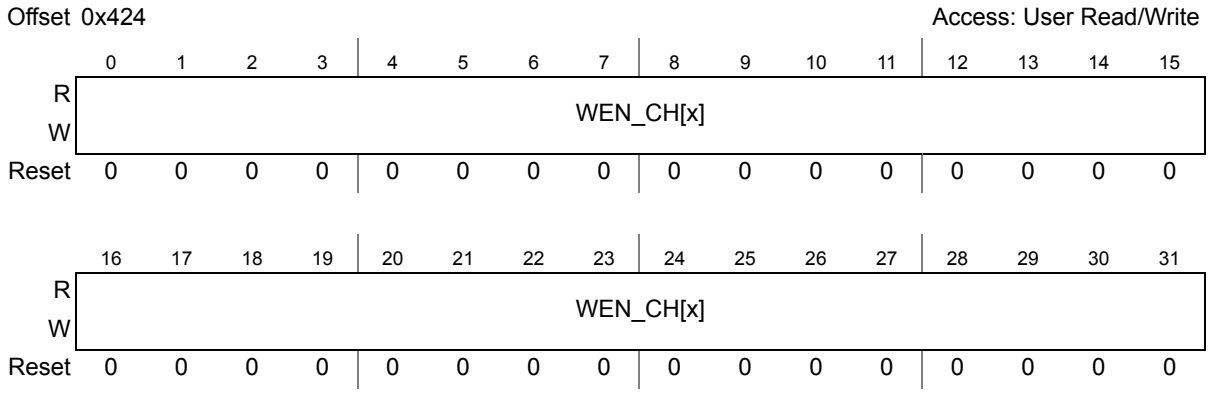


Figure 306. Test Channel Watchdog Enable Register (TCWENR)

Table 384. 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 385. Test Channel Watchdog Enable Registers to Channel Association

Register	Register bits 31:0
TCWENR	WEN_CH[127:96]

Parameter num_testch vector has positional inference for registers TCWENR such that num_testch(i) (i = 0–31) corresponds to TCWENR bit[0:31]. If any of the num_testch(i) value is ‘0’, the corresponding bit is not implemented and read access returns ‘0’ on that bit location.

If the complete num_testch is all 0’s, TCWENR is not implemented and treated as reserved space. 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 387*.

Offset 0x0428 Access: User Read Only

	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 307. Test Channel Analog Watchdog Out of Range Register (TCAWORR)

Table 386. TCAWORR field descriptions

Field	Description
0–31 AWOR_CH[x]	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.

Note: In case either of num_watchdog is zero, these registers are not implemented.

Table 387. Analog Watchdog Register to Channel Association

Register	Register bits 31:0
TCAWORR	AWOR_CH[127:96]

Parameter num_testch vector has positional inference for registers TCAWORR such that num_testch(i) (i = 0–31) corresponds to TCAWORR bit[0:31]. If any of the num_testch(i) value is ‘0’, the corresponding bit is not implemented and read access returns ‘0’ on that bit location.

If the complete num_testch is all 0’s, TCAWORR is not implemented and treated as reserved space. Any attempt to access this space generates a transfer error.

34.5.1.28 Test Channel Connection with Analog Pin Registers 0–7 (TCCAPR0–TCCAPR7)

Each test channel 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 389*.

Offset 0x430–0x44C — Access: User Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	ESIC_TCH3	ICSEL_TCH3							ESIC_TCH2	ICSEL_TCH2						
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	ESIC_TCH1	ICSEL_TCH1							ESIC_TCH0	ICSEL_TCH0						
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 308. Test Channel Connection with Analog Pin Registers 0–7 (TCCAPR0–TCCAPR7)

For analog test channel assignment, refer to the device configuration or ADC configuration section.

Table 388. Test Channel Connection with Analog Pin Registers 0–7 (TCCAPR0–TCCAPR7) field descriptions

Field (Little Endian MSB = bit 31)	Field (Big Endian MSB = bit 0)	Description
Bit 8x+7 ESIC_TCH[x]	Bit 24-8x ESIC_TCH[x]	Enable short with internal channel for test channel x 0 Test channel x cannot be shorted with internal channel 1 Test channel x can be shorted with internal channel
Bit 8x+6 : 8x ICSEL_TCH[x]	Bit 25-8x : 31-8x ICSEL_TCH[x]	Internal channel selection for short with test channel x

Table 389. Test Channel Short Registers to Channel Association

Register	Register bits 31:0 or 0:31 (arrangement of bit fields left to right)
TCCAPRx (x = 0 to 7)	ESIC_TCH[4*x+3], ICSEL_TCH[4*x+3], ESIC_TCH[4*x+2], ICSEL_TCH[4*x+2], ESIC_TCH[4*x+1], ICSEL_TCH[4*x+1], ESIC_TCH[4*x], ICSEL_TCH[4*x]

Parameter num_testch vector has positional inference for registers TCCAPRx, such that:

- num_testch(i) (i = 0–3) corresponds to TCCAPR0.
- num_testch(i) (i = 4–7) corresponds to TCCAPR1.
- ...
- num_testch(i) (i = 28–31) corresponds to TCCAPR7.

If all the num_testch values in a TCCAPR are ‘0’, the TCCAPR word is not implemented and treated as reserved space. Any attempt to access this space generates a transfer error.

An additional parameter testch_short_intch vector also has positional inference for registers TCCAPRx, such that:

- testch_short_intch(0) corresponds to TCCAPR0[7:0].
- testch_short_intch(1) corresponds to TCCAPR0[15:8].
- testch_short_intch(2) corresponds to TCCAPR0[23:16].
- testch_short_intch(3) corresponds to TCCAPR0[31:24].
- testch_short_intch(4) corresponds to TCCAPR1[7:0].
- ...
- testch_short_intch(31) corresponds to TCCAPR7[31:24].

If a testch_short_intch value is zero, the corresponding byte in the relevant TCCAPR is not implemented and is treated as reserved space.

The parameter num_testch has higher precedence than testch_map_enable. For example, if num_testch[3:0] = ‘0011’ and testch_map_enable[3:0] = ‘1101’, only the TCCAPR0[7:0] byte is implemented.

34.5.1.29 Test Channel Data Registers 96–127 (TCDR96–TCDR127)

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.

Offset 0x450–0x4CC 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[15:0]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 309. Test Channel Data Registers (TCDR96–TCDR127)

Table 390. TCDR96–TCDR127 field descriptions

Field	Description
0–3	Reserved. Write of any value has no effect; read value is always zero.
4 PCE	Precharge Enable This bit enables the precharging phase during channel conversion. 0 Precharge phase is disabled 1 Precharge phase is enabled
5	Reserved. Write of any value has no effect; read value is always zero.
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
8–10	Reserved Write of any value has no effect; read value is always 0.

Table 390. TCDR96–TCDR127 field descriptions(Continued)

Field	Description
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.
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[1:0]	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.

The length of the useful converted data out of 16 bits of CDATA field also depends on the generic *resolution* and *conv_mode_select*. Each of TCDRx corresponds to a test channel inferred through generic num_testch and if any of the num_testch(i) is ‘0’, the corresponding TCDRx is not implemented and any access to it generates a transfer error.

These registers are accessed 32-bit R/W.

The REFSEL bit implementation is dependent on parameter vector altref_exists_ch[127:96].

34.6 Start of conversion pulse delay

The following table lists delays between various trigger points (initiation request for conversion as registered at system clock) and actual issue of start pulse (adc_start) to the ADC hardmacrocell. Here each possible trigger source is described with the number of

clock cycles required for different operations until trigger of adc_start. This table donot take into account any software delay involved in programming or polling of any bit in order to measure conversion timing.

Table 391. Start of conversion pulse delay table

Trigger source	Register control (ipg_clk_s)	Inter domain synchronization (ipg_clk)	ADC state machine (ipg_clk_adc)	Inter domain synchronization (ipg_clk_adc)	Comments
Normal conversion triggered by software (setting of NSTART of MCR)	1	—	2	1-2(2)	Cumulative delay from wen(byte_en) assertion to actual issue of adc_start pulse = 3-4 ipg_clk_adc cycles + 1 ipg_clk cycle.
Normal conversion triggered by external trigger (HW trigger through ipp_ind_start_trg port)	—	—	2	1–2(2)	Cumulative delay from the triggering edge (rising or falling) to the actual issue of adc_start pulse = 3–4 ipg_clk_adc cycles.
Normal conversion within a chain	—	—	0 (from EOC of previous conversion) 1 (from last_evaluation_cyc of previous conversion)	—	Delay from the EOC of last conversion, The start of next conversion is issued exactly on the same cycle when EOC is expected for previous conversion, start pulse in issued looking at last_evalution_cyc_i which is received one cycle prior to EOC.
Injected conversion triggered by software (setting of JSTART bit of MCR)	1	—	3	1–2(2)	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 ipg_clk_adc cycles + 1 ipg_clk cycle.
Injected conversion triggered by external Trigger (HW trigger through ipp_ind_injection_trg port)	—	—	3	1–2(2)	Cumulative delay from triggering edge (rising or falling) to actual issue of adc_start = 4–5 ipg_clk_adc cycles.

Table 391. Start of conversion pulse delay table(Continued)

Trigger source	Register control (ipg_clk_s)	Inter domain synchronization (ipg_clk)	ADC state machine (ipg_clk_adc)	Inter domain synchronization (ipg_clk_adc)	Comments
Injected conversion within a chain (delay between consecutive conversion of injected chain)	—	—	0 (from EOC of previous conversion) 1 (from last_evaluation_cyc of previous conversion)	—	Delay from EOC 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 is expected for previous conversion, start pulse in issued looking at last_evalution_cyc_i which is received one cycle prior to EOC
Injected conversion over normal conversion (software injected conversion chain)	1	1–2(2)	3–5(1)	1–2(2)	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 ipg_clk_adc cycles + 2-3 ipg_clk cycles.
Injection of conversion over normal conversion in JTRGSEQ mode	—	—	3	1–2(2)	Cumulative delay from triggering edge (rising or falling) to actual issue of adc_start = 4–5 ipg_clk_adc cycles.
Abort of a Normal/Injected Channel conversion during Normal Chain/Injected Chain execution (setting of MCR[ABORT] bit)	1	1–2(2)	3–5(1)	1–2(2)	Delay from writing of ABORT bit (wen/byte_en assertion) to actual issue of adc_start pulse for the next channel conversion = 4–7 ipg_clk_adc cycles + 2–3 ipg_clk cycles.

1. The above delay values are calculated over a range of system and ADC clock frequencies. With ipg_clk(max) = 50 MHz to ipg_clk(min) = 40 MHz and ipg_clk_adc(max) = 16 MHz to ipg_clk_adc(min) = 14 MHz. L.H.S value corresponds to max frequencies and RHS corresponds to min frequencies.
2. 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: steps 3-7 can also be executed before step 2.
3. Program the required conversion mode and control options for e.g. 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 Cross Triggering Unit (CTU)

35.1 Introduction

The cross triggering unit (CTU) is intended to completely avoid CPU involvement in the time acquisitions of state variables during the control cycle that can be the PWM cycle, the half PWM cycle or a number of PWM cycles. In such cases the pre-setting of the acquisition times needs to be completed during the previous control cycle, where the actual acquisitions are to be made, and a double-buffered structure for the CTU registers is used, in order to activate the new settings at the beginning of the next control cycle. Additionally, four FIFOs inside the CTU are available to store the ADC results.

35.2 CTU overview

The CTU receives various incoming signals from different sources (timers, position decoder and/or external pins). These signals are then processed to generate as many as eight trigger events. An input can be a rising edge, a falling edge or both, edges of each incoming signal. The output can be a pulse or a command (or a stream of consecutive commands for over-sampling support) or both, to one or more peripherals (for example, ADC or timers).

The CTU interfaces to the following peripherals:

- eTimer — 12 inputs (EXT_IN0-EXT_IN11)
- GPIO — 1 input (EXT_IN15)

The 13 input signals are digital signals and the CTU must be able to detect a rising and/or a falling edge for each of them.

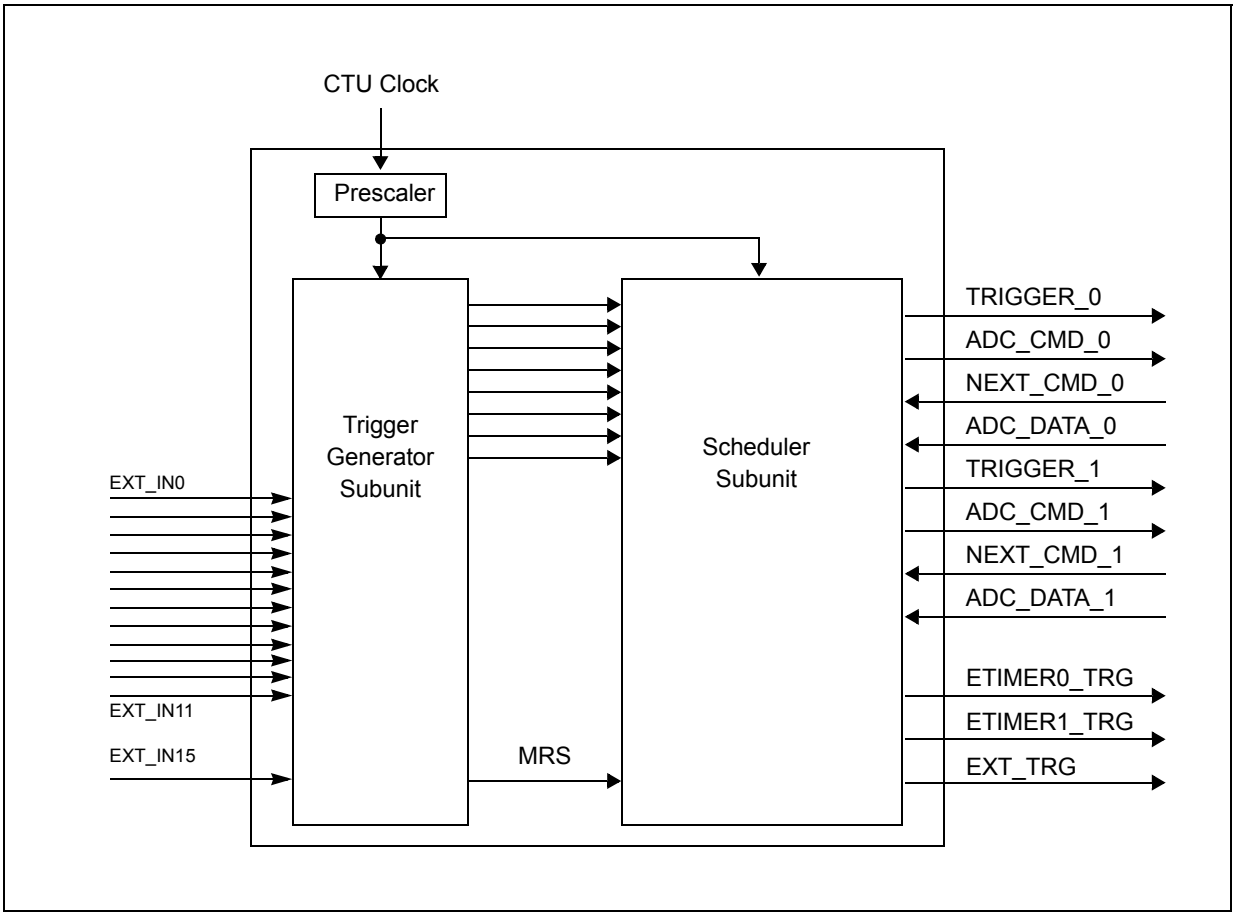
Refer to *Table 7: Peripheral multiplexing description* for description of EXT_IN.

The CTU comprises the following:

- Input signals interface
- User interface (such as configuration registers)
- ADC interface
- Timers interface

The block diagram of the CTU is shown in *Figure 310*.

Figure 310. Cross triggering unit diagram



The CTU consists of two subunits:

- Trigger generator
- Scheduler

The trigger generator subunit handles incoming signals, selecting for each signal, the active edges to generate the Master Reload signal, and generates as many as eight trigger events (signals). The scheduler subunit generates the trigger event output according to the occurred trigger event (signal).

35.3 Functional description

The following describes the functionality of the CTU.

35.3.1 Trigger events features

The TGS is capable of generating as many as eight trigger events. Each trigger event has the following characteristics:

- generation of the trigger event is sequential in time
- the triggers list uses eight 16-bit double-buffered registers
- the new triggers list is loaded on each Master Reload Signal (MRS)

Note: The triggers list is only reloaded on an MRS occurrence if the reload enable bit is set.

35.3.2 Trigger generator subunit (TGS)

The trigger generator subunit has the following two modes:

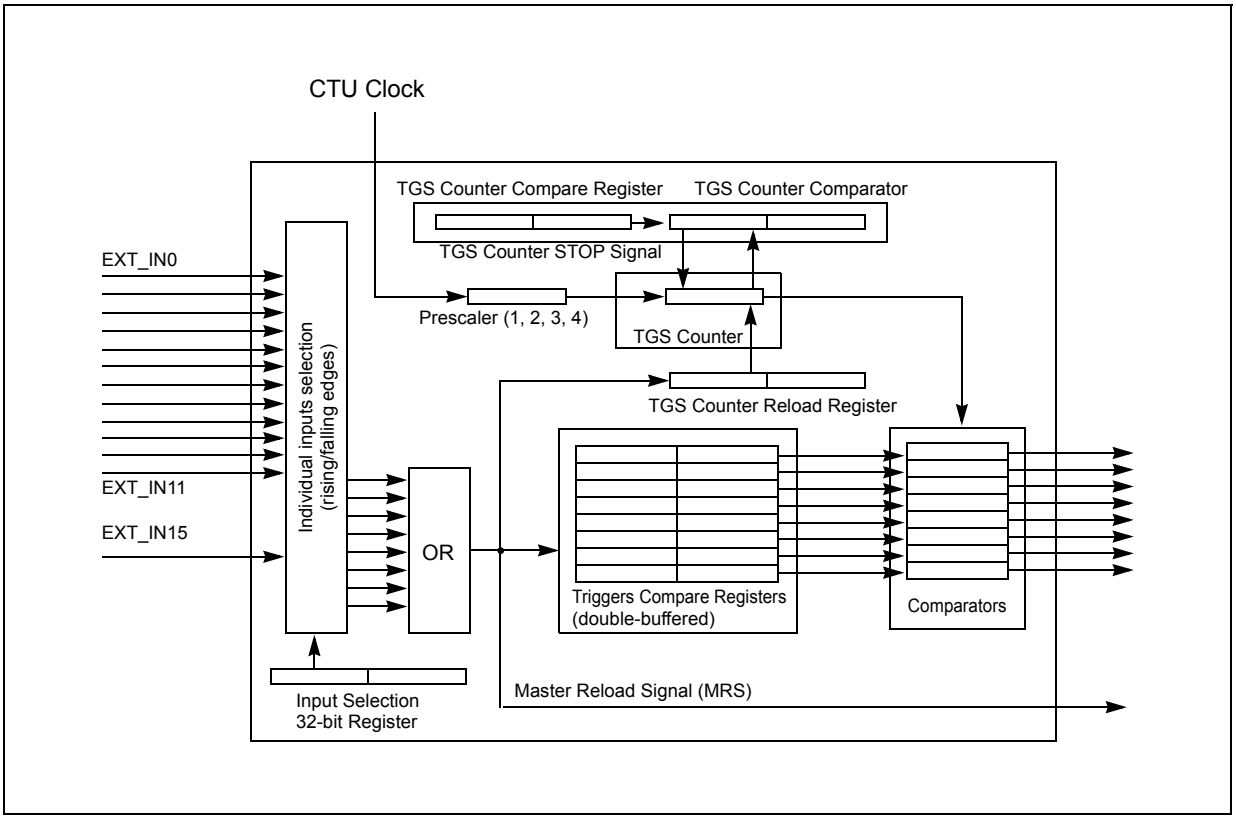
- Triggered mode—Each event source for the incoming signals can generate as many as eight trigger event outputs. For the ADC, a commands list is received from the CPU, and each event source can generate as many as eight commands or command streams.
- Sequential mode—An event source from the incoming signals can generate one trigger event output, the next event source generates the next trigger event output, and so on in a predefined sequence. For the ADC, a commands list is received from the CPU and the sequence of the selected incoming trigger events generate commands or command streams.

The TGS Mode is selected using the TGS_M bit in the TGS Control Register.

35.3.3 TGS in triggered mode

The structure of the TGS in Triggered mode is shown in *Figure 311*.

Figure 311. TGS in triggered mode



The TGS has 13 input signals, each of which is selected from the input selection register (TGSISR) which enables or disables rising or falling edge signals independently. These signals are ORed in order to generate the MRS. At the beginning of control cycle *n* (defined by the MRS occurrence), the MRS preloads the TGS counter register using the preload

value written to the double-buffered register (TGSCRR) during the control cycle *n* – 1, and reloads all the double-buffered registers (such as Trigger Compare registers, TGSCR, TGSCRR itself).

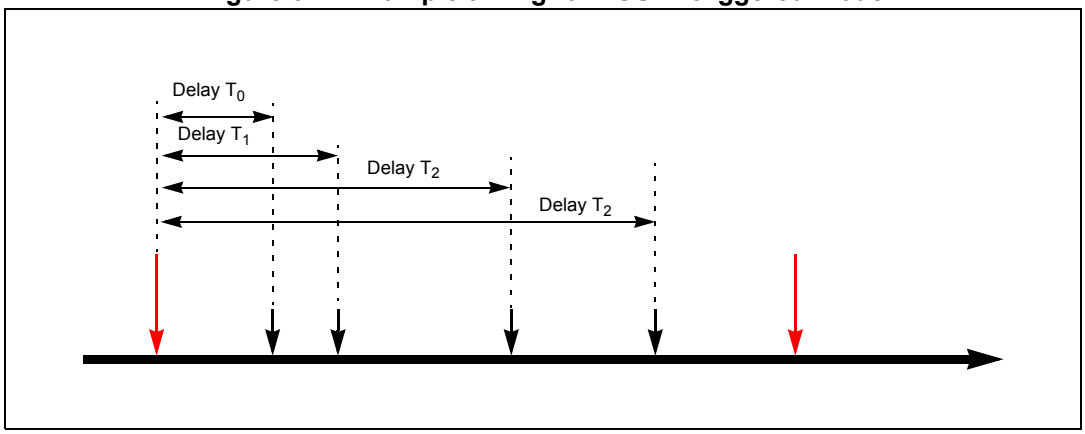
The triggers list registers consist of eight compare registers. Each triggers list register is associated with a comparator. On reload (MRS occurrence), the comparators are disabled. One TGS clock cycle is necessary to enable them and to start the counting. The MRS is output together with individual trigger signals. The MRS can be performed by hardware or by software. The MRS_SG bit in the CTU control register, if set to '1', generates equivalent software MRS (that is, resets/reloads TGS Counter and reloads all double-buffered registers). This bit is cleared by each hardware or software MRS occurrence.

The TGS counter compare register and the TGS counter comparator are used to stop the TGS counter when it reaches the value stored in the TGS counter compare register before an MRS occurs.

The prescaler for TGS and SU can be 1,2,3,4 (PRES bits in the TGS Control Register).

An example timing for the TGS in Triggered Mode is shown in *Figure 312*. The red arrows indicate the MRS occurrences, while the black arrows indicate the trigger event occurrences, with the relevant delay with respect to the last MRS occurrence.

Figure 312. Example timing for TGS in triggered mode



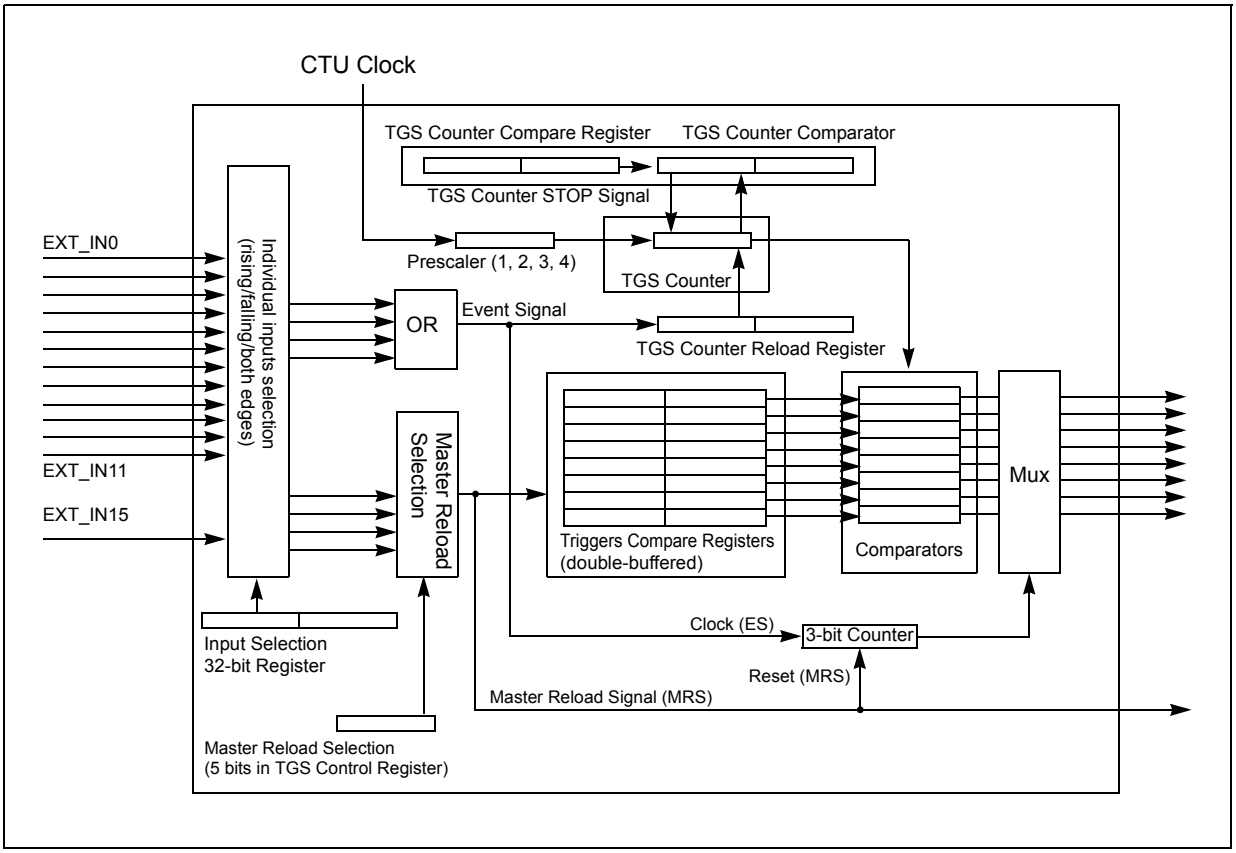
35.3.4 TGS in sequential mode

The structure of the TGS in sequential mode is shown in *Figure 313*.

The 26 input events (13 signals with two edges for signal), which can be individually enabled, are ORed in order to generate the event signal (ES). The ES enables the reload of the TGS counter register and pilots the 3-bit counter in order to select the next active trigger. One of the 26 input events can be selected, through the MRS_SM (master reload selection sequential mode) bits in the TGS control register, to be the MRS, that enables the reload of the triggers list and resets the 3-bit counter (incoming events counter), that is, the MRS is the signal linked with the control cycle defined as the time window between two consecutive MRSs. In this mode, each incoming event sequentially enables only one trigger event through the 3-bit counter and the MUX. The MUX is a selection switch that enables, according to the number of event signals occurred, only one of the eight trigger signals to the scheduler subunit. Sequences of as many as eight trigger events can be supported within this control cycle.

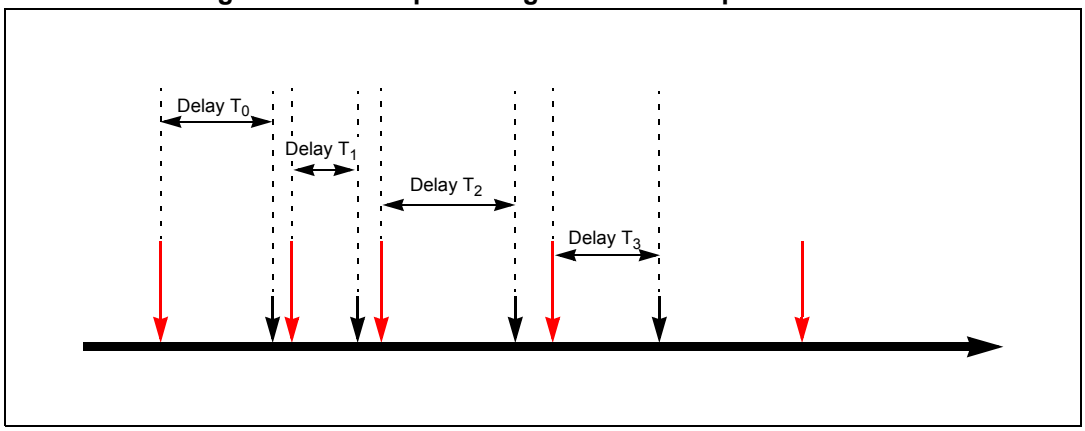
For the other features see the previous paragraphs.

Figure 313. TGS in sequential mode



An example timing diagram for TGS in sequential mode is shown in *Figure 314*. The red arrows indicate the MRS occurrences and ES occurrences, while the black arrows indicate the trigger event occurrences with the relevant delay with respect to the ES occurrence. The first red arrow indicates the first ES occurrence, which is also the MRS.

Figure 314. Example timing for TGS in sequential mode

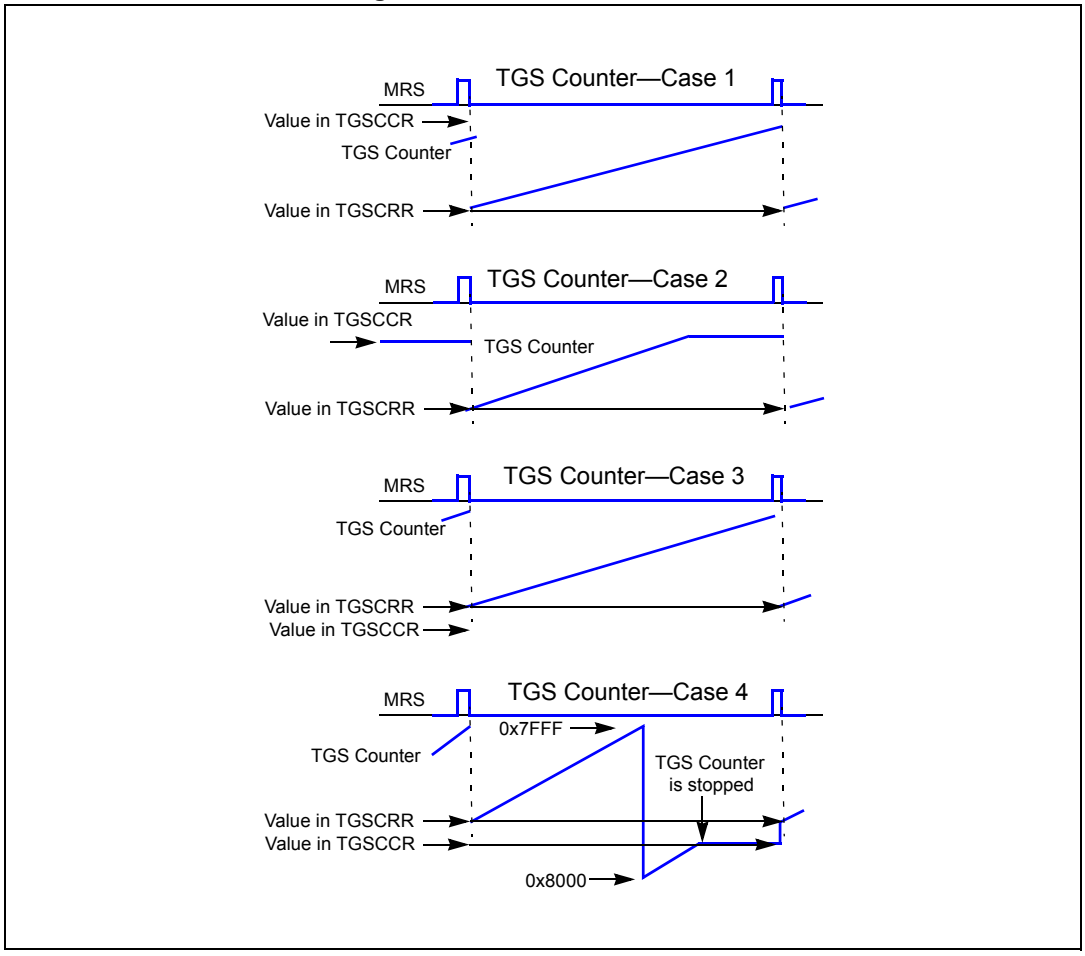


35.3.5 TGS counter

The TGS counter is able to count from negative to positive, that is, from 0x8000 to 0x7FFF. *Figure 315* shows examples in order to explain the TGS counter counts. 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 as the value of the TGSCCR.

Figure 315. TGS counter cases



35.4 Scheduler subunit (SU)

The structure of the SU is shown in *Figure 316*.

The SU generates the trigger event output according to the occurred trigger event, and it has the same functionality in both TGS modes (triggered mode and sequential mode). Each of the four SU outputs:

- ADC command or ADC stream of commands
- eTimer1 pulse (ETIMER0_TRG internally connected to eTimer_0/1/2/3 ETC_0)(o)
- eTimer2 pulse (ETIMER1_TRG internally connected to eTimer_0/1/2/3 ETC_1)(o)
- External trigger pulse (EXT_TRG)

o. Refer to *Table 7: Peripheral multiplexing description*.

can be linked to any of eight trigger events by the Trigger Handler block. Each trigger event can be linked to one or more SU outputs.

If two events at the same time are linked to the same output only one output is generated and an error is provided. The output is generated using the trigger with the lowest index. For example, if trigger 0 and trigger 1 are linked to the ADC output and they occur together, an error is generated and the output linked with the trigger 0 is generated.

When a trigger is linked to the ADC, an associated ADC command (or stream of commands) is generated. The ADC Commands List Control Register (CLCR*x*) sets the assignment to an ADC command or to a stream of commands. When a trigger is linked to a timer or to the external trigger, a pulse with an active rising edge is generated. Additional features for the external triggers are available:

The external trigger output has:

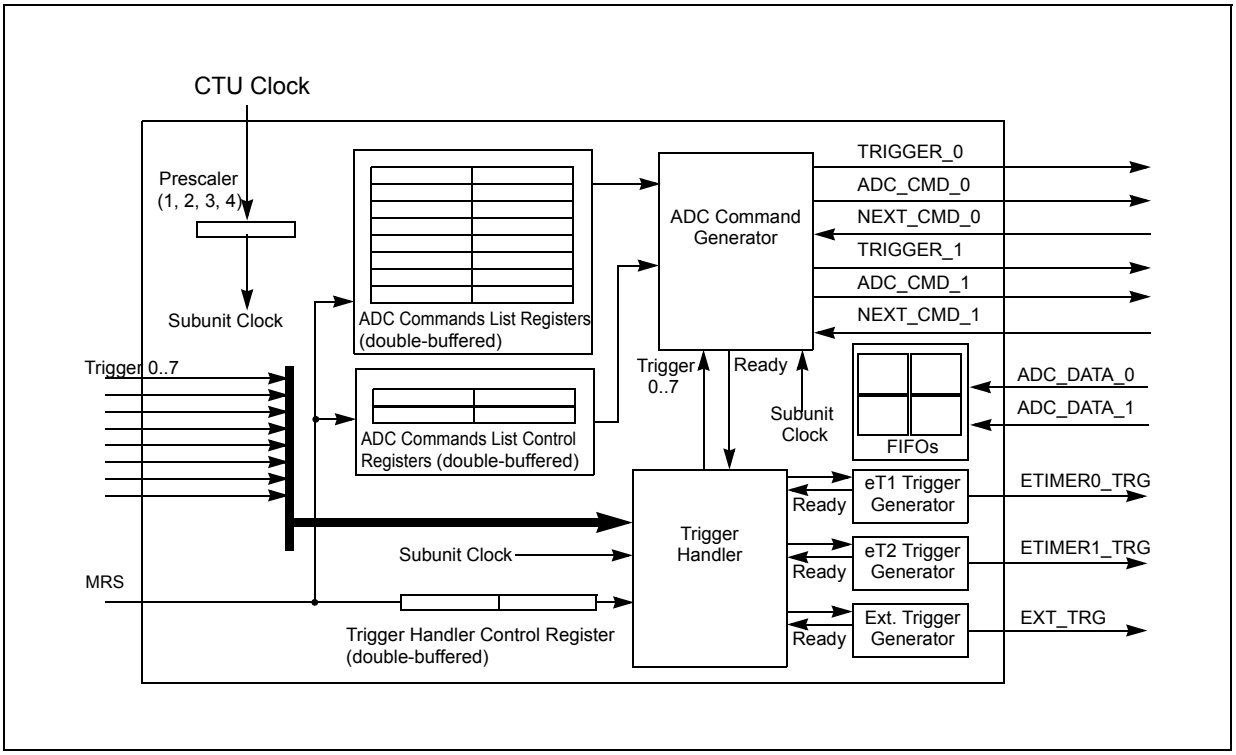
- Pulse mode
- Toggle mode

In Toggle Mode, each trigger event is linked to the external trigger, the external trigger pin toggles. The ON-Time for both modes (Pulse mode and Toggle mode) of the triggers is defined from a COTR register (Control 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 has past. The ON-Time and the Guard Time are only used for external Triggers.

External signals can be asynchronous with motor control clock. For this reason a programmable digital filter is available. The external signal is considered at 1 if it is latched N time at 1, and is considered at 0 if it is latched N time at 0, where N is a value in the digital filter control register.

Trigger events in the SU can be initiated by hardware or by software, and an additional software control is possible for each trigger event (as for the MRS), so 1 bit for each trigger event in the CTU Control Register is used to generate an equivalent software trigger event. Each of these bits is cleared by a respective hardware or software trigger event.

Figure 316. Scheduler subunit



35.4.1 ADC commands list

The ADC can be controlled by the CPU (CPU Control Mode) and by the CTU (CTU Control Mode). The CTU can control the ADC by sending an ADC command only when the ADC is in CTU control mode. During the CTU control mode, the CPU is able to write to the ADC registers but it can not start a new conversion. A control bit is allowed to select from the classic interface of the CTU control mode. Once selected, no change is possible unless a reset occurs.

The SU uses a Commands List in order to select the command to send to the ADC when a trigger event occurs. The commands list can hold twenty-four 16-bit commands (see *Section 35.4.2*) and it is double-buffered, that is, the commands list can be updated at any time between two consecutive MRS, but the changes become workable only after the next MRS occurs, and a correct reload is performed. In order to manage the commands list, 5 bits are available in the CLCRx (ADC Commands List Control Register *x*), for the position of the first command in the list of commands for each trigger event. The number of commands piloted by the same trigger event is defined directly in the commands list. For each command, a bit defines whether or not it is the first command of a commands list.

35.4.2 ADC commands list format

The CTU can be interfaced with two ADCs, supporting the Single Conversion Mode and the Dual Conversion Mode.

In Single Conversion Mode only one ADC starts a conversion at a time. In Dual Conversion Mode both ADCs start a conversion at the same time; in particular both the ADC conversions are performed at the same time while the storage of the results is performed in series. In Dual Conversion Mode, 4 bits select each channel number, and the conversion

mode selection bit selects the Dual Conversion Mode. If the Single Conversion Mode is selected, 5 of the 8 bits reserved to select the channels in Dual Conversion Mode are re-used to select the channel (4 bits) and the ADC unit (1 bit). See *Section 35.8.10: Commands list registers (CLRn)*.

The result of each conversion is stored in one of the four available FIFOs.

The interrupt request bit is used as an interrupt request when ADC will complete the command with this bit set and it is only for CTU internal use. Before the next command to the CTU controls is sent, the value of the first command (FC) bit is checked to see if it is the current command is the first command of a new stream of consecutive commands or not. If not, the CTU sends the command.

According to the previous considerations, the commands in the list allow control on:

- Channel 0: number of ADC channel to sample from ADC unit 0 (4 bits)
- Channel 1: number of ADC channel to sample from ADC unit 1 (4 bits)
- FIFO selection bits for the ADC unit 0/1 (2 bits)
- Conversion Mode selection bit
- First command bit (only for CTU internal use)
- Interrupt request bit (only for CTU internal use)

35.4.3 ADC results

ADC results can be stored in the channel relevant standard result register and/or in one of the four FIFOs: the different FIFOs allow to dispatch ADC results according to their type of acquisition (for example phase currents, rotor position or ground-noise). Each FIFO has its own interrupt line and DMA request signal (plus an individual overflow error bit in the FIFO status register). The store 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.

Each of the FIFOs has 16 entries.

- Results in each FIFO can be read by a 16-bit read transaction (only the result is read in order to minimize the CPU load before computing on results) or by a 32-bit read transaction (both the result and the channel number are read in order to avoid blind acquisitions), 5 bits in the upper 16 bits indicate the ADC unit (1 bit) and the channel number (4 bits). The result registers (only for the FIFOs) can be read from two different addresses in the ADC memory map. The format of the result depends on the address from which it is read. The available formats are:
- Unsigned right-justified

  Conversion result is unsigned right-justified data. Bits [11:0] are used for 12-bit resolution and bits [15:12] always return zero when read.
- Signed left-justified

  Conversion result is signed left-justified data. Bit [15] is reserved for sign and is always read as zero for this ADC. Bits[14:3] are used for 12-bit resolution and bits[2:0] always return zero when read.

35.5 Reload mechanism

Some CTU registers are double-buffered, and the reload is controlled by a reload enable bit, as the TGSISR_RE bit or the DFE bit, but for the most of the double-buffered registers, the

reload is controlled by the MRS occurrence, and it is synchronized with the beginning of the CTU control period.

If the MRS occurs while the user is updating some double-buffered registers, for example some registers of the triggers list, the new triggers list will be a mix of the old triggers list and the new triggers list, because the user has not ended the update of the triggers list before the MRS occurrence.

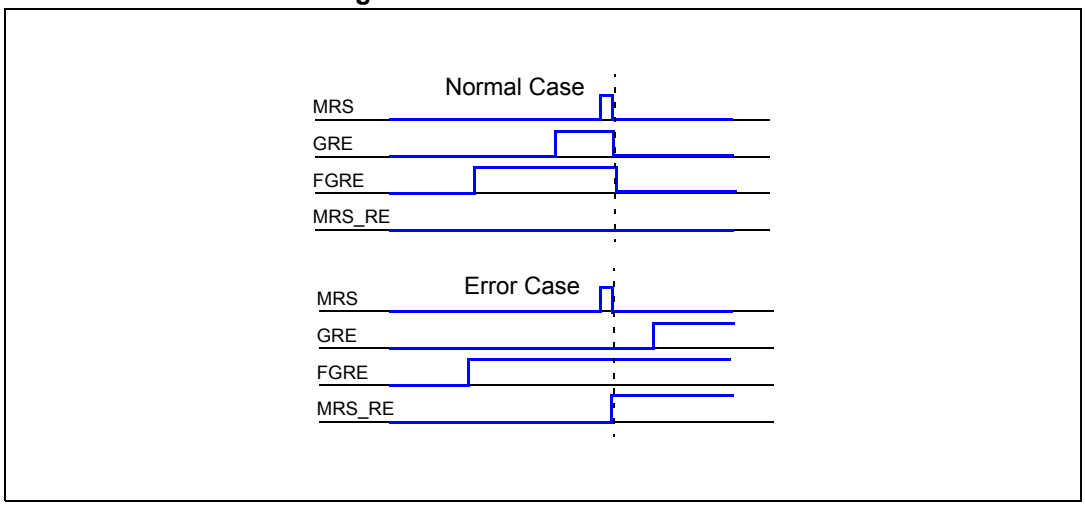
In order to avoid this case, 1 bit enables the reload operation, that is, to inform the CTU that the user has ended updates to the double-buffered registers, and the reload can be performed without problems of mixed scenarios. In order to guarantee the coherency, the reload of all double-buffered registers is enabled by setting GRE (General Reload Enable) bit in the CTU Control Register. The user must ensure that all intended double-buffered registers are updated before a new MRS occurrence. If an MRS occurs before a GRE bit is set (for example, wrong application timing), the update is not performed, the previous values of all double-buffered registers remain active, the error flag is set (the MRS_RE bit in the CTU Error Flag Register) and, if enabled, CTU performs an interrupt request.

All the double-buffered registers use the General Reload Enable (GRE) bit to enable the reload when the MRS occurs. The GRE bit is R/S (Read/Set) and if this bit is 1, the reload can be performed, while if this bit is 0, the reload is not performed. A correct reload resets the GRE bit. None of the double-buffered registers can be written while the GRE bit remains set. The GRE bit can be reset by the occurrence of the next MRS (that is, a correct reload) or by software setting the CGRE bit.

The CGRE is reset by hardware after that GRE bit is reset. If the user sets the CGRE bit and at the same time a MRS occurs, CGRE has the priority so GRE is reset and the reload is not performed. In the same way, the GRE has the priority when compared with the MRS occurrence, and the CGRE has the priority compared with the GRE (the two bits are in the same register so they can be set in the same time). MRS has the priority compared with the re-synchronization bit of the TGSISR.

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, this flag is set to ‘1’ and it is reset by a correct reload. When the MRS occurs while FGRE is 1 and GRE is 1, a correct reload is performed (because all intended registers have been updated before the MRS occurs). If FGRE is 1 and GRE is 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 ended before the MRS occurrence). If FGRE is 0 it is not necessary to perform a reload because all the double-buffered registers are unchanged (see *Figure 317*).

Figure 317. Reload error scenario



35.6 Power safety mode

To reduce power consumption a STOP mode mechanism is implemented.

35.6.0.1 STOP mode

To reduce consumption, it is also possible to enable a stop request from the Mode Entry module. The FIFOs are considered a lot like memory mapped registers, otherwise there could be some problems if a read operation occurs. When the clock is started after a stop signal, some mistakes could occur. For example, a wrong trigger could be provided because it was programmed before the stop signal was performed, and some incorrect write operations into the FIFOs could happen. In order to avoid the problems linked to a wrong trigger, the CTU output can be disabled by the CTU_ODIS bit and the ADC interface state machine can be reset by the CRU_ADC_R (see *Section 35.8.21: Cross triggering unit control register (CTUCR)*).

35.7 Interrupts and DMA requests

35.7.1 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).

Note: If enabled, the DMA request on the MRS occurrence is performed only if a reload is performed, that is, only if the GRE bit is set.

Moreover, this CTU implementation requires DMA support for reading the data from the FIFOs. One DMA channel is available for each FIFO. Each FIFO can perform a DMA request when the number of words stored in the FIFO reaches the threshold value.

35.7.2 CTU faults and errors

Faults and errors that could occur during the programming include:

- An MRS occurs while user is updating the double-buffered registers and the MRS_RE bit is set.
- Receiving more than eight EVs before that the next MRS occurs in TGS sequential mode and the SM_TO bit is set.
- 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 makes a mistake and a trigger event occurs when another one is processed, the incoming trigger event will be lost and an error occurs).

  There are four overrun flags (one for each type of output). The general mechanism shall be as in *Figure 316*.

  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, eT1 Trigger Generator, eT2 Trigger 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 the case of ADC command generator, the Ready signal shall be kept low until the last conversion in the batch is finished. The respective overrun flag is set at the following conditions:

  – Ready signal is low.
  – The rising edge of the respective trigger signal (from Trigger Handler to generator sub-block) occurs.

  This architecture allows the user to preset a trigger to the eTimer0 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. One of the following bits is set: ADC_OE, T0_OE, T1_OE, or ET_OE.
- Invalid (unrecognized) ADC command and the ICE bit is set.
- The MRS occurs before the enabled trigger events occur and the MRS_O bit is set.
- TGS overrun in sequential mode: a new incoming EV occurs before than the trigger event selected by the previous EV occurs. The incoming EV sets an internal busy flag. The outgoing trigger event (all line are ORed) resets this flag to 0. TGS Overrun in the sequential mode shall be generated under the following conditions:

  – TGS is in sequential mode
  – there is an incoming EV while the busy flag is high. the TGS_OSM bit is set.

The faults/errors flags in the CTU error flag register and in the CTU interrupt flag register can be cleared by writing a 1 while writing a 0 has no effect. The CTU does not support a write-protection mechanism.

35.7.3 CTU interrupt/DMA requests

The CTU can perform the following interrupt/DMA requests (15 interrupt lines):

- Error interrupt request (see *Section 35.7.2*) (1 interrupt line)
- ADC command interrupt request (1 interrupt line)
- Interrupt request on MRS occurrence (1 interrupt line)
- Interrupt request on each trigger event occurrence (1 interrupt line for each trigger event)
- FIFOs interrupt requests and/or DMA transfer request (1 interrupt line for each FIFO)
- DMA transfer request on the MRS occurrence if GRE bit is set

The interrupt flags are shown in *Table 392*.

Table 392. CTU interrupts

Category	Interrupt	Interrupt function
Managed individually	MRS_I	MRS Interrupt flag (IRQ800)
	T0_I	Trigger 0 interrupt flag (IRQ803)
	T1_I	Trigger 1 interrupt flag (IRQ804)
	T2_I	Trigger 2 interrupt flag (IRQ805)
	T3_I	Trigger 3 interrupt flag (IRQ806)
	T4_I	Trigger 4 interrupt flag (IRQ807)
	T5_I	Trigger 5 interrupt flag (IRQ808)
	T6_I	Trigger 6 interrupt flag (IRQ809)
	T7_I	Trigger 7 interrupt flag (IRQ810)
	ADC_I	ADC command interrupt flag (IRQ801)
ORed onto FIFO0_I (IRQ811)	FIFO_FULL0	This bit is set to 1 if the FIFO 0 is full.
	FIFO_EMPTY0	This bit is set to 1 if the FIFO 0 is empty.
	FIFO_OVERFLOW0	This bit is set to 1 if the number of words exceeds the value set in the threshold 0.
	FIFO_OVERRUN0	This bit is set to 1 if a write operation occurs when corresponding FIFO_FULL0 flag is set.
ORed onto FIFO1_I (IRQ812)	FIFO_FULL1	This bit is set to 1 if the FIFO 1 is full.
	FIFO_EMPTY1	This bit is set to 1 if the FIFO 1 is empty.
	FIFO_OVERFLOW1	This bit is set to 1 if the number of words exceeds the value set in the threshold 1.
	FIFO_OVERRUN1	This bit is set to 1 if a write operation occurs when corresponding FIFO_FULL1 flag is set.

Table 392. CTU interrupts(Continued)

Category	Interrupt	Interrupt function
ORed onto FIFO2_I (IRQ813)	FIFO_FULL2	This bit is set to 1 if the FIFO 2 is full.
	FIFO_EMPTY2	This bit is set to 1 if the FIFO 2 is empty.
	FIFO_OVERFLOW2	This bit is set to 1 if the number of words exceeds the value set in the threshold 2.
	FIFO_OVERRUN2	This bit is set to 1 if a write operation occurs when corresponding FIFO_FULL2 flag is set.
ORed onto FIFO3_I (IRQ814)	FIFO_FULL3	This bit is set to 1 if the FIFO 3 is full.
	FIFO_EMPTY3	This bit is set to 1 if the FIFO 3 is empty.
	FIFO_OVERFLOW3	This bit is set to 1 if the number of words exceeds the value set in the threshold 3.
	FIFO_OVERRUN3	This bit is set to 1 if a write operation occurs when corresponding FIFO_FULL3 flag is set.
ORed onto ERR_I (IRQ802)	MRS_RE	Master Reload Signal Reload Error
	SM_TO	Trigger Overrun (more than 8 EV) in TGS Sequential Mode
	ICE	Invalid Command Error
	MRS_O	Master Reload Signal Overrun
	TGS_OSM	TGS Overrun in Sequential Mode
	ADC_OE	ADC command generation Overrun Error
	T0_OE	Timer 0 trigger generation Overrun Error
	T1_OE	Timer 1 trigger generation Overrun Error
	ET_OE	External Trigger generation Overrun Error

35.8 Memory map

Table 393. CTU memory map

Offset from CTU_BASE (0xFFC2_C000)	Register	Location
0x0000	TGSISR — Trigger Generator Subunit Input Selection Register	*Section 35.8.1*
0x0004	TGSCR — Trigger Generator Subunit Control Register	*Section 35.8.2*
0x0006	T0CR — Trigger 0 Compare Register	*Section 35.8.3*
0x0008	T1CR — Trigger 1 Compare Register	*Section 35.8.3*
0x000A	T2CR — Trigger 2 Compare Register	*Section 35.8.3*
0x000C	T3CR — Trigger 3 Compare Register	*Section 35.8.3*
0x000E	T4CR — Trigger 4 Compare Register	*Section 35.8.3*

Table 393. CTU memory map(Continued)

Offset from CTU_BASE (0xFFC2_C000)	Register	Location
0x0010	T5CR — Trigger 5 Compare Register	Section 35.8.3
0x0012	T6CR — Trigger 6 Compare Register	Section 35.8.3
0x0014	T7CR — Trigger 7 Compare Register	Section 35.8.3
0x0016	TGSCCR — TGS Counter Compare Register	Section 35.8.4
0x0018	TGSCRR — TGS Counter Reload Register	Section 35.8.5
0x001A-0x001B	Reserved	
0x001C	CLCR1 — Commands List Control Register 1	Section 35.8.6
0x0020	CLCR2 — Commands List Control Register 2	Section 35.8.7
0x0024	THCR1 — Trigger Handler Control Register 1	Section 35.8.8
0x0028	THCR2 — Trigger Handler Control Register 2	Section 35.8.9
0x002C	CLR1 — Commands List Register 1	Section 35.8.10
0x002E	CLR2 — Commands List Register 2	Section 35.8.10
0x0030	CLR3 — Commands List Register 3	Section 35.8.10
0x0032	CLR4 — Commands List Register 4	Section 35.8.10
0x0034	CLR5 — Commands List Register 5	Section 35.8.10
0x0036	CLR6 — Commands List Register 6	Section 35.8.10
0x0038	CLR7 — Commands List Register 7	Section 35.8.10
0x003A	CLR8 — Commands List Register 8	Section 35.8.10
0x003C	CLR9 — Commands List Register 9	Section 35.8.10
0x003E	CLR10 — Commands List Register 10	Section 35.8.10
0x0040	CLR11 — Commands List Register 11	Section 35.8.10
0x0042	CLR12 — Commands List Register 12	Section 35.8.10
0x0044	CLR13 — Commands List Register 13	Section 35.8.10
0x0046	CLR14 — Commands List Register 14	Section 35.8.10
0x0048	CLR15 — Commands List Register 15	Section 35.8.10
0x004A	CLR16 — Commands List Register 16	Section 35.8.10
0x004C	CLR17 — Commands List Register 17	Section 35.8.10
0x004E	CLR18 — Commands List Register 18	Section 35.8.10
0x0050	CLR19 — Commands List Register 19	Section 35.8.10
0x0052	CLR20 — Commands List Register 20	Section 35.8.10
0x0054	CLR21 — Commands List Register 21	Section 35.8.10
0x0056	CLR22 — Commands List Register 22	Section 35.8.10
0x0058	CLR23 — Commands List Register 23	Section 35.8.10

Table 393. CTU memory map(Continued)

Offset from CTU_BASE (0xFFC2_C000)	Register	Location
0x005A	CLR24 — Commands List Register 24	*Section 35.8.10*
0x005C—0x006B	Reserved	
0x006C	FDCR — FIFO DMA Control Register	*Section 35.8.11*
0x0070	FCR — FIFO Control Register	*Section 35.8.12*
0x0074	FTH — FIFO Threshold Register	*Section 35.8.13*
0x0078–0x007B	Reserved	
0x007C	FST — FIFO Status Register	*Section 35.8.14*
0x0080	FR0 — FIFO Right aligned data register 0	*Section 35.8.15*
0x0084	FR1 — FIFO Right aligned data register 1	*Section 35.8.15*
0x0088	FR2 — FIFO Right aligned data register 2	*Section 35.8.15*
0x008C	FR3 — FIFO Right aligned data register 3	*Section 35.8.15*
0x0080–0x009F	Reserved	
0x00A0	FL0 — FIFO Left aligned data register 0	*Section 35.8.16*
0x00A4	FL1 — FIFO Left aligned data register 1	*Section 35.8.16*
0x00A8	FL2 — FIFO Left aligned data register 2	*Section 35.8.16*
0x00AC	FL3 — FIFO Left aligned data register 3	*Section 35.8.16*
0x00B0–0x00BF	Reserved	
0x00C0	CTUEFR — Cross Triggering Unit Error Flag Register	*Section 35.8.17*
0x00C2	CTUIFR — Cross Triggering Unit Interrupt Flag Register	*Section 35.8.18*
0x00C4	CTUIR — Cross Triggering Unit Interrupt Register	*Section 35.8.19*
0x00C6	COTR — Control ON-Time Register	*Section 35.8.20*
0x00C8	CTUCR — Cross triggering unit control register	*Section 35.8.21*
0x00CA	CTUDF — Cross Triggering Unit Digital Filter register	*Section 35.8.22*
0x00CC–0x3FFF	Reserved	

Table 394. TGS registers

Offset from CTU_BASE	Register	Double-buffered	Synchronization	Reset value
0x0000	TGSISR — Trigger Generator Subunit Input Selection Register	Yes	TGSISR_RE	0x0000_0000
0x0004	TGSCR — Trigger Generator Subunit Control Register	Yes	MRS	0x0000
0x0006	T0CR — Trigger 0 Compare Register	Yes	MRS	0x0000
0x0008	T1CR — Trigger 1 Compare Register	Yes	MRS	0x0000

Table 394. TGS registers(Continued)

Offset from CTU_BASE	Register	Double-buffered	Synchronization	Reset value
0x000A	T2CR — Trigger 2 Compare Register	Yes	MRS	0x0000
0x000C	T3CR — Trigger 3 Compare Register	Yes	MRS	0x0000
0x000E	T4CR — Trigger 4 Compare Register	Yes	MRS	0x0000
0x0010	T5CR — Trigger 5 Compare Register	Yes	MRS	0x0000
0x0012	T6CR — Trigger 6 Compare Register	Yes	MRS	0x0000
0x0014	T7CR — Trigger 7 Compare Register	Yes	MRS	0x0000
0x0016	TGSCCR — TGS Counter Compare Register	Yes	MRS	0x0000
0x0018	TGSCRR — TGS Counter Reload Register	Yes	MRS	0x0000

Table 395. SU registers

Offset from CTU_BASE	Register	Double-buffered	Synchronization	Reset value
0x001C	CLCR1 — Commands List Control Register 1	Yes	MRS	0x0000_0000
0x0020	CLCR2 — Commands List Control Register 2	Yes	MRS	0x0000_0000
0x0024	THCR1 — Trigger Handler Control Register 1	Yes	MRS	0x0000_0000
0x0028	THCR2 — Trigger Handler Control Register 2	Yes	MRS	0x0000_0000
0x002C – 0x005A	CLR*n* — Commands List Register *n* (n = 1 to 24)	Yes	MRS	0x0000

Table 396. CTU registers

Offset from CTU_BASE	Register	Double-buffered	Synchronization	Reset value
0x00C0	CTUEFR — Cross Triggering Unit Error Flag Register	No	—	0x0000
0x00C2	CTUIFR — Cross Triggering Unit Interrupt Flag Register	No	—	0x0000
0x00C4	CTUIR — Cross Triggering Unit Interrupt Register	No	—	0x0000
0x00C6	COTR — Control ON-Time Register	Yes	MRS	0x0000
0x00C8	CTUCR — Cross triggering unit control register	No	—	0x0000
0x00CA	CTUDF — Cross Triggering Unit Digital Filter	Yes	DFE	0x0000

Table 397. FIFO registers

Offset from CTU_BASE	Register	Double-buffered	Synchronization	Reset value
0x006C	FDCR — FIFO DMA Control Register	No	—	0x0000
0x0070	FCR — FIFO Control Register	No	—	0x0000_0000

Table 397. FIFO registers(Continued)

Offset from CTU_BASE	Register	Double-buffered	Synchronization	Reset value
0x0074	FTH — FIFO Threshold Register	No	—	0x0000_0000
0x007C	FST — FIFO Status Register	No	—	0x0000_0000
0x0080	FR0 — FIFO Right aligned data 0	No	—	0xXXXX_XXXX
0x0084	FR1 — FIFO Right aligned data 1	No	—	0xXXXX_XXXX
0x0088	FR2 — FIFO Right aligned data 2	No	—	0xXXXX_XXXX
0x008C	FR3 — FIFO Right aligned data 3	No	—	0xXXXX_XXXX
0x00A0	FL0 — FIFO Left aligned data 0	No	—	0xXXXX_XXXX
0x00A4	FL1 — FIFO Left aligned data 1	No	—	0xXXXX_XXXX
0x00A8	FL2 — FIFO Left aligned data 2	No	—	0xXXXX_XXXX
0x00AC	FL3 — FIFO Left aligned data 3	No	—	0xXXXX_XXXX

35.8.1 Trigger Generator Sub-unit Input Selection Register (TGSISR)

Address: 0x0000 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	0(1)	0(1)	0(1)	0(1)	0(1)	0(1)	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

1. This bit is not implemented and its value has no impact.

Figure 318. Trigger Generator Sub-unit Input Selection Register (TGSISR)

Table 398. TGSISR field descriptions

Field	Description
0 I15_FE	Input 15, Falling edge Enable: EXT_IN15 0 Disabled 1 Enabled
1 I15_RE	Input 15, Rising edge Enable: EXT_IN15 0 Disabled 1 Enabled
8 I11_FE	Input 11, Falling edge Enable: EXT_IN11 0 Disabled 1 Enabled

Table 398. TGSISR field descriptions(Continued)

Field	Description
9 I11_RE	Input 11, Rising edge Enable: EXT_IN11 0 Disabled 1 Enabled
10 I10_FE	Input 10, Falling edge Enable: EXT_IN10 0 Disabled 1 Enabled
11 I10_RE	Input 10, Rising edge Enable: EXT_IN10 0 Disabled 1 Enabled
12 I9_FE	Input 9, Falling edge Enable: EXT_IN9 0 Disabled 1 Enabled
13 I9_RE	Input 9, Rising edge Enable: EXT_IN9 0 Disabled 1 Enabled
14 I8_FE	Input 8, Falling edge Enable: EXT_IN8 0 Disabled 1 Enabled
15 I8_RE	Input 8, Rising edge Enable: EXT_IN8 0 Disabled 1 Enabled
16 I7_FE	Input 7, Falling edge Enable: EXT_IN7 0 Disabled 1 Enabled
17 I7_RE	Input 7, Rising edge Enable: EXT_IN7 0 Disabled 1 Enabled
18 I6_FE	Input 6, Falling edge Enable: EXT_IN6 0 Disabled 1 Enabled
19 I6_RE	Input 6, Rising edge Enable: EXT_IN6 0 Disabled 1 Enabled
20 I5_FE	Input 5, Falling edge Enable: EXT_IN5 0 Disabled 1 Enabled
21 I5_RE	Input 5, Rising edge Enable: EXT_IN5 0 Disabled 1 Enabled
22 I4_FE	Input 4, Falling edge Enable: EXT_IN4 0 Disabled 1 Enabled
23 I4_RE	Input 4, Rising edge Enable: EXT_IN4 0 Disabled 1 Enabled

Table 398. TGSISR field descriptions(Continued)

Field	Description
24 I3_FE	Input 3, Falling edge Enable: EXT_IN3 0 Disabled 1 Enabled
25 I3_RE	Input 3, Rising edge Enable: EXT_IN3 0 Disabled 1 Enabled
26 I2_FE	Input 2, Falling edge Enable: EXT_IN2 0 Disabled 1 Enabled
27 I2_RE	Input 2, Rising edge Enable: EXT_IN2 0 Disabled 1 Enabled
28 I1_FE	Input 1, Falling edge Enable: EXT_IN1 0 Disabled 1 Enabled
29 I1_RE	Input 1, Rising edge Enable: EXT_IN1 0 Disabled 1 Enabled
30 I0_FE	Input 0, Falling edge Enable: EXT_IN0 0 Disabled 1 Enabled
31 I0_RE	Input 0, Rising edge Enable: EXT_IN0 0 Disabled 1 Enabled

35.8.2 Trigger Generator Sub-unit Control Register (TGSCR)

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	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 319. Trigger Generator Sub-unit Control Register (TGSCR)

Table 399. TGSCR field descriptions

Field	Description
7 ET_TM	This bit enables toggle mode for external triggers.
8:9 PRES	TGS and SU prescaler selection bits 00 1 01 2 10 3 11 4

Table 399. TGSCR field descriptions(Continued)

Field	Description
10:14 MRS_SM	Master Reload Selection in Sequential Mode (5 bits to select one of 32 inputs)
15 TGS_M	Trigger Generator Subunit Mode 0 Triggered Mode 1 Sequential Mode

35.8.3 Trigger *n* Compare Register (T*n*CR)

Address: 0x0006 + *n**0x2 (*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	TnCRV															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 320. Trigger *n* Compare Register (T*n*CR)

Table 400. T*n*CR field descriptions

Field	Description
0:15 TnCRV	Trigger *n* Compare Register Value

35.8.4 TGS Counter Compare Register (TGSCCR)

Address: 0x0016 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	TGSCCV															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 321. TGS Counter Compare Register (TGSCCR)

Table 401. TGSCCR field format

Field	Description
0:15 TGSCCV	TGS Counter Compare Value

35.8.5 TGS Counter Reload Register (TGSCRR)

Address: 0x0018 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	TGSCRV															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 322. TGS Counter Reload Register (TGSCRR)

Table 402. TGSCRR field descriptions

Field	Description
0:15 TGSCRV	TGS Counter Reload Value

35.8.6 Commands list control register 1 (CLCR1)

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	T3_INDEX					0	0	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	0	0	T1_INDEX					0	0	0	T0_INDEX				
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 323. Commands list control register 1 (CLCR1)

Table 403. CLCR1 field descriptions

Field	Description
3:7 T3_INDEX	Trigger 3 Commands List first command address
11:15 T2_INDEX	Trigger 2 Commands List first command address
19:23 T1_INDEX	Trigger 1 Commands List first command address
27:31 T0_INDEX	Trigger 0 Commands List first command address

35.8.7 Commands list control register 2 (CLCR2)

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	T7_INDEX					0	0	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	0	0	T5_INDEX					0	0	0	T4_INDEX				
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 324. Commands list control register 2 (CLCR2)

Table 404. CLCR2 field descriptions

Field	Description
3:7 T7_INDEX	Trigger 7 Commands List first command address
11:15 T6_INDEX	Trigger 6 Commands List first command address
19:23 T5_INDEX	Trigger 5 Commands List first command address
27:31 T4_INDEX	Trigger 4 Commands List first command address

35.8.8 Trigger handler control register 1 (THCR1)

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	T3_E	T3_ETE	T3_T1E	T3_T0E	T3_ADCE	0	0	0	T2_E	T2_ETE	T2_T1E	T2_T0E	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_T1E	T1_T0E	T1_ADCE	0	0	0	T0_E	T0_ETE	T0_T1E	T0_T0E	T0_ADCE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 325. Trigger handler control register 1 (THCR1)

Table 405. 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_T1E	Trigger 3 Timer 1 output enable 0 Disabled 1 Enabled
6 T3_T0E	Trigger 3 Timer 0 output enable 0 Disabled 1 Enabled
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_T1E	Trigger 2 Timer 1 output enable 0 Disabled 1 Enabled
14 T2_T0E	Trigger 2 Timer 0 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_T1E	Trigger 1 Timer 1 output enable 0 Disabled 1 Enabled
22 T1_T0E	Trigger 1 Timer 0 output enable 0 Disabled 1 Enabled
23 T1_ADCE	Trigger 1 ADC command output enable 0 Disabled 1 Enabled

Table 405. THCR1 field descriptions(Continued)

Field	Description
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_T1E	Trigger 0 Timer 1 output enable 0 Disabled 1 Enabled
30 T0_T0E	Trigger 0 Timer 0 output enable 0 Disabled 1 Enabled
31 T0_ADCE	Trigger 0 ADC command output enable 0 Disabled 1 Enabled

35.8.9 Trigger handler control register 2 (THCR2)

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	T7_E	T7_ETE	T7_T1E	T7_T0E	T7_ADCE	0	0	0	T6_E	T6_ETE	T6_T1E	T6_T0E	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_T1E	T5_T0E	T5_ADCE	0	0	0	T4_E	T4_ETE	T4_T1E	T4_T0E	T4_ADCE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 326. Trigger handler control register 2 (THCR2)

Table 406. 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_T1E	Trigger 7 Timer 1 output enable 0 Disabled 1 Enabled

Table 406. THCR2 field descriptions(Continued)

Field	Description
6 T7_T0E	Trigger 7 Timer 0 output enable 0 Disabled 1 Enabled
7 T7_ADCE	Trigger 7 ADC command output enable 0 Disabled 1 Enabled
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_T1E	Trigger 6 Timer 1 output enable 0 Disabled 1 Enabled
14 T6_T0E	Trigger 6 Timer 0 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_T1E	Trigger 5 Timer 1 output enable 0 Disabled 1 Enabled
22 T5_T0E	Trigger 5 Timer 0 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_T1E	Trigger 4 Timer 1 output enable 0 Disabled 1 Enabled

Table 406. THCR2 field descriptions(Continued)

Field	Description
30 T4_T0E	Trigger 4 Timer 0 output enable 0 Disabled 1 Enabled
31 T4_ADCE	Trigger 4 ADC command output enable 0 Disabled 1 Enabled

35.8.10 Commands list registers (CLR*n*)

Figure 327 and *Table 407* show the register configured for ADC command format in single conversion mode (CMS = 0) (CLR*n*).

Address: 0x002C + *(n-1)**0x2 (*n* =1 to 24)
(See *Table 393*)

Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	CIR	LC	CMS	0	FIFO		0	0	0	0	SU	0	CH			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 327. Commands list register *n* (CLR*n*) (CMS = 0)

Table 407. CLR*n* (CMS = 0) field descriptions

Field	Description
0 CIR	Command Interrupt Request bit 0 Disabled 1 Enabled
1 LC	Last command bit 0 Not last 1 Last
2 CMS	Conversion mode selection bit 0 Single conversion mode 1 Dual conversion mode
4:5 FIFO	FIFO for ADC unit 0/1 00 FIFO 0 selected 01 FIFO 1 selected 10 FIFO 2 selected 11 FIFO 3 selected
10 SU	Selection Unit bit 0 ADC unit 0 selected 1 ADC unit 1 selected
12:15 CH	ADC unit channel number

Figure 328 and *Table 408* show the register configured for ADC command format in dual conversion mode (CMS = 1).

Address: 0x002C + (*n-1*)*0x2 (*n* =1 to 24)
(See *Table 393*)
Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	CIR	LC	CMS	0	FIFO		0		CH_B			0	CH_A			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 328. Commands list register *n* (CLR*n*) (CMS = 1)

Table 408. CLR*n* (CMS = 1) field descriptions

Field	Description
0 CIR	Command Interrupt Request bit 0 Disabled 1 Enabled
1 LC	Last command bit 0 Not last 1 Last
2 CMS	Conversion mode selection 0 Single conversion mode 1 Dual conversion mode
4:5 FIFO	FIFO for ADC unit A/B
7:10 CH_B	ADC unit B channel number
12:15 CH_A	ADC unit A channel number

35.8.11 FIFO DMA control register (FDCR)

Address: 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	0	0	0	DE3	DE2	DE1	DE0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 329. FIFO DMA control register (FDCR)

Table 409. FDCR field descriptions

Field	Description
12 DE3	This bit enables DMA for the FIFO 3 0 Disabled 1 Enabled
13 DE2	This bit enables DMA for the FIFO 2 0 Disabled 1 Enabled
14 DE1	This bit enables DMA for the FIFO 1 0 Disabled 1 Enabled
15 DE0	This bit enables DMA for the FIFO 0 0 Disabled 1 Enabled

35.8.12 FIFO control register (FCR)

Address: 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	0	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	OR_EN3	OF_EN3	EMPTY_EN3	FULL_EN3	OR_EN2	OF_EN2	EMPTY_EN2	FULL_EN2	OR_EN1	OF_EN1	EMPTY_EN1	FULL_EN1	OR_EN0	OF_EN0	EMPTY_EN0	FULL_EN0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 330. FIFO control register (FCR)

Table 410. FCR field descriptions

Field	Description
16 OR_EN3	FIFO 3 Overrun interrupt enable 0 Disabled 1 Enabled
17 OF_EN3	FIFO 3 threshold Overflow interrupt enable 0 Disabled 1 Enabled
18 EMPTY_EN3	FIFO 3 Empty interrupt enable 0 Disabled 1 Enabled

Table 410. FCR field descriptions(Continued)

Field	Description
19 FULL_EN3	FIFO 3 Full interrupt enable 0 Disabled 1 Enabled
20 OR_EN2	FIFO 2 Overrun interrupt enable 0 Disabled 1 Enabled
21 OF_EN2	FIFO 2 threshold Overflow interrupt enable 0 Disabled 1 Enabled
22 EMPTY_EN2	FIFO 2 Empty interrupt enable 0 Disabled 1 Enabled
23 FULL_EN2	FIFO 2 Full interrupt enable 0 Disabled 1 Enabled
24 OR_EN1	FIFO 1 Overrun interrupt enable 0 Disabled 1 Enabled
25 OF_EN1	FIFO 1 threshold Overflow interrupt enable 0 Disabled 1 Enabled
26 EMPTY_EN1	FIFO 1 Empty interrupt enable 0 Disabled 1 Enabled
27 FULL_EN1	FIFO 1 Full interrupt enable 0 Disabled 1 Enabled
28 OR_EN0	FIFO 0 Overrun interrupt enable 0 Disabled 1 Enabled
29 OF_EN0	FIFO 0 threshold Overflow interrupt enable 0 Disabled 1 Enabled
30 EMPTY_EN0	FIFO 0 Empty interrupt enable 0 Disabled 1 Enabled
31 FULL_EN0	FIFO 0 Full interrupt enable 0 Disabled 1 Enabled

35.8.13 FIFO threshold register (FTH)

Address: 0x0074 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	TH3								TH2							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TH1								TH0							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 331. FIFO threshold register (FTH)

Table 411. FTH field descriptions

Field	Description
0:7 TH3	FIFO 3 Threshold
8:15 TH2	FIFO 2 Threshold
16:23 TH1	FIFO 1 Threshold
24:31 TH0	FIFO 0 Threshold

35.8.14 FIFO status register (FST)

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	OR3	OF3	EMP3	FULL3	OR2	OF2	EMP2	FULL2	OR1	OF1	EMP1	FULL1	OR0	OF0	EMP0	FULL0
W	w1c				w1c				w1c				w1c			
Reset	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0

Figure 332. FIFO status register (FST)

Table 412. FST field descriptions

Field	Description
16 OR3	FIFO 3 Overrun interrupt flag A read of this bit clears it. 0 Interrupt has not occurred 1 Interrupt has occurred
17 OF3	FIFO 3 threshold Overflow interrupt flag 0 Interrupt has not occurred 1 Interrupt has occurred
18 EMP3	FIFO 3 Empty interrupt flag 0 Interrupt has not occurred 1 Interrupt has occurred
19 FULL3	FIFO 3 Full interrupt flag 0 Interrupt has not occurred 1 Interrupt has occurred
20 OR2	FIFO 2 Overrun interrupt flag A read of this bit clears it. 0 Interrupt has not occurred 1 Interrupt has occurred
21 OF2	FIFO 2 threshold Overflow interrupt flag 0 Interrupt has not occurred 1 Interrupt has occurred
22 EMP2	FIFO 2 Empty interrupt flag 0 Interrupt has not occurred 1 Interrupt has occurred
23 FULL2	FIFO 2 Full interrupt flag 0 Interrupt has not occurred 1 Interrupt has occurred
24 OR1	FIFO 1 Overrun interrupt flag A read of this bit clears it 0 Interrupt has not occurred 1 Interrupt has occurred
25 OF1	FIFO 1 threshold Overflow interrupt flag 0 Interrupt has not occurred 1 Interrupt has occurred
26 EMP1	FIFO 1 Empty interrupt flag 0 Interrupt has not occurred 1 Interrupt has occurred
27 FULL1	FIFO 1 Full interrupt flag 0 Interrupt has not occurred 1 Interrupt has occurred
28 OR0	FIFO 0 Overrun interrupt flag A read of this bit clears it. 0 Interrupt has not occurred 1 Interrupt has occurred
29 OF0	FIFO 0 threshold Overflow interrupt flag 0 Interrupt has not occurred 1 Interrupt has occurred

Table 412. FST field descriptions(Continued)

Field	Description
30 EMP0	FIFO 0 Empty interrupt flag 0 Interrupt has not occurred 1 Interrupt has occurred
31 FULL0	FIFO 0 Full interrupt flag 0 Interrupt has not occurred 1 Interrupt has occurred

35.8.15 FIFO Right aligned data *n* (FR*n*)

Address: 0x0080 + *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	0	0	0	0	0	0	0	0	0	0	0	N_CH				
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	0	0	0	0	DATA											
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 333. FIFO Right aligned data *n* (FR*n*)

Table 413. FR*n* field descriptions

Field	Description
11:15 N_CH	Number of stored channel 0xxxx Result comes from an ADC_1 channel 1xxxx Result comes from an ADC_0 channel
20:31 DATA	Data of stored channel

35.8.16 FIFO signed Left aligned data *n* (FL*n*)

Address: 0x00A0 + *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	0	0	0	0	0	0	0	0	0	0	0	N_CH				
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	0	DATA												0	0	0
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 334. FIFO signed Left aligned data *n* (FL*n*)

Table 414. FL*n* field descriptions

Field	Description
11:15 N_CH	Number of stored channel 0xxxx Result comes from an ADC_1 channel 1xxxx Result comes from an ADC_0 channel
17:28 DATA	Data of stored channel

35.8.17 Cross triggering unit error flag register (CTUEFR)

Address: 0x00C0 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	ET_OE	T1_OE	T0_OE	ADC_OE	TGS_OSM	MRS_O	ICE	SM_TO	MRS_RE
W								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 335. Cross triggering unit error flag register (CTUEFR)

Table 415. CTUEFR field descriptions

Field	Description
7 ET_OE	External Trigger generation Overrun Error 0 Error has not occurred 1 Error has occurred
8 T1_OE	Timer 1 trigger generation Overrun Error 0 Error has not occurred 1 Error has occurred

Table 415. CTUEFR field descriptions(Continued)

Field	Description
9 T0_OE	Timer 0 trigger generation Overrun Error 0 Error has not occurred 1 Error has occurred
10 ADC_OE	ADC command generation Overrun Error 0 Error has not occurred 1 Error has occurred
11 TGS_OSM	TGS Overrun in Sequential Mode 0 Error has not occurred 1 Error has occurred
12 MRS_O	Master Reload Signal Overrun 0 Error has not occurred 1 Error has occurred
13 ICE	Invalid Command Error 0 Error has not occurred 1 Error has occurred
14 SM_TO	Trigger Overrun (more than 8 EV) in TGS Sequential Mode 0 Error has not occurred 1 Error has occurred
15 MRS_RE	Master Reload Signal Reload Error 0 Error has not occurred 1 Error has occurred

35.8.18 Cross triggering unit interrupt flag register (CTUIFR)

Address: 0x00C2 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	ADC_I	T7_I	T6_I	T5_I	T4_I	T3_I	T2_I	T1_I	T0_I	MRS_I
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 336. Cross triggering unit interrupt flag register (CTUIFR)

Table 416. CTUIFR field descriptions

Field	Description
6 ADC_I	ADC command interrupt flag 0 Interrupt has not occurred 1 Interrupt has occurred
7 T7_I	Trigger 7 interrupt flag 0 Interrupt has not occurred 1 Interrupt has occurred
8 T6_I	Trigger 6 interrupt flag 0 Interrupt has not occurred 1 Interrupt has occurred

Table 416. CTUIFR field descriptions(Continued)

Field	Description
9 T5_I	Trigger 5 interrupt flag 0 Interrupt has not occurred 1 Interrupt has occurred
10 T4_I	Trigger 4 interrupt flag 0 Interrupt has not occurred 1 Interrupt has occurred
11 T3_I	Trigger 3 interrupt flag 0 Interrupt has not occurred 1 Interrupt has occurred
12 T2_I	Trigger 2 interrupt flag 0 Interrupt has not occurred 1 Interrupt has occurred
13 T1_I	Trigger 1 interrupt flag 0 Interrupt has not occurred 1 Interrupt has occurred
14 T0_I	Trigger 0 interrupt flag 0 Interrupt has not occurred 1 Interrupt has occurred
15 MRS_I	MRS Interrupt flag 0 Interrupt has not occurred 1 Interrupt has occurred

35.8.19 Cross triggering unit interrupt/DMA register (CTUIR)

Address: 0x00C4 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	T7_IE	T6_IE	T5_IE	T4_IE	T3_IE	T2_IE	T1_IE	T0_IE	0	0	0	0	0	MRS_DMAE	MRS_IE	IEE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 337. Cross triggering unit interrupt/DMA register (CTUIR)

Table 417. CTUIR field descriptions

Field	Description
0 T7_IE	Trigger 7 Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
1 T6_IE	Trigger 6 Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
2 T5_IE	Trigger 5 Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled

Table 417. CTUIR field descriptions(Continued)

Field	Description
3 T4_IE	Trigger 4 Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
4 T3_IE	Trigger 3 Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
5 T2_IE	Trigger 2 Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
6 T1_IE	Trigger 1 Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
7 T0_IE	Trigger 0 Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
13 MRS_DMAE	DMA transfer Enable on MRS occurrence if GRE bit is set 0 Interrupt disabled 1 Interrupt enabled
14 MRS_IE	MRS Interrupt Enable 0 Interrupt disabled 1 Interrupt enabled
15 IEE	Interrupt Error Enable 0 Interrupt disabled 1 Interrupt enabled

35.8.20 Control ON time register (COTR)

Address: 0x00C6 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	COTR							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 338. Control ON time register (COTR)

Table 418. COTR field descriptions

Field	Description
8:15 COTR	Control ON-Time and Guard Time for external trigger

35.8.21 Cross triggering unit control register (CTUCR)

Address: 0x00C8 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	T7_SG	T6_SG	T5_SG	T4_SG	T3_SG	T2_SG	T1_SG	T0_SG	CRU_ADC_R	CTU_ODIS	DFE	CGRE	FGRE	MRS_SG	GRE	TGSISR_RE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 339. Cross triggering unit control register (CTUCR)

Table 419. CTUCR field descriptions

Field	Description
0 T7_SG	Trigger 7 Software Generated 0 Trigger has not been generated 1 Trigger has been generated
1 T6_SG	Trigger 6 Software Generated 0 Trigger has not been generated 1 Trigger has been generated
2 T5_SG	Trigger 5 Software Generated 0 Trigger has not been generated 1 Trigger has been generated
3 T4_SG	Trigger 4 Software Generated 0 Trigger has not been generated 1 Trigger has been generated
4 T3_SG	Trigger 3 Software Generated 0 Trigger has not been generated 1 Trigger has been generated
5 T2_SG	Trigger 2 Software Generated 0 Trigger has not been generated 1 Trigger has been generated
6 T1_SG	Trigger 1 Software Generated 0 Trigger has not been generated 1 Trigger has been generated
7 T0_SG	Trigger 0 Software Generated 0 Trigger has not been generated 1 Trigger has been generated
8 CRU_ADC_R	CTU/ADC state machine Reset
9 CTU_ODIS	CTU Output Disable
10 DFE	Digital Filter Enable **Note:** This bit can be read by software and can only be set to 1 by software.

Table 419. CTUCR field descriptions(Continued)

Field	Description
11 CGRE	Clear GRE **Note:** This bit can only be set to 1 by software.
12 FGRE	Flag GRE
13 MRS_SG	MRS Software Generated
14 GRE	General Reload Enable **Note:** This bit can be read by software and can only be set to 1 by software.
15 TGSISR_RE	TGS Input Selection Register Reload Enable **Note:** This bit can be read by software and can only be set to 1 by software.

35.8.22 Cross triggering unit digital filter (CTUDF)

Address: 0x00CA 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	N							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 340. Cross triggering unit digital filter (CTUDF)

Table 420. CTUDF field descriptions

Field	Description
8:15 N	Digital Filter value (the external signal is considered at 1 if it is latched N time at 1 and is considered at 0 if it is latched N time at 0)

36 Enhanced Motor Control Timer (eTimer)

36.1 Introduction

36.1.1 Overview

An eTimer module provides:

- 6 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.

36.1.2 Features

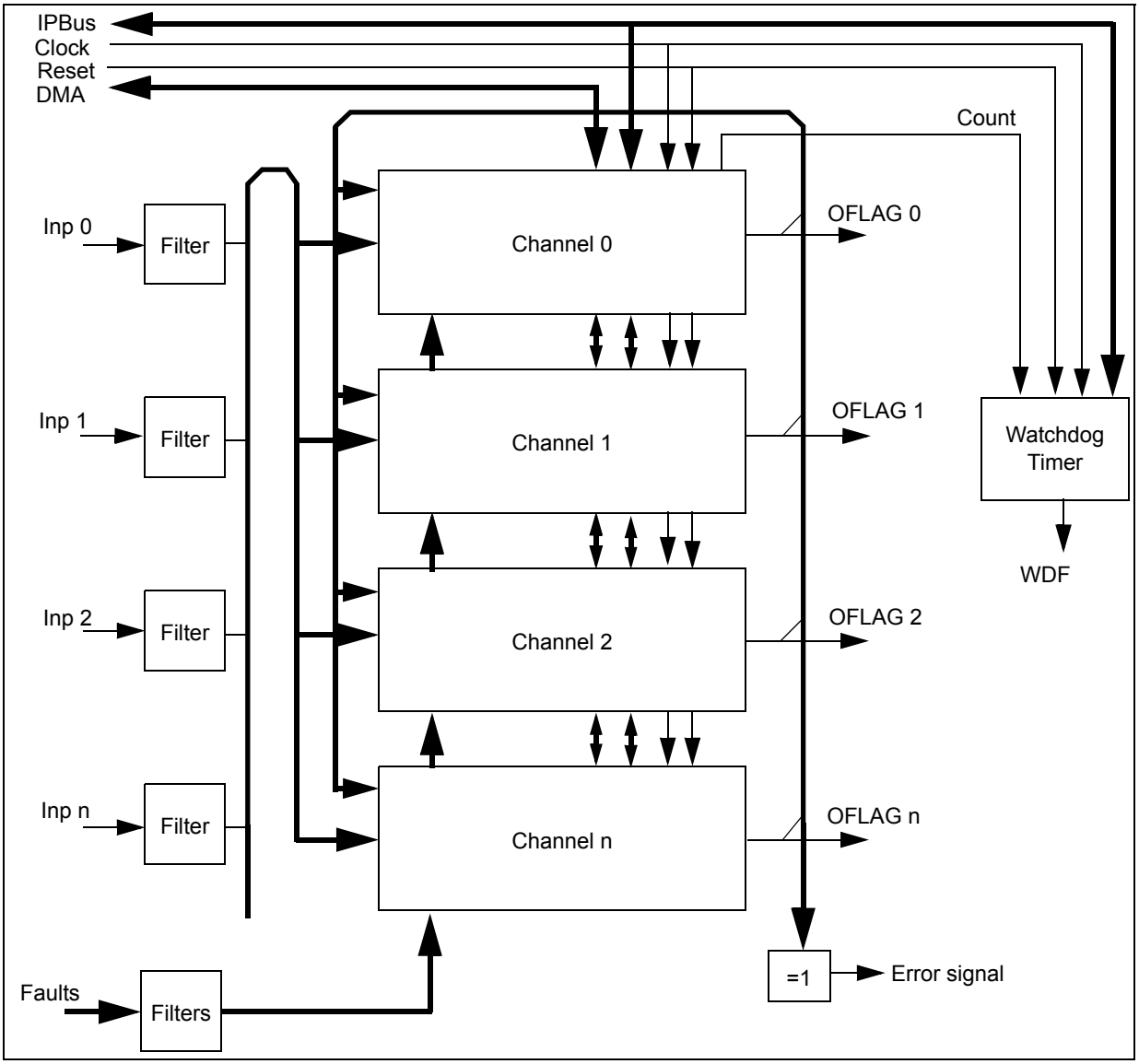
The eTimer module design includes these distinctive features:

- 16-bit counters/timers: 6
- Count up/down
- Counters are cascadable
- Enhanced programmable up/down modulo counting
- 2 fault inputs.
- 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.

36.1.3 Module block diagram

The eTimer block diagram is shown in *Figure 341*.

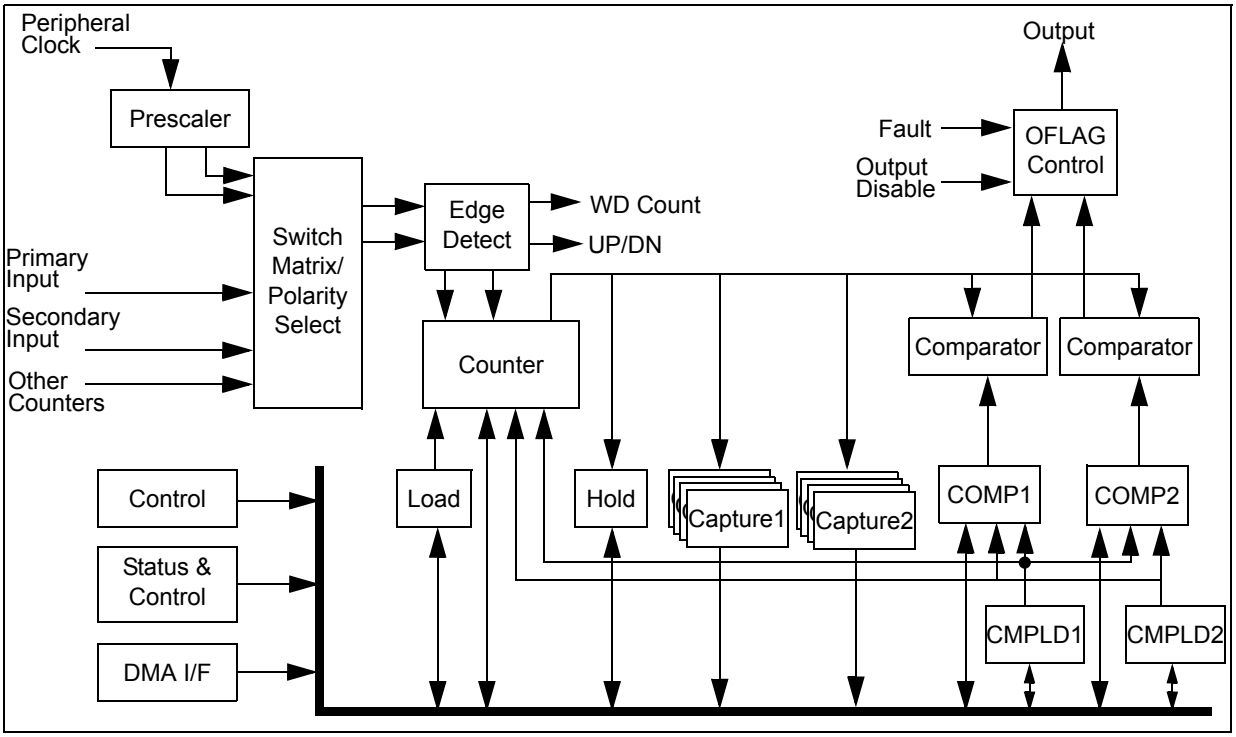
Figure 341. eTimer block diagram



36.1.4 Channel block diagram

Each of the timer/counter channels within the eTimer are shown in *Figure 342*.

Figure 342. eTimer channel block diagram



36.2 External signal descriptions

The eTimer module has:

- 6 external signals that can be used as either inputs or outputs
- Interfaces to the Peripheral Bus

36.2.1 TIO[*n*:0] - Timer Input/Outputs

These pins can be independently configured to be either timer input sources or output flags. In *Figure 342*, the TIO signals are shown as the Primary Inputs and the Outputs from the OFLAG Control block

36.3 Functional Description

36.3.1 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.
- The response of the OFLAG output to a fault input 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.

36.3.2 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.

36.3.2.1 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.

36.3.2.2 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 36.3.2.9: CASCADE-COUNT Mode*, through *Section 36.3.2.12: VARIABLE-FREQUENCY PWM Mode*, for additional capabilities of this operating mode.

36.3.2.3 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.

36.3.2.4 GATED-COUNT Mode

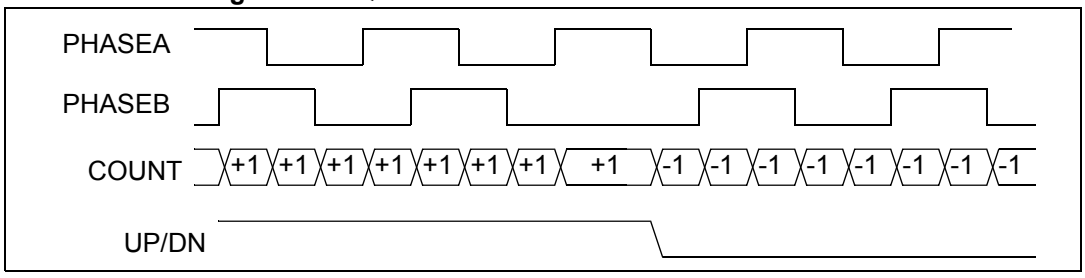
If the CNTMODE field is set to ‘011’, the counter will count while the selected secondary input signal is high. This mode is used to time the duration of external events. If the selected input is inverted by setting the SIPS bit, then the counter will count while the selected secondary input is low.

36.3.2.5 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 343 shows a timing diagram illustrating the basic operation of a quadrature incremental position encoder.

Figure 343. Quadrature Incremental Position Encoder



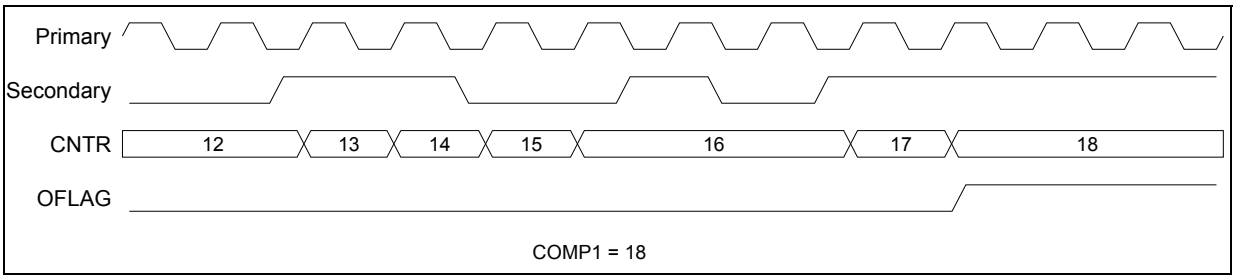
36.3.2.6 SIGNED-COUNT Mode

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).

36.3.2.7 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.

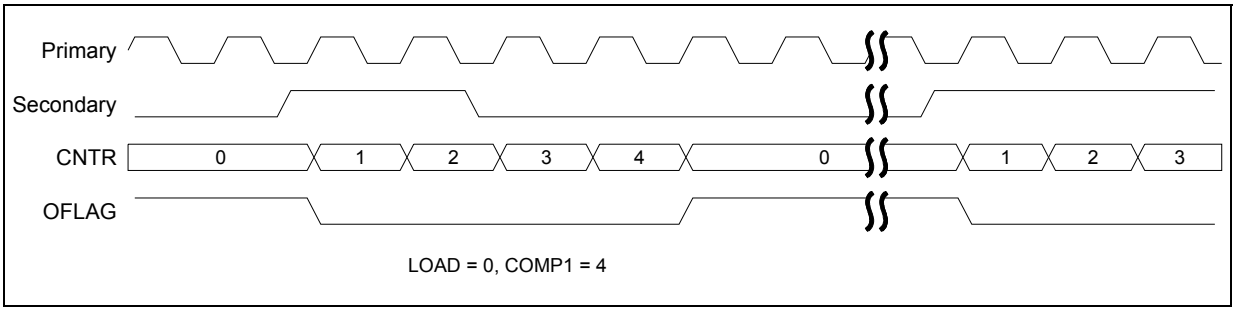
Figure 344. Triggered Count Mode (Length=1)



36.3.2.8 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 345. One-Shot Mode (Length=1)



36.3.2.9 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.

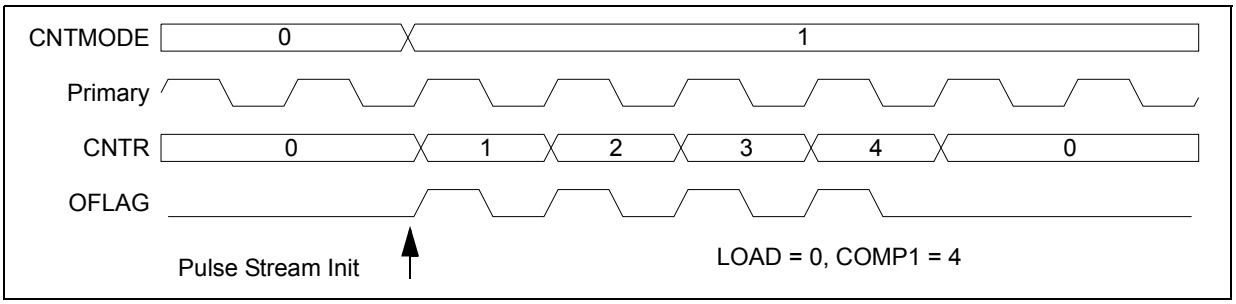
A channel can be cascaded with any other channel, but you can't cascade more than 2 channels together. You can create separate cascades of pairs of channels. For example, you can cascade channels 0 and 1 and separately cascade channels 6 and 5. You can't cascade channels 0, 1, and 5.

36.3.2.10 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 346. Pulse Output Mode



36.3.2.11 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.

36.3.2.12 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 347. Variable PWM Waveform

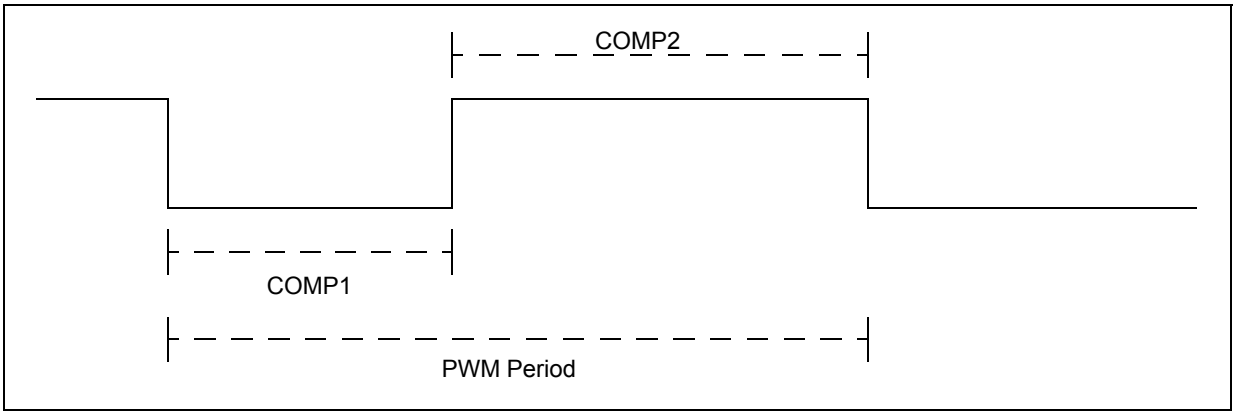
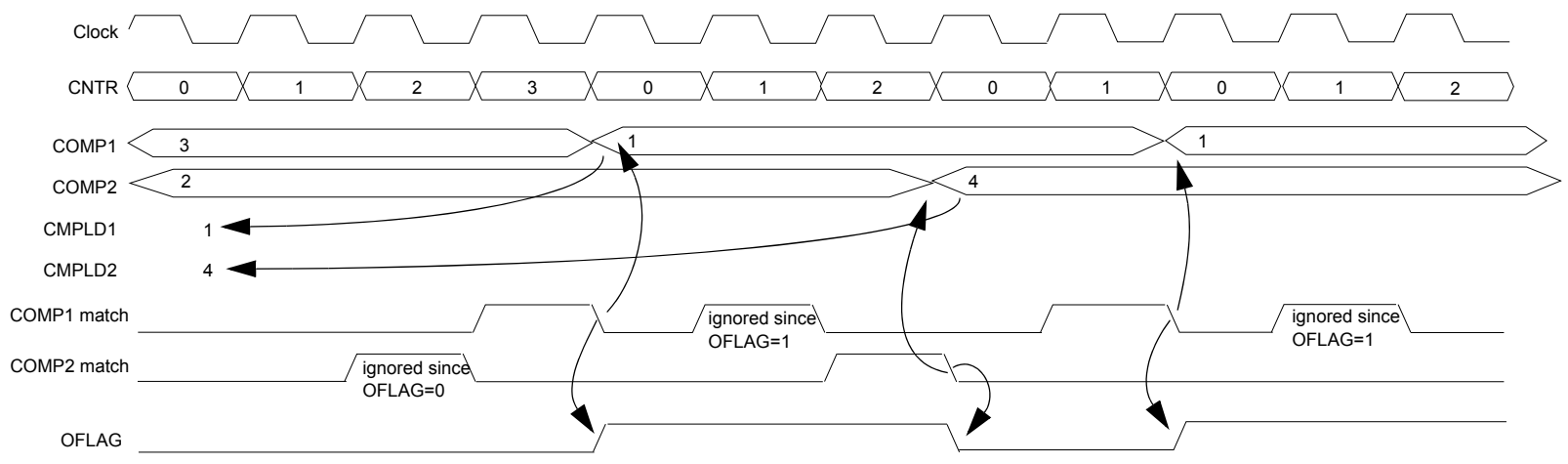


Figure 348. Variable Frequency PWM Mode Timing



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

36.3.2.13 Usage of Compare Registers

The dual compare registers (COMP1 and COMP2) provide a bidirectional modulo count capability.

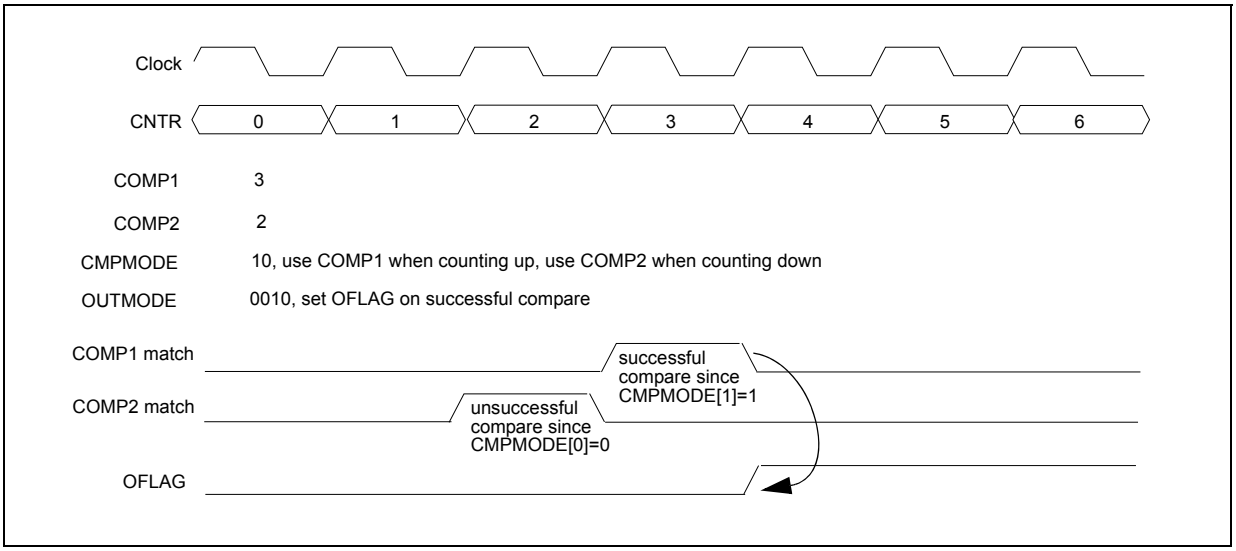
The COMP1 register should be set to the desired maximum count value or 0xFFFF to indicate the maximum unsigned value prior to roll-over, and the COMP2 register should be set to the minimum count value or 0x0000 to indicate the minimum unsigned value prior to roll-under.

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 349. Compare Register and OFLAG Timing



36.3.2.14 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 347*.

36.3.2.15 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).

36.3.2.16 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 350. OFLAG timing during non-continuous counting

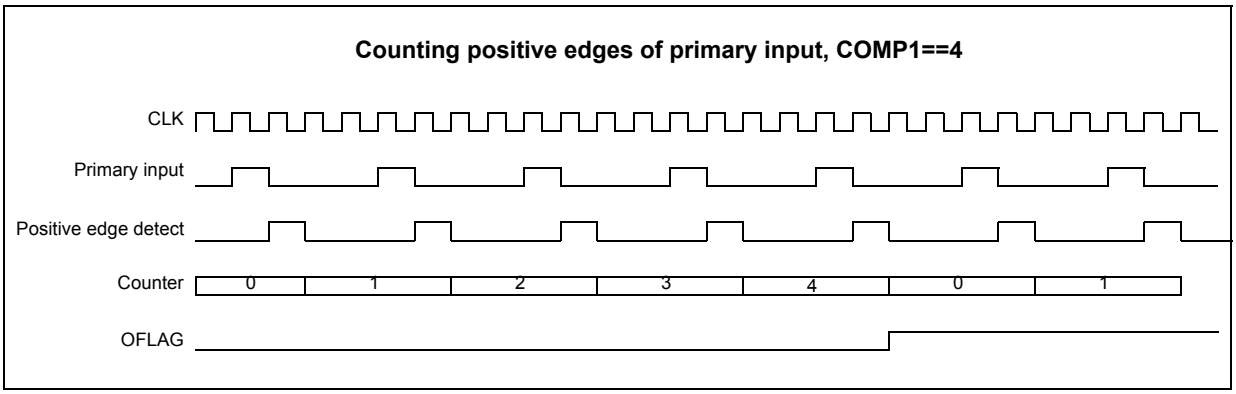
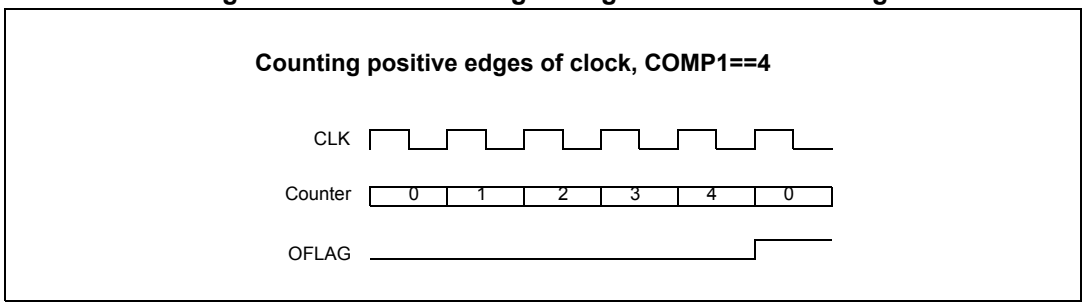


Figure 351. OFLAG timing during continuous counting



36.3.3 Other Features

36.3.3.1 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.

36.3.3.2 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.

36.3.3.3 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 436*. 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.

36.3.3.4 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.

36.3.3.5 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.

36.4 Memory Map and Registers

36.4.1 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 "Device Configuration" chapter.

36.4.2 Module Memory Map

Table 421. 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 36.4.4.1*
0x0002 + n*0x20	COMP2n	Compare Register 2	*Section 36.4.4.2*
0x0004 + n*0x20	CAPT1n	Capture Register 1	*Section 36.4.4.3*
0x0006 + n*0x20	CAPT2n	Capture Register 2	*Section 36.4.4.4*
0x0008 + n*0x20	LOADn	Load Register	*Section 36.4.4.5*
0x000A + n*0x20	HOLDn	Hold Register	*Section 36.4.4.6*
0x000C + n*0x20	CNTRn	Counter Register	*Section 36.4.4.7*
0x000E + n*0x20	CTRL1n	Control Register 1	*Section 36.4.4.8*
0x0010 + n*0x20	CTRL2n	Control Register 2	*Section 36.4.4.9*
0x0012 + n*0x20	CTRL3n	Control Register 3	*Section 36.4.4.10*
0x0014 + n*0x20	STSn	Status Register	*Section 36.4.4.11*

Table 421. eTimer Memory Map(Continued)

Address offset	Reg Name	Description	Location
0x0016 + n*0x20	INTDMAn	Interrupt and DMA Enable Register	*Section 36.4.4.12*
0x0018 + n*0x20	CMPLD1n	Comparator Load Register 1	*Section 36.4.4.13*
0x001A + n*0x20	CMPLD2n	Comparator Load Register 2	*Section 36.4.4.14*
0x001C + n*0x20	CCCTRLn	Compare and Capture Control Register	*Section 36.4.4.15*
0x001E + n*0x20	FILTn	Input Filter Register	*Section 36.4.4.16*
Watchdog timer registers(1)			
0x0100	WDTOL	Watchdog Time-out Low World Register	*Section 36.4.5.1*
0x0102	WDTOH	Watchdog Time-out High World Register	*Section 36.4.5.2*
Fault registers			
0x0104	FCTRL	Fault Control Register	*Section 36.4.6.1*
0x0106	FSTS	Fault Status Register	*Section 36.4.6.2*
0x0108	FFILT	Fault Filter Register	*Section 36.4.6.3*
Configuration Registers			
0x010C	ENBL	Channel Enable Register	*Section 36.4.7.1*
0x0110	DREQ0	DMA Request 0 Select Register	*Section 36.4.7.2*
0x0112	DREQ1	DMA Request 1 Select Register	*Section 36.4.7.3*

1. If applicable. Not all eTimer module instantiations implement a watchdog timer.

36.4.3 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, a set for the fault inputs, 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.

36.4.4 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.

36.4.4.1 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 36.3.2.13: 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 352. Compare Register 1n (COMP1n)

Table 422. Compare Register 1n (COMP1n) field descriptions

Field	Description
0:15 COMP1	Compare Register1

36.4.4.2 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 36.3.2.13: Usage of Compare Registers*.

Offset: 0x0002 + 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	COMP2															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 353. Compare Register 2n (COMP2n)

Table 423. Compare Register 2n (COMP2n) field descriptions

Field	Description
0:15 COMP2	Compare Register2

36.4.4.3 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

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	CAPT1															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 354. Capture Register 1n (CAPT1n)

Table 424. Capture Register 1n (CAPT1n) field descriptions

Field	Description
0:15 CAPT1	Capture Register1

36.4.4.4 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 355. Capture Register 2n (CAPT2n)

Table 425. Capture Register 2n (CAPT2n) field descriptions

Field	Description
0:15 CAPT2	Capture Register2

36.4.4.5 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 356. Load Register n (LOADn)

Table 426. Load Register n (LOADn) field descriptions

Field	Description
0:15 LOAD	Load Register

36.4.4.6 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

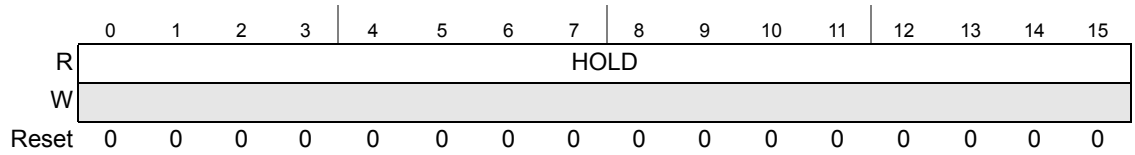


Figure 357. Hold Register n (HOLDn)

Table 427. Hold Register n (HOLDn) field descriptions

Field	Description
0:15 HOLD	Hold Register

36.4.4.7 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

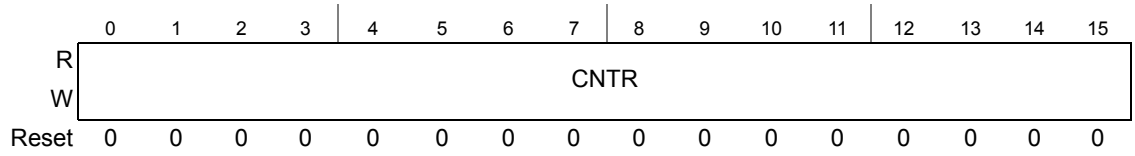


Figure 358. Counter Register n (CNTRn)

Table 428. Counter Register n (CNTRn) field descriptions

Field	Description
0:15 CNTR	Counter Register

36.4.4.8 Control Register 1n (CTRL1n)

Offset: 0x000E + n*0x20 (n=0 to 5) Access: R/W

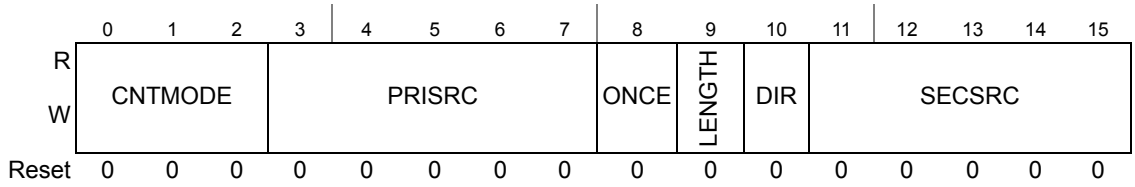


Figure 359. Control Register 1n (CTRL1n)

Table 429. 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 Reserved 01001 Reserved 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 429. 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 Reserved 01001 Reserved 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.

36.4.4.9 Control Register 2n (CTRL2n)

Offset: 0x0010 + n*0x20 (n=0 to 5) Access: R/W

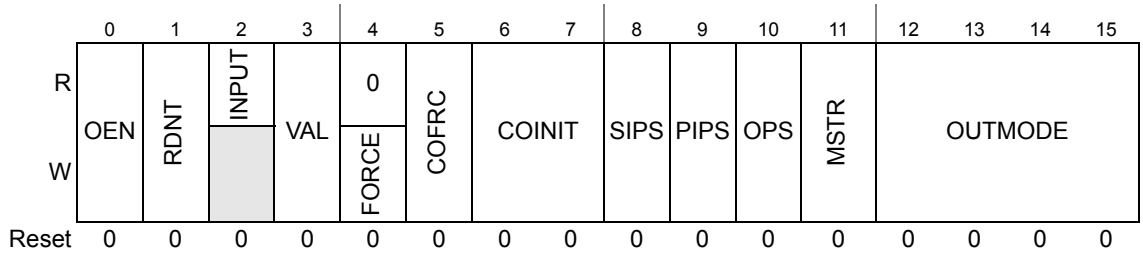


Figure 360. Control Register 2n (CTRL2n)

Table 430. 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 430. 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 430. 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

36.4.4.10 Control Register 3n (CTRL3n)

Offset: 0x0012 + n*0x20 (n=0 to 5) Access: R/W

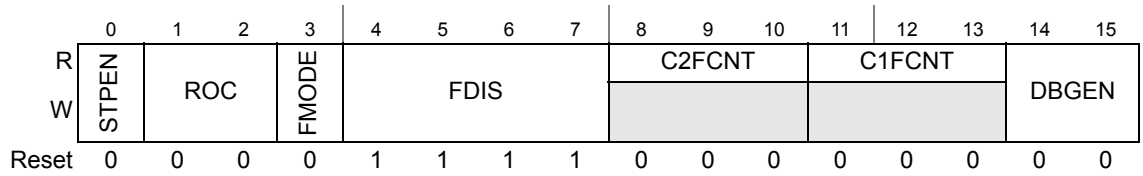


Figure 361. Control Register 3n (CTRL3n)

Table 431. 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.

Table 431. Control Register 3n (CTRL3n) field descriptions(Continued)

Field	Description
3 FMODE	Fault Safing Mode This bit determines the safing mode of the OFLAG output signal under fault conditions. 1 Output is forced to a logic 0 subject to the OPS bit. 0 Output is tristated.
4:7 FDIS	Fault Disable Mask Each of these four bits is one-to-one associated with the four FAULT inputs. The channel output will be safed when there is a fault condition on a FAULTx input and a 1 in the corresponding bit of the FDIS field. 1 The channel OFLAG output is affected by this FAULT pin. 0 The channel OFLAG output is not affected by this FAULT pin.
8:10 C2FCNT	CAPT2 FIFO Word Count This field reflects the number of words in the CAPT2 FIFO
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.

36.4.4.11 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 362. Status Register n (STSn)

Table 432. 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).
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.

36.4.4.12 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 363. Interrupt and DMA Enable Register n (INTDMAn)

Table 433. 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.
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.

Table 433. Interrupt and DMA Enable Register n (INTDMAn) field descriptions

Field	Description
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.

36.4.4.13 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 36.3.2.14: 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 364. Comparator Load 1n (CMPLD1n)

Table 434. CMPLD1n field descriptions

Field	Description
0:15 CMPLD1	Comparator Load Register 1

36.4.4.14 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 36.3.2.14: 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 365. Comparator Load 2n (CMPLD2n)

Table 435. CMPLD2n field descriptions

Field	Description
0:15 CMPLD2	Comparator Load Register 2

36.4.4.15 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 366. Compare and Capture Control Register n (CCCTRLn)

Table 436. 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.

Table 436. Compare and Capture Control Register n (CCCTRLn) field descriptions(Continued)

Field	Description
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
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

Table 436. Compare and Capture Control Register n (CCCTRLn) field descriptions(Continued)

Field	Description
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.

36.4.4.16 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.

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 367. Input Filter Register n (FILTn)

Table 437. 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 36.4.4.16.1: 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 36.4.4.16.1: Input Filter Considerations*.

36.4.4.16.1 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.

36.4.5 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.

36.4.5.1 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 368. Watchdog Time-out Low Word Register (WDTOL)

Table 438. 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 36.3.3.5: Watchdog Timer,* for more information on the use of the watchdog timer.

36.4.5.2 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 369. Watchdog Time-out High Word Register (WDTOH)

Table 439. 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 36.3.3.5: Watchdog Timer,* for more information on the use of the watchdog timer.

36.4.6 Fault Registers

The base address of the fault registers is equal to the base address of the eTimer plus an offset of 0x0104.

36.4.6.1 Fault Control Register (FCTRL)

Offset: 0x104 Access: R/W

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	FTEST	Reserved				0	0	0	0	FLVL			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. Writes to these bits is possible but this will have no functional impact.

Figure 370. Fault Control Register (FCTRL)

Table 440. FCTRL field descriptions

Field	Description
3 FTEST	Fault Test This 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 eTimer outputs depending on the FDIS settings. Clearing this bit removes the simulated fault condition. 1 Cause a simulated fault. 0 No fault.
12:15 FLVL	Fault Active Logic 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.

36.4.6.2 Fault Status Register (FSTS)

Offset: 0x106 Access: R/W

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	FFPIN				0	0	0	0	FFLAG			
W																
Reset	0	0	0	0	U	U	U	U	0	0	0	0	0	0	0	0

Figure 371. Fault Status Register (FSTS)

Table 441. FSTS field descriptions

Field	Description
4:7 FFPIN	Filtered Fault Pin 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 input. A reset has no effect on FFPIN.
12:15 FFLAG	Fault Flag These read only flags are set within 2 IPBus clock cycles after a transition to active on the FAULTx input pin. Clear FFLAGx by writing a logic 1 to it. A reset clears FFLAG. 1 Fault on the FAULTx pin. 0 No fault on the FAULTx pin.

36.4.6.3 Fault Filter Registers (FFILT)

Offset: 0x108 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	FFILT_CNT			FFILT_PER							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 372. Fault Filter Register (FFILT)

Table 442. FFILT field descriptions

Field	Description
5:7 FFILT_CNT	FAULT Filter Sample Count These bits represent the number of consecutive samples that must agree prior to the fault input filter accepting an input transition. A value of 0 represents 3 samples. A value of 7 represents 10 samples. The value of FFILT_CNT affects the input latency as described in *Section 36.4.4.16.1: Input Filter Considerations*.
8:15 FFILT_PER	FAULT Filter Sample Period These bits represent the sampling period (in IPBus clock cycles) of the FAULT input signals. Each input is sampled multiple times at the rate specified by FFILT_PER. If FFILT_PER is 0x00 (default), then the fault input filter is bypassed. The value of FFILT_PER affects the input latency as described in *Section 36.4.4.16.1: Input Filter Considerations*.

36.4.7 Configuration Registers

The base address of the configuration registers is equal to the base address of the eTimer plus an offset of 0x010C.

36.4.7.1 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 373. Channel Enable Register (ENBL)

Table 443. 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.

36.4.7.2 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 374. DMA Request 0 Select Register (DREQ0)

Table 444. 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

36.4.7.3 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 375. DMA Request 1 Select Register (DREQ1)

Table 445. 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 445. 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

36.5 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.

36.6 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) and the fault logic each 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.

36.7 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 446. 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.

36.8 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 447. 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

37 System Timer Module (STM)

37.1 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.

37.1.1 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 system clock divided by an 8-bit prescale value (1 to 256).

37.1.2 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

37.1.3 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.

37.2 External signal description

The STM does not have any external interface signals.

37.3 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.

37.3.1 Memory map

The STM memory map is shown in *Table 448*.

Table 448. STM memory map

Address offset	Register	Location
0x0000	STM Control Register (STM_CR)	*Section 37.3.2.1*
0x0004	STM Counter Value (STM_CNT)	*Section 37.3.2.2*
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 37.3.2.3*
0x0014 + n*0x10	STM Channel n Interrupt Register (STM_CIRn)	*Section 37.3.2.4*
0x0018 + n*0x10	STM Channel n Compare Register (STM_CMPn)	*Section 37.3.2.5*
0x001C + n*0x10	Reserved	
0x004C–0x3FFF	Reserved	

37.3.2 Register descriptions

37.3.2.1 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 376. STM Control Register (STM_CR)

Table 449. STM_CR field descriptions

Field	Description
16:23 CPS	Counter Prescaler. Selects the clock divide value for the prescaler (1 - 256). 0x00 Divide system clock by 1 0x01 Divide system clock by 2 ... 0xFF Divide system 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

37.3.2.2 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 377. STM Count Register (STM_CNT)

Table 450. 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 system clock divided by the prescale value.

37.3.2.3 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 378. STM Channel Control Register (STM_CCR*n*)

Table 451. STM_CCR*n* field descriptions

Field	Description
31 CEN	Channel Enable. 0 The channel is disabled 1 The channel is enabled

37.3.2.4 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 379. STM Channel Interrupt Register (STM_CIR*n*)

Table 452. 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

37.3.2.5 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 380. STM Channel Compare Register (STM_CMP*n*)

Table 453. 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.

37.4 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 system 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 system clock is stopped.

38 Software Watchdog Timer (SWT)

38.1 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.

38.1.1 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.

38.1.2 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

38.1.3 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 38.3.1.1: 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 38.3.1.1: SWT Control Register (SWT_CR)*. If the STP bit is set, the counter is stopped in stop mode, otherwise it continues to run.

38.2 External signal description

The SWT module does not have any external interface signals.

38.3 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 454*.

Table 454. SWT memory map

Offset	Register	Location
0x0000	SWT Control Register (SWT_CR)	*Section 38.3.1.1*
0x0004	SWT Interrupt Register (SWT_IR)	*Section 38.3.1.2*
0x0008	SWT Time-Out Register (SWT_TO)	*Section 38.3.1.3*
0x000C	SWT Window Register (SWT_WN)	*Section 38.3.1.4*
0x0010	SWT Service Register (SWT_SR)	*Section 38.3.1.5*
0x0014	SWT Counter Output Register (SWT_CO)	*Section 38.3.1.6*
0x0018	SWT Service Key Register (SWT_SK)	*Section 38.3.1.7*
0x001C–0x3FFF	Reserved	

38.3.1 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.

38.3.1.1 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 381. SWT Control Register (SWT_CR)

Table 455. 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 crossbar switch. 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 Reserved—do not use. Writing this value can cause the SWT not to be serviced. 11 Reserved—do not use. Writing this value can cause the SWT not to be serviced.
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

38.3.1.2 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 382. SWT Interrupt Register (SWT_IR)

Table 456. 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.

38.3.1.3 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: 0x008 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	WTO							
Reset(1)	* * * *	* * * *	* * * *	* * * *	* * * *	* * * *	* * * *	* * * *

1. The reset value of the SWT_TO register is device specific. See the device configuration chapter.

Figure 383. SWT Time-Out Register (SWT_TO)

Table 457. 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.

38.3.1.4 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: 0x00C 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 384. SWT Window Register (SWT_WN)

Table 458. 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.

38.3.1.5 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: 0x010 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 385. SWT Service Register (SWT_SR)

Table 459. 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 38.3.1.1: 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.

38.3.1.6 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: 0x014 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 386. SWT Counter Output Register (SWT_CO)

Table 460. 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.

38.3.1.7 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: 0x018 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 387. SWT Service Key Register (SWT_SK)

Table 461. 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.

38.4 Functional description

38.4.1 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.

38.4.1.1 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[p]. 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.

38.4.1.2 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.

38.4.1.3 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.

p. See SWT section in device configuration chapter.

38.4.2 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.

38.4.2.1 Hard lock

The hard lock is enabled by setting the SWT_CR[HLK] bit, which can only be cleared by a reset.

38.4.2.2 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.

38.4.3 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.

38.4.4 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

38.4.4.1 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.

38.4.4.2 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 30*. 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 30

$$SK_{n+1} = (17*SK_n+3) \bmod 2^{16}$$

38.4.4.3 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.

38.4.5 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.

38.4.6 Initialization

All registers should be initialized before setting the SWT_CR[WEN] bit to enable the watchdog. Registers can be initialized in any sequence.

39 Periodic Interrupt Timer (PIT)

39.1 Introduction

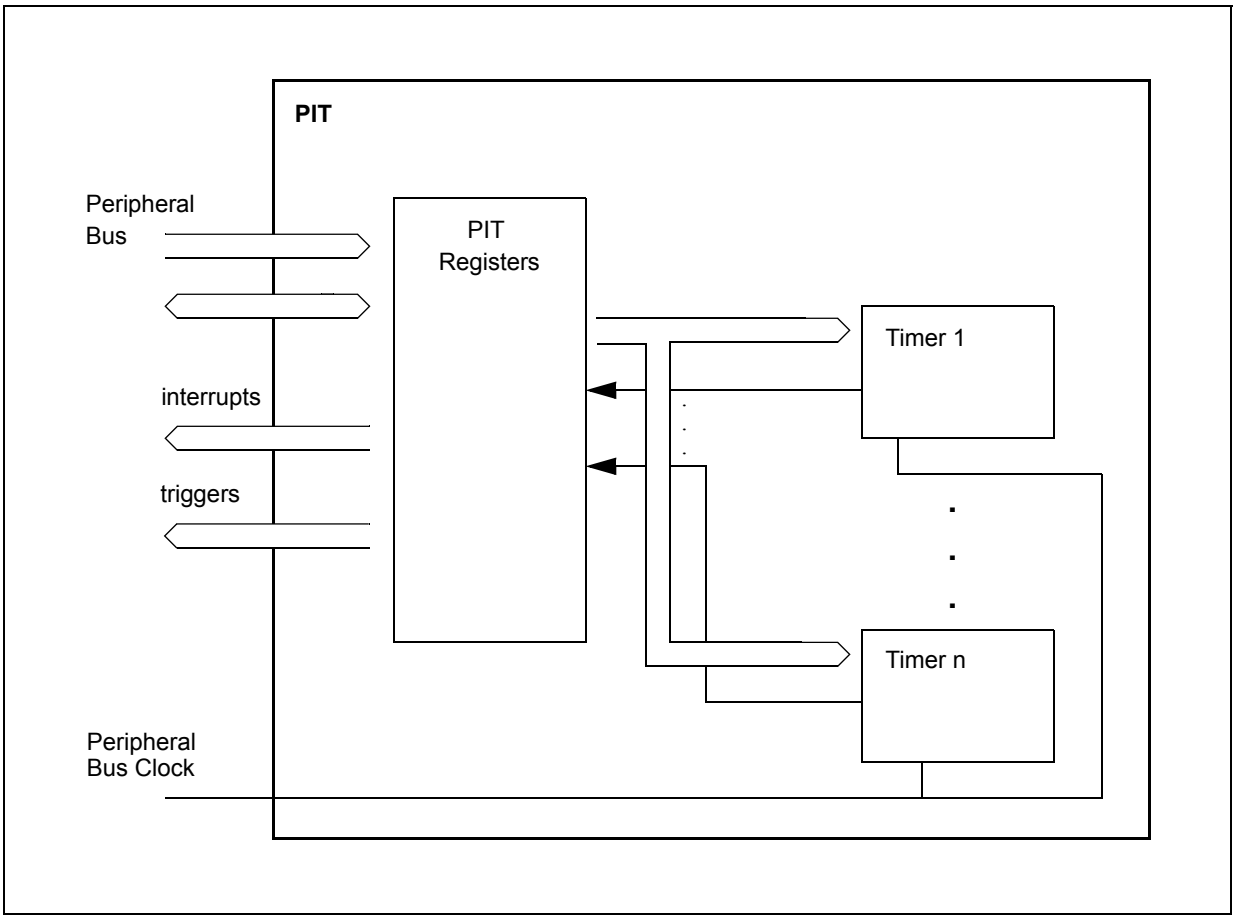
The PIT timer module is an array of timers that can be used to raise interrupts and trigger DMA channels.

Note: For the chip-specific implementation details of this module's instances see the Device Configuration chapter.

39.1.1 Block diagram

Figure 388 shows the PIT block diagram.

Figure 388. Block diagram of the PIT



Note: Refer to the Chip Configuration information for the number of PIT channels used in this MCU.

39.1.2 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

39.2 Signal description

This module has no external pins.

39.2.1 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 462. PIT memory map

Offset (hex)	Register name	Location
0x0000	PIT Control Register (MCR)	*Section 39.2.1.1*
0x0004–0x00FF	Reserved	
PIT registers per channel (n=0 to 3)		
0x0100 + n * 0x10	PIT Timer Load Value Register n (LDVALn)	*Section 39.2.1.2*
0x0104 + n * 0x10	PIT Current Timer Value Register n (CVALn)	*Section 39.2.1.3*
0x0108 + n * 0x10	PIT Timer Control Register n (TCTRLn)	*Section 39.2.1.4*
0x010C + n * 0x10	PIT Timer Flag Register n (TFLGn)	*Section 39.2.1.5*
0x0140–0x3FFF	Reserved	

39.2.1.1 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 389. PIT Module Control Register (MCR)

Table 463. 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.

39.2.1.2 PIT Timer Load Value Register *n* (LDVAL*n*)

Offset: 0x0100 + n * 0x10 (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	TSV[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	TSV[16:31]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 390. PIT Timer Load Value Register *n* (LDVAL*n*)

Table 464. LDVAL*n* field descriptions

Field	Description
0:31 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.

39.2.1.3 PIT Current Timer Value Register *n* (CVAL*n*)

Offset: 0x0104 + n * 0x10 (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	TVL[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	TVL[16:31]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 391. PIT Current Timer Value Register *n* (CVAL*n*)

Table 465. CVAL*n* field descriptions

Field	Description
0:31 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.

39.2.1.4 PIT Timer Control Register *n* (TCTRL*n*)

Offset: 0x0108 + n * 0x10 (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	0	0	0	0	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	TIE	TEN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 392. Timer Control Register (TCTRL*n*)

Table 466. TCTRL*n* field descriptions

Field	Description
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.

39.2.1.5 PIT Timer Flag Register *n* (TFLG*n*)

Offset: 0x010C + n * 0x10 (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	0	0	0	0	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 393. Timer Flag Register *n* (TFLG*n*)

Table 467. 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.

39.3 Functional description

This section provides the functional description of the module.

39.3.1 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.

39.3.1.1 Timers

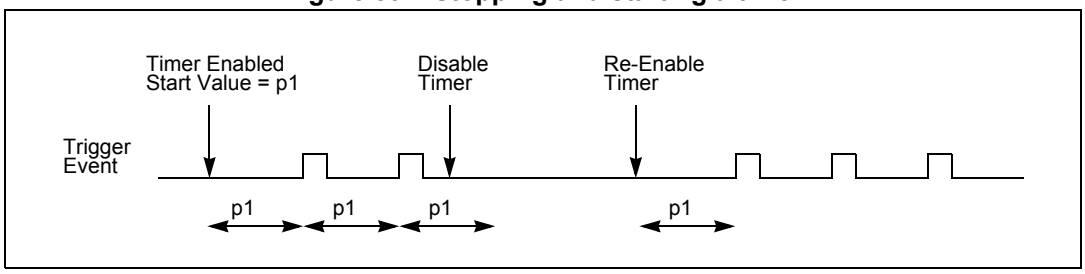
The timers generate triggers at periodic intervals, when enabled. They load their start values, as specified in their LDVAL registers, then count down until they reach 0. Then they load their respective start value again. Each time a timer reaches 0, it will generate a trigger pulse and set the interrupt 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.

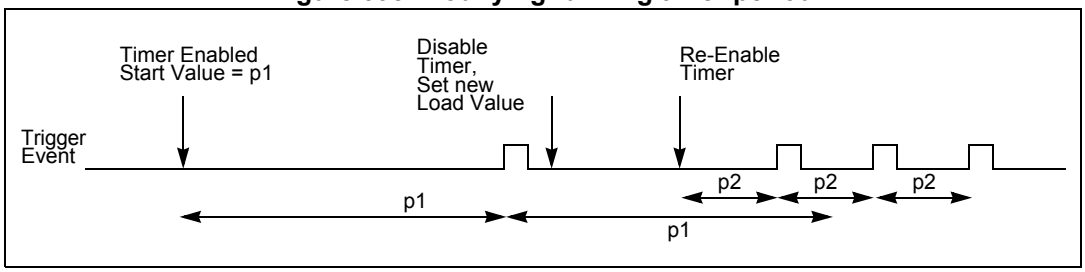
The counter period can be restarted, by first disabling, then enabling the timer with the TEN bit (see the following figure).

Figure 394. Stopping and starting a timer



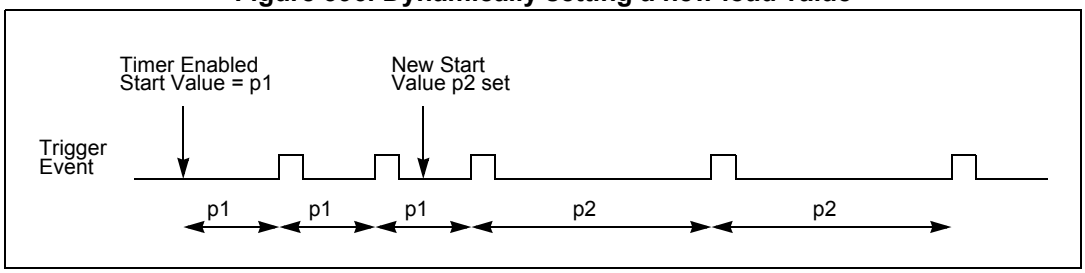
The counter period of a running timer can be modified, by first disabling the timer, setting a new load value and then enabling the timer again (see the following figure).

Figure 395. 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 396. Dynamically setting a new load value



39.3.1.2 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.

39.3.2 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.

39.3.3 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.

39.4 Initialization and application information

In the example configuration:

- The PIT clock has a frequency of 50 MHz.
- Timer 1 creates an interrupt every 5.12 ms.
- Timer 3 creates a trigger event every 30 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 5.12 ms/20 ns = 256000 cycles and timer 3 every 30 ms/20 ns = 1500000 cycles. The value for the LDVAL register trigger is calculated as:

LDVAL trigger = (period / clock period) - 1

This means LDVAL1 should be written with 0x0003E7FF, and LDVAL3 should be written with 0x0016E35F.

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 2

```
// turn on PIT
PIT_MCR = 0x00;
```

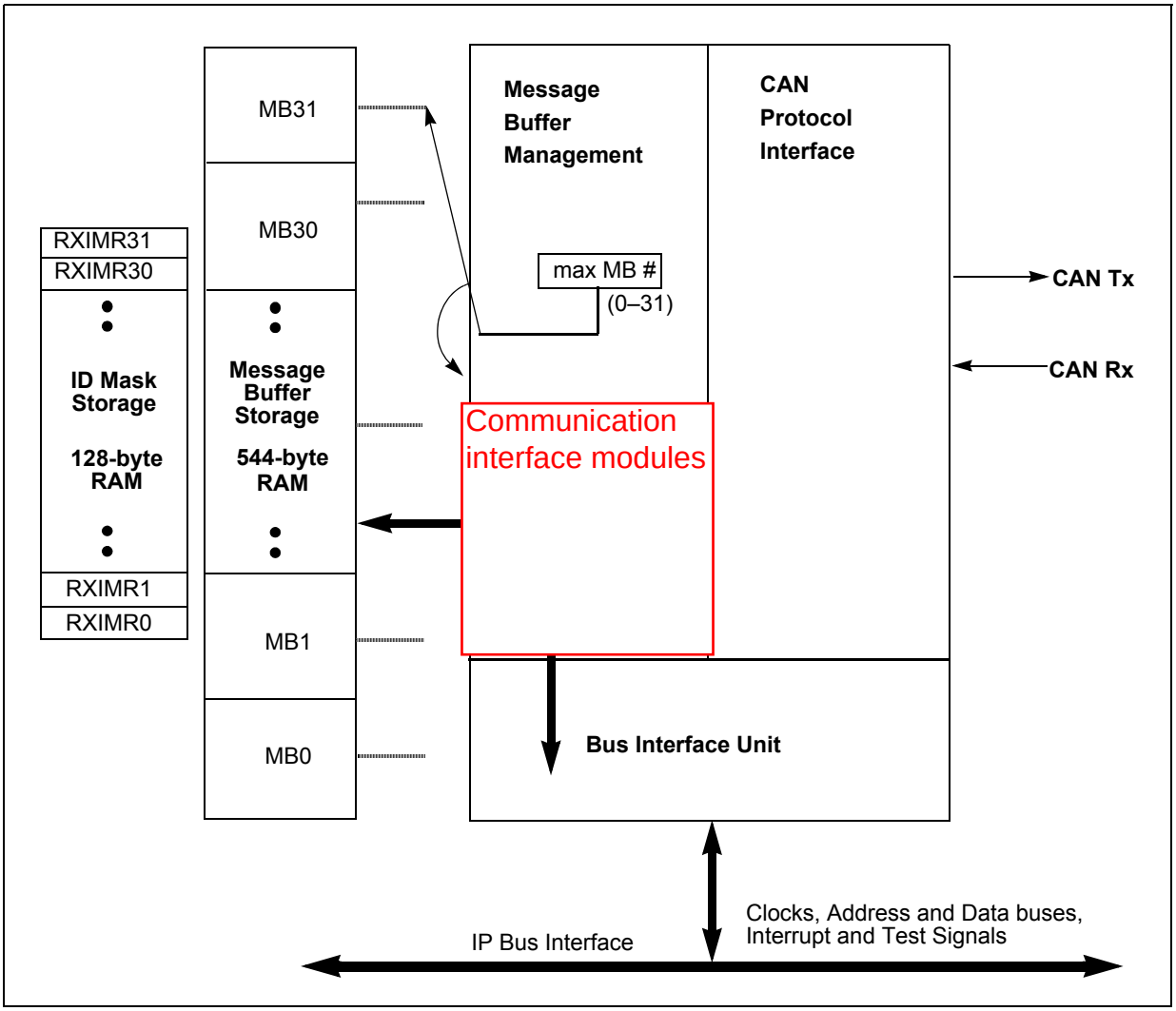
```
// Timer 1
PIT_LDVAL1 = 0x0003E7FF; // setup timer 1 for 256000 cycles
PIT_TCTRL1 = TIE; // enable Timer 1 interrupts
PIT_TCTRL1 |= TEN; // start Timer 1

// Timer 3
PIT_LDVAL3 = 0x0016E35F; // setup timer 3for 1500000 cycles
PIT_TCTRL3 |= TEN; // start Timer 3
```

40 FlexCAN

40.1 Introduction

The FlexCAN module is a communication controller implementing the CAN protocol according to the CAN 2.0B protocol specification. A general block diagram is shown in *Figure 397*, which describes the main subblocks implemented in the FlexCAN module, including two embedded memories, one for storing Message Buffers (MB) and another one for storing Rx Individual Mask Registers. Support for 32 MBs is provided. The functions of the submodules are described in subsequent sections.

Figure 397. FlexCAN block diagram



40.1.1 Overview

The CAN protocol was primarily, but not only, designed to be used as a vehicle serial data bus, meeting the specific requirements of this field: real-time processing, reliable operation in the EMI environment of a vehicle, cost-effectiveness and required bandwidth. The

FlexCAN module is a full implementation of the CAN protocol specification, Version 2.0 B, which supports both standard and extended message frames. 32 Message Buffers are supported. The Message Buffers are stored in an embedded RAM dedicated to the FlexCAN module.

The CAN Protocol Interface (CPI) submodule manages the serial communication on the CAN bus, requesting RAM access for receiving and transmitting message frames, validating received messages and performing error handling. The Message Buffer Management (MBM) submodule handles Message Buffer selection for reception and transmission, taking care of arbitration and ID matching algorithms. The Bus Interface Unit (BIU) submodule controls the access to and from the internal interface bus, in order to establish connection to the CPU and to other blocks. Clocks, address and data buses, interrupt outputs and test signals are accessed through the Bus Interface Unit.

Two FlexCAN modules are implemented on the SPC570Sx device: FlexCAN_0 and FlexCAN_1.

40.1.2 FlexCAN module features

The FlexCAN module includes these features:

- Full Implementation of the CAN protocol specification, Version 2.0B
  - Standard data and remote frames
  - Extended data and remote frames
  - 0 to 8 bytes data length
  - Programmable bit rate as high as 1 Mbit/s
  - Content-related addressing
- 32 Flexible Message Buffers (MBs) of 0 to 8 bytes data length
- Each MB configurable as Rx or Tx, all supporting standard and extended messages
- Individual Rx Mask Registers per Message Buffer
- Includes 544 bytes (32 MBs) of RAM used for MB storage
- Includes 128 bytes (32 MBs) of RAM used for individual Rx Mask Registers
- Full-featured Rx FIFO with storage capacity for 6 frames and internal pointer handling
- Powerful Rx FIFO ID filtering, capable of matching incoming IDs against either 8 extended, 16 standard, or 32 partial (8-bit) IDs, with individual masking capability
- Selectable backwards compatibility with previous FlexCAN version
- Programmable clock source to the CAN Protocol Interface, either bus clock or crystal oscillator
- Unused MB and Rx Mask Register space can be used as general purpose RAM space
- Listen-only mode capability
- Programmable loop-back mode supporting self-test operation
- Programmable transmission priority scheme: lowest ID, lowest buffer number or highest priority
- Time Stamp based on 16-bit free-running timer
- Global network time, synchronized by a specific message
- Maskable interrupts
- Independent of the transmission medium (an external transceiver is assumed)
- Short latency time due to an arbitration scheme for high-priority messages
- Low power modes

40.1.3 Modes of operation

The FlexCAN module has four functional modes: Normal mode (User and Supervisor), Freeze mode, Listen-Only Mode, and Loop-Back mode. There are also these low power modes: Disable mode and Stop mode.

- Normal mode (User or Supervisor)

  In Normal mode, the module operates receiving and/or transmitting message frames, errors are handled normally and all the CAN Protocol functions are enabled. User and Supervisor modes differ in the access to some restricted control registers.

- Freeze mode

  It is enabled when the FRZ bit in the Module Configuration Register (MCR) is asserted. If enabled, Freeze Mode is entered when the HALT bit in the MCR is set or when Debug Mode is requested at MCU level. In this mode, no transmission or reception of

frames is done and synchronicity to the CAN bus is lost. See *Section 40.4.9.1: Freeze mode* for more information.

- Listen-Only mode

  The module enters this mode when the LOM bit in the Control Register is asserted. In this mode, transmission is disabled, all error counters are frozen and the module operates in a CAN Error Passive mode . Only messages acknowledged by another CAN station will be received. If FlexCAN detects a message that has not been acknowledged, it will flag a BIT0 error (without changing the REC), as if it was trying to acknowledge the message.

- Loop-Back mode

  The module enters this mode when the LPB bit in the Control Register is asserted. In this mode, FlexCAN performs an internal loop back that can be used for self test operation. The bit stream output of the transmitter is internally feedback to the receiver input. The Rx CAN input pin is ignored and the Tx CAN output goes to the recessive state (logic 1). FlexCAN behaves as it normally does when transmitting and treats its own transmitted message as a message received from a remote node. In this mode, FlexCAN ignores the bit sent during the ACK slot in the CAN frame acknowledge field to ensure proper reception of its own message. Both transmit and receive interrupts are generated.

- Module Disable mode

  This low power mode is entered when the MDIS bit in the MCR Register is asserted by the CPU. When disabled, the module requests to disable the clocks to the CAN Protocol Interface and Message Buffer Management submodules. Exit from this mode is done by negating the MDIS bit in the MCR . See *Section 40.4.9.2: Module disable mode* for more information.

- Stop mode

  This low power mode is entered when Stop mode is requested at device level. When in Stop mode, the module puts itself in an inactive state and then informs the CPU that the clocks can be shut down globally. Exit from this mode happens when the Stop mode request is removed. See *Section 40.4.9.3: Stop mode* for more information.

40.2 External signal description

40.2.1 Overview

The FlexCAN module has two I/O signals connected to the external MCU pins. These signals are summarized in *Table 468* and described in more detail in the next subsections.

Table 468. FlexCAN signals

Signal name	Direction	Description
RXD	Input	CAN receive pin
TXD	Output	CAN transmit pin

40.2.2 Signal Descriptions

40.2.2.1 RXD

This pin is the receive pin from the CAN bus transceiver . Dominant state is represented by logic level 0 . Recessive state is represented by logic level 1.

40.2.2.2 TXD

This pin is the transmit pin to the CAN bus transceiver . Dominant state is represented by logic level 0 . Recessive state is represented by logic level 1.

40.3 Memory map and registers description

This section describes the registers and data structures in the FlexCAN module. The base address of the module depends on the particular memory map of the device. The addresses presented here are relative to the base address.

The address space occupied by FlexCAN has 96 bytes for registers starting at the module base address, followed by MB storage space in embedded RAM starting at address 0x0060, and an extra ID Mask storage space in a separate embedded RAM starting at address 0x0880.

40.3.1 FlexCAN memory mapping

The complete memory map for a FlexCAN module with 32 MBs capability is shown in *Table 469*. The access type can be Supervisor (S) or Unrestricted (U), also called User access. Most of the registers can be configured to have either Supervisor or Unrestricted access by programming the SUPV bit in the MCR.

The Rx Global Mask (RXGMASK), Rx Buffer 14 Mask (RX14MASK) and the Rx Buffer 15 Mask (RX15MASK) registers are provided for backwards compatibility, and are not used when the BCC bit in the MCR is asserted.

The address ranges 0x0060–0x027F and 0x0880–0x08FF are occupied by two separate embedded memories of RAM, of 544 bytes and 128 bytes, respectively. When the FlexCAN is configured with 32 MBs, the memory sizes are 544 and 128 bytes, so the address ranges 0x0280–0x047F and 0x0900–0x097F are considered reserved space. Furthermore, if the BCC bit in the MCR is negated, then the whole Rx Individual Mask Registers address range (0x0880–0x097F) is considered reserved space.

Table 469. FlexCAN module memory map

Offset from FlexCAN_BASE 0xFFEC_0000 (CAN_0) 0xFBEC_0000 (CAN_1)	Register	Access type	Location
0x0000	Module Configuration Register (MCR)	S	*Section 40.3.4.1*
0x0004	Control Register (CTRL)	S/U	*Section 40.3.4.2*
0x0008	Free Running Timer (TIMER)	S/U	*Section 40.3.4.3*
0x000C	Reserved		
0x0010	Rx Global Mask (RXGMASK)	S/U	*Section 40.3.4.4*

Table 469. FlexCAN module memory map(Continued)

Offset from FlexCAN_BASE 0xFFEC_0000 (CAN_0) 0xFBEC_0000 (CAN_1)	Register	Access type	Location
0x0014	Rx Buffer 14 Mask (RX14MASK)	S/U	*Section 40.3.4.5*
0x0018	Rx Buffer 15 Mask (RX15MASK)	S/U	*Section 40.3.4.6*
0x001C	Error Counter Register (ECR)	S/U	*Section 40.3.4.7*
0x0020	Error and Status Register (ESR)	S/U	*Section 40.3.4.8*
0x0024	Reserved		
0x0028	Interrupt Masks 1 (IMASK1)	S/U	*Section 40.3.4.9*
0x002C	Reserved		
0x0030	Interrupt Flags 1 (IFLAG1)	S/U	*Section 40.3.4.10*
0x0034–0x005F	Reserved		
0x0060–0x007F	Serial Message Buffers (SMB0–SMB1) – Reserved		
0x0080–0x027F	Message Buffers MB0–MB31	S/U	*Section 40.3.2*
0x0280–0x087F	Reserved		
0x0880–0x08FF	Rx Individual Mask Registers RXIMR0–RXIMR31	S/U	*Section 40.3.4.11*
0x0900–0x3FFF	Reserved		

Table 470. FlexCAN register reset status

Register	Affected by hard reset	Affected by soft reset
Module Configuration Register (MCR)	Yes	Yes
Control Register (CTRL)	Yes	No
Free Running Timer (TIMER)	Yes	Yes
Reserved		
Rx Global Mask (RXGMASK)	Yes	No
Rx Buffer 14 Mask (RX14MASK)	Yes	No
Rx Buffer 15 Mask (RX15MASK)	Yes	No
Error Counter Register (ECR)	Yes	Yes
Error and Status Register (ESR)	Yes	Yes
Interrupt Masks 1 (IMASK1)	Yes	Yes
Interrupt Flags 1 (IFLAG1)	Yes	Yes
Serial Message Buffers (SMB0–SMB1) – Reserved	No	No
Message Buffers MB0–MB31	No	No
Rx Individual Mask Registers RXIMR0–RXIMR31	No	No

The FlexCAN module stores CAN messages for transmission and reception using a Message Buffer structure. Each individual MB is formed by 16 bytes mapped on memory as described in *Table 471*. The module also implements two additional Message Buffers called SMB0 and SMB1 (Serial Message Buffers), in the address ranges 0x60–0x6F and 0x70–0x7F, which are not accessible to the end user. They are used for temporary storage of frames during the reception and transmission processes. *Table 471* shows a Standard/Extended Message Buffer (MB0) memory map, using 16 bytes total (0x80–0x8F space).

Table 471. Message Buffer MB0 memory mapping

Address offset	MB field
0x0080	Control and Status (C/S)
0x0084	Identifier Field
0x0088–0x008F	Data Field 0 to Data Field 7 (1 byte each)

40.3.2 Message buffer structure

The Message Buffer structure used by the FlexCAN module is represented in *Figure 398* . Both Extended and Standard Frames (29-bit Identifier and 11-bit Identifier, respectively) used in the CAN specification (Version 2.0 Part B) are represented.

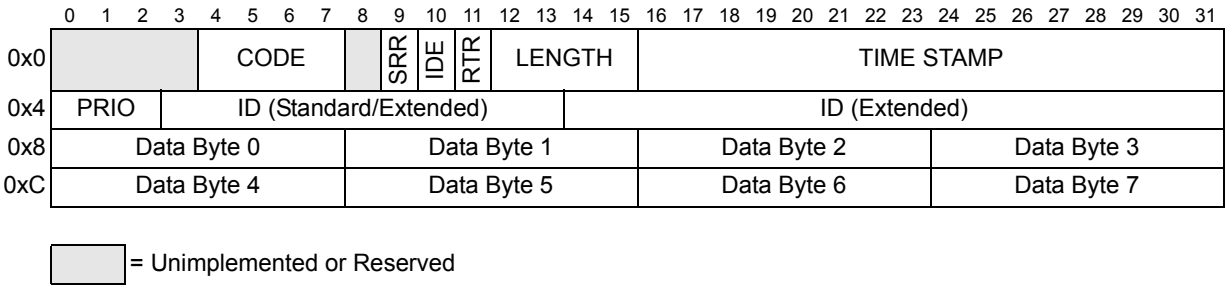


Figure 398. Message buffer structure

Table 472. Message Buffer structure field description

Field	Description
CODE	Message Buffer Code This 4-bit field can be accessed (read or write) by the CPU and by the Flexcan module itself, as part of the message buffer matching and arbitration process. The encoding is shown in *Table 473* and *Table 474*. See *Section 40.4* for additional information.
SRR	Substitute Remote Request Fixed recessive bit, used only in extended format. It must be set to ‘1’ by the user for transmission (Tx Buffers) and will be stored with the value received on the CAN bus for Rx receiving buffers. It can be received as either recessive or dominant. If FlexCAN receives this bit as dominant, then it is interpreted as arbitration loss. 0 Dominant is not a valid value for transmission in Extended Format frames 1 Recessive value is compulsory for transmission in Extended Format frames

Table 472. Message Buffer structure field description(Continued)

Field	Description
IDE	ID Extended Bit This bit identifies whether the frame format is standard or extended. 0 Frame format is standard 1 Frame format is extended
RTR	Remote Transmission Request This bit is used for requesting transmissions of a data frame. If FlexCAN transmits this bit as 1 (recessive) and receives it as 0 (dominant), it is interpreted as arbitration loss. If this bit is transmitted as 0 (dominant), then if it is received as 1 (recessive), the FlexCAN module treats it as bit error. If the value received matches the value transmitted, it is considered as a successful bit transmission. 0 Indicates the current MB has a Data Frame to be transmitted 1 Indicates the current MB has a Remote Frame to be transmitted
LENGTH	Length of Data in Bytes This 4-bit field is the length (in bytes) of the Rx or Tx data, which is located in offset 0x8 through 0xF of the MB space (see *Figure 398*). In reception, this field is written by the FlexCAN module, copied from the DLC (Data Length Code) field of the received frame. In transmission, this field is written by the CPU and corresponds to the DLC field value of the frame to be transmitted. When RTR=1, the Frame to be transmitted is a Remote Frame and does not include the data field, regardless of the Length field.
TIME STAMP	Free-Running Counter Time Stamp This 16-bit field is a copy of the Free-Running Timer, captured for Tx and Rx frames at the time when the beginning of the Identifier field appears on the CAN bus.
PRIO	Local priority This 3-bit field is only used when LPRIO_EN bit is set in MCR and it only makes sense for Tx buffers. These bits are not transmitted. They are appended to the regular ID to define the transmission priority. See *Section 40.4.3.*
ID	Frame Identifier In Standard Frame format, only the 11 most significant bits (3 to 13) are used for frame identification in both receive and transmit cases. The 18 least significant bits are ignored. In Extended Frame format, all bits are used for frame identification in both receive and transmit cases.
DATA	Data Field As many as 8 bytes can be used for a data frame. For Rx frames, the data is stored as it is received from the CAN bus. For Tx frames, the CPU prepares the data field to be transmitted within the frame.

Table 473. Message buffer code for Rx buffers

Rx code BEFORE Rx new frame	Description	Rx code AFTER Rx new frame	Comment
0000	INACTIVE: MB is not active .	–	MB does not participate in the matching process.
0100	EMPTY: MB is active and empty .	0010	MB participates in the matching process. When a frame is received successfully, the code is automatically updated to FULL.
0010	FULL: MB is full.	0010	The act of reading the C/S word followed by unlocking the MB does not make the code return to EMPTY. It remains FULL. If a new frame is written to the MB after the C/S word was read and the MB was unlocked, the code still remains FULL.
		0110	If the MB is FULL and a new frame is overwritten to this MB before the CPU had time to read it, the code is automatically updated to OVERRUN. Refer to *Section 40.4.5* for details about overrun behavior.
0110	OVERRUN: a frame was overwritten into a full buffer .	0010	If the code indicates OVERRUN but the CPU reads the C/S word and then unlocks the MB, when a new frame is written to the MB the code returns to FULL.
		0110	If the code already indicates OVERRUN, and yet another new frame must be written, the MB will be overwritten again, and the code will remain OVERRUN. Refer to *Section 40.4.5* for details about overrun behavior.
0XY1(1)	BUSY: Flexcan is updating the contents of the MB. The CPU must not access the MB .	0010	An EMPTY buffer was written with a new frame (XY was 01).
		0110	A FULL/OVERRUN buffer was overwritten (XY was 11).

1. Note that for Tx MBs (see *Table 474*), the BUSY bit should be ignored upon read, except when AEN bit is set in the MCR.

Table 474. Message Buffer code for Tx buffers

RTR	Initial Tx code	Code after successful transmission	Description
X	1000	–	INACTIVE: MB does not participate in the arbitration process.
X	1001	–	ABORT: MB was configured as Tx and CPU aborted the transmission. This code is only valid when AEN bit in MCR is asserted. MB does not participate in the arbitration process.
0	1100	1000	Transmit data frame unconditionally once. After transmission, the MB automatically returns to the INACTIVE state.
1	1100	0100	Transmit remote frame unconditionally once. After transmission, the MB automatically becomes an Rx MB with the same ID.
0	1010	1010	Transmit a data frame whenever a remote request frame with the same ID is received. This MB participates simultaneously in both the matching and arbitration processes. The matching process compares the ID of the incoming remote request frame with the ID of the MB. If a match occurs this MB is allowed to participate in the current arbitration process and the Code field is automatically updated to ‘1110’ to allow the MB to participate in future arbitration runs. When the frame is eventually transmitted successfully, the Code automatically returns to ‘1010’ to restart the process again.
0	1110	1010	This is an intermediate code that is automatically written to the MB by the MBM as a result of match to a remote request frame. The data frame will be transmitted unconditionally once and then the code will automatically return to ‘1010’. The CPU can also write this code with the same effect.

Table 475. MB0–MB31 addresses

Address	Register	Address	Register
Base + 0x0080	MB0	Base + 0x0180	MB16
Base + 0x0090	MB1	Base + 0x0190	MB17
Base + 0x00A0	MB2	Base + 0x01A0	MB18
Base + 0x00B0	MB3	Base + 0x01B0	MB19
Base + 0x00C0	MB4	Base + 0x01C0	MB20
Base + 0x00D0	MB5	Base + 0x01D0	MB21
Base + 0x00E0	MB6	Base + 0x01E0	MB22
Base + 0x00F0	MB7	Base + 0x01F0	MB23
Base + 0x0100	MB8	Base + 0x0200	MB24
Base + 0x0110	MB9	Base + 0x0210	MB25
Base + 0x0120	MB10	Base + 0x0220	MB26
Base + 0x0130	MB11	Base + 0x0230	MB27
Base + 0x0140	MB12	Base + 0x0240	MB28

Table 475. MB0–MB31 addresses(Continued)

Address	Register	Address	Register
Base + 0x0150	MB13	Base + 0x0250	MB29
Base + 0x0160	MB14	Base + 0x0260	MB30
Base + 0x0170	MB15	Base + 0x0270	MB31

40.3.3 Rx FIFO structure

When the FEN bit is set in the MCR, the memory area from 0x80 to 0xFF (which is normally occupied by MBs 0 to 7) is used by the reception FIFO engine. *Figure 399* shows the Rx FIFO data structure. The region 0x80–0x8F contains an MB structure that is the port through which the CPU reads data from the FIFO (the oldest frame received and not read yet). The region 0x90–0xDF is reserved for internal use of the FIFO engine. The region 0xE0–0xFF contains an 8-entry ID table that specifies filtering criteria for accepting frames into the FIFO. *Table 400* shows the three different formats that the elements of the ID table can assume, depending on the IDAM field of the MCR. Note that all elements of the table must have the same format. See *Section 40.4.7* for more information.

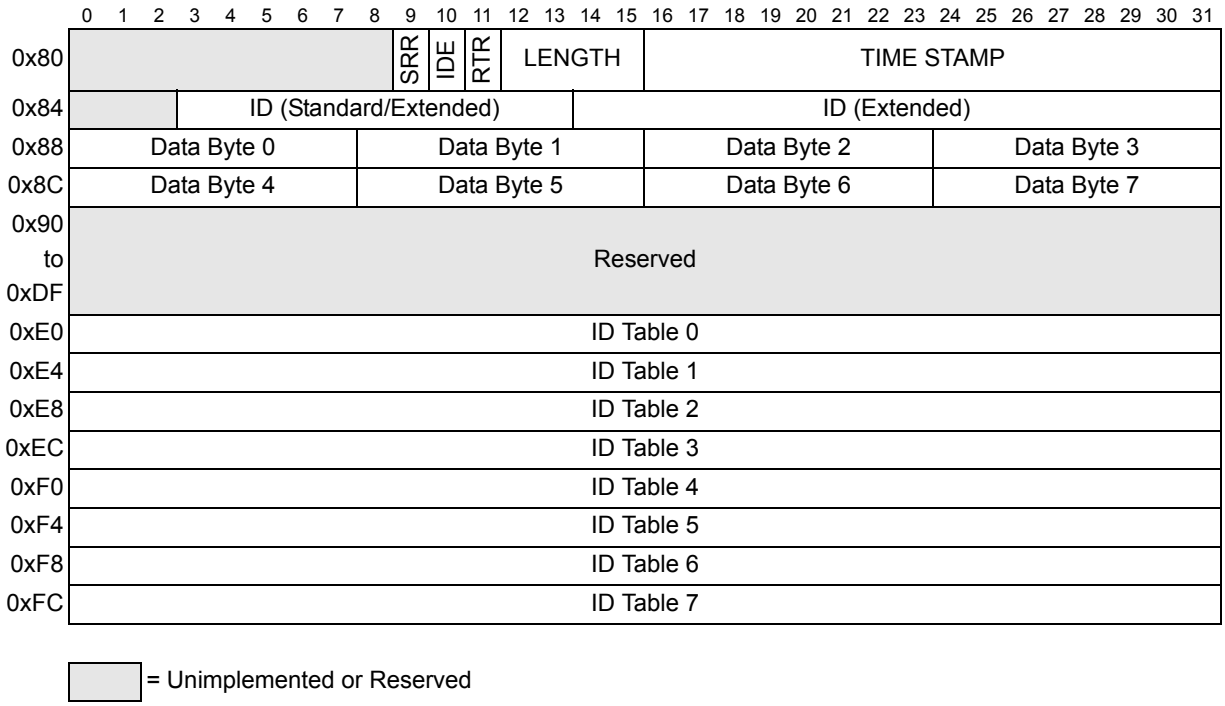


Figure 399. Rx FIFO structure

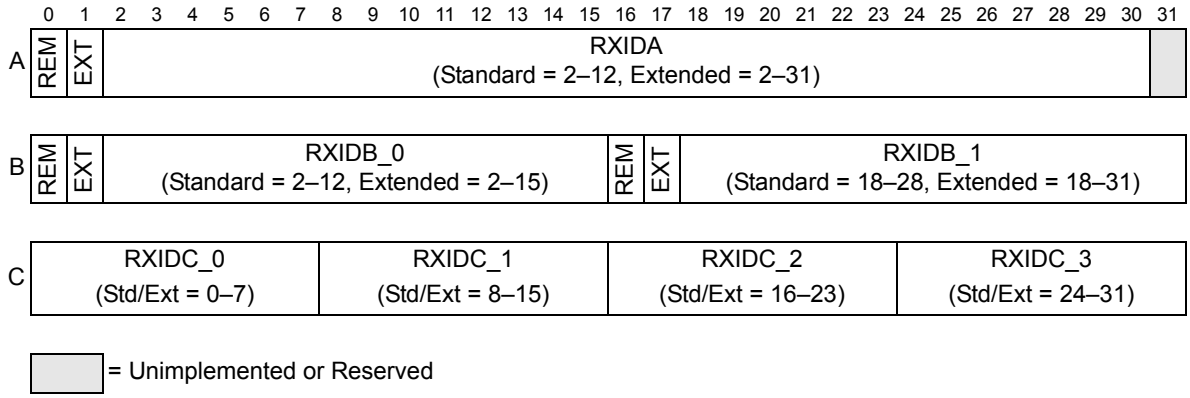


Figure 400. ID Table 0–7

Table 476. Rx FIFO Structure field description

Field	Description
REM	Remote Frame This bit specifies if Remote Frames are accepted into the FIFO if they match the target ID. 0 Remote Frames are rejected and data frames can be accepted 1 Remote Frames can be accepted and data frames are rejected
EXT	Extended Frame Specifies whether extended or standard frames are accepted into the FIFO if they match the target ID. 0 Extended frames are rejected and standard frames can be accepted 1 Extended frames can be accepted and standard frames are rejected
RXIDA	Rx Frame Identifier (Format A) Specifies an ID to be used as acceptance criteria for the FIFO. In the standard frame format, only the 11 most significant bits (3 to 13) are used for frame identification. In the extended frame format, all bits are used.
RXIDB_0 RXIDB_1	Rx Frame Identifier (Format B) Specifies an ID to be used as acceptance criteria for the FIFO. In the standard frame format, the 11 most significant bits (a full standard ID) (3 to 13) are used for frame identification. In the extended frame format, all 14 bits of the field are compared to the 14 most significant bits of the received ID.
RXIDC_0 RXIDC_1 RXIDC_2 RXIDC_3	Rx Frame Identifier (Format C) Specifies an ID to be used as acceptance criteria for the FIFO. In both standard and extended frame formats, all 8 bits of the field are compared to the 8 most significant bits of the received ID.

40.3.4 Registers description

The FlexCAN registers are described in this section in ascending address order.

40.3.4.1 Module Configuration Register (MCR)

This register defines global system configurations, such as the module operation mode (for example low power) and maximum message buffer configuration. Most of the fields in this

register can be accessed at any time, except the MAXMB field, which should only be changed while the module is in Freeze Mode.

Address: 0x0000 Access: Supervisor read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	MDIS	FRZ	FEN	HALT	NOT_RDY	0(1)	SOFT_RST	FRZ_ACK	SUPV	0(1)	WRN_EN	LPM_ACK	0(1)	0(1)	SRX_DIS	BCC
W																
Reset	1	1	0	1	1	0	0	0	1	0	0	1	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	LPRIO_EN	AEN	0	0	IDAM		0	0	MAXMB					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1

1. This reserved bit has no impact on the device and read value is always the last one written.

Figure 401. Module Configuration Register (MCR)

Table 477. MCR field descriptions

Field	Description
0 MDIS	Module Disable This bit controls whether FlexCAN is enabled or not. When disabled, FlexCAN shuts down the clocks to the CAN Protocol Interface and Message Buffer Management submodules. This is the only bit in MCR not affected by soft reset. See *Section 40.4.9.2: Module disable mode* for more information. 0 Enable the FlexCAN module 1 Disable the FlexCAN module
1 FRZ	Freeze Enable The FRZ bit specifies the FlexCAN behavior when the HALT bit in the MCR is set or when Debug Mode is requested at MCU level. When FRZ is asserted, FlexCAN is able to enter Freeze Mode. Negation of this bit field causes FlexCAN to exit from Freeze Mode. 0 Not able to enter Freeze Mode 1 Able to enter Freeze Mode
2 FEN	FIFO Enable This bit controls whether the FIFO feature is enabled or not. When FEN is set, MBs 0 to 7 cannot be used for normal reception and transmission because the corresponding memory region (0x80-0xFF) is used by the FIFO engine. See *Section 40.3.3* and *Section 40.4.7* for more information. 0 FIFO disabled 1 FIFO enabled

Table 477. MCR field descriptions(Continued)

Field	Description
3 HALT	Halt FlexCAN Assertion of this bit puts the FlexCAN module into Freeze Mode. The CPU should clear it after initializing the Message Buffers and Control Register. No reception or transmission is performed by FlexCAN before this bit is cleared. While in Freeze Mode, the CPU has write access to the Error Counter Register, that is otherwise read-only. Freeze Mode cannot be entered while FlexCAN is in any of the low power modes. See *Section 40.4.9.1: Freeze mode* for more information. 0 No Freeze Mode request 1 Enters Freeze Mode if the FRZ bit is asserted
4 NOT_RDY	FlexCAN Not Ready This read-only bit indicates that FlexCAN is either in Disable Mode, Stop Mode or Freeze Mode . It is negated once FlexCAN has exited these modes. 0 FlexCAN module is either in Normal Mode, Listen-Only Mode or Loop-Back Mode 1 FlexCAN module is either in Disable Mode, Stop Mode or Freeze Mode
6 SOFT_RST	Soft Reset When this bit is asserted, FlexCAN resets its internal state machines and some of the memory mapped registers . The following registers are reset: MCR (except the MDIS bit), TIMER, ECR, ESR, IMASK1, IFLAG1. Configuration registers that control the interface to the CAN bus are not affected by soft reset. The following registers are unaffected: – CTRL – RXIMR0–RXIMR31 – RXGMASK, RX14MASK, RX15MASK – all Message Buffers The SOFT_RST bit can be asserted directly by the CPU when it writes to the MCR. Since soft reset is synchronous and has to follow a request/acknowledge procedure across clock domains, it may take some time to fully propagate its effect . The SOFT_RST bit remains asserted while reset is pending, and is automatically negated when reset completes . Therefore, software can poll this bit to know when the soft reset has completed. Soft reset cannot be applied while clocks are shut down in any of the low power modes. The module should be first removed from low power mode, and then soft reset can be applied . 0 No reset request 1 Resets the registers marked as “affected by soft reset” in *Table 469*
7 FRZ_ACK	Freeze Mode Acknowledge This read-only bit indicates that FlexCAN is in Freeze mode and its prescaler is stopped. The Freeze mode request cannot be granted until current transmission or reception processes have finished . Therefore the software can poll the FRZ_ACK bit to know when FlexCAN has actually entered Freeze Mode . If Freeze mode request is negated, then this bit is negated once the FlexCAN prescaler is running again. If Freeze mode is requested while FlexCAN is in any of the low power modes, then the FRZ_ACK bit will only be set when the low power mode is exited. See *Section 40.4.9.1: Freeze mode* for more information. 0 FlexCAN not in Freeze mode, prescaler running 1 FlexCAN in Freeze mode, prescaler stopped

Table 477. MCR field descriptions(Continued)

Field	Description
8 SUPV	Supervisor Mode This bit configures some of the FlexCAN registers to be either in Supervisor or Unrestricted memory space . The registers affected by this bit are marked as S/U in the Access Type column of *Table 469*. The reset value of this bit is 1, so the affected registers start with Supervisor access restrictions. 0 Affected registers are in Unrestricted memory space 1 Affected registers are in Supervisor memory space. Any access without supervisor permission behaves as though the access was done to an unimplemented register location
10 WRN_EN	Warning Interrupt Enable When asserted, this bit enables the generation of the TWRN_INT and RWRN_INT flags in the Error and Status Register. If WRN_EN is negated, the TWRN_INT and RWRN_INT flags will always be zero, independent of the values of the error counters, and no warning interrupt will ever be generated. 0 TWRN_INT and RWRN_INT bits are zero, independent of the values in the error counters 1 TWRN_INT and RWRN_INT bits are set when the respective error counter transition from < 96 to >= 96
11 LPM_ACK	Low Power Mode Acknowledge This read-only bit indicates that FlexCAN is either in Disable Mode or Stop Mode . Either of these low power modes cannot be entered until all current transmission or reception processes have finished, so the CPU can poll the LPM_ACK bit to know when FlexCAN has actually entered low power mode. See *Section 40.4.9.2: Module disable mode* and *Section 40.4.9.3: Stop mode* for more information. 0 FlexCAN not in any of the low power modes 1 FlexCAN is either in Disable Mode or Stop mode
14 SRX_DIS	Self Reception Disable This bit defines whether FlexCAN is allowed to receive frames transmitted by itself. If this bit is asserted, frames transmitted by the module will not be stored in any MB, regardless if the MB is programmed with an ID that matches the transmitted frame, and no interrupt flag or interrupt signal will be generated due to the frame reception. 0 Self reception enabled 1 Self reception disabled
15 BCC	Backwards Compatibility Configuration This bit is provided to support Backwards Compatibility with previous FlexCAN versions. When this bit is negated, the following configuration is applied: – For MCUs supporting individual Rx ID masking, this feature is disabled. Instead of individual ID masking per MB, FlexCAN uses its previous masking scheme with RXGMASK, RX14MASK and RX15MASK. – The reception queue feature is disabled. Upon receiving a message, if the first MB with a matching ID that is found is still occupied by a previous unread message, FlexCAN will not look for another matching MB. It will override this MB with the new message and set the CODE field to ‘0110’ (overrun). Upon reset this bit is negated, allowing legacy software to work without modification. 0 Individual Rx masking and queue feature are disabled 1 Individual Rx masking and queue feature are enabled

Table 477. MCR field descriptions(Continued)

Field	Description
18 LPRIO_EN	Local Priority Enable This bit is provided for backwards compatibility reasons. It controls whether the local priority feature is enabled or not. It extends the ID used during the arbitration process. With this extended ID concept, the arbitration process is done based on the full 32-bit word, but the actual transmitted ID still has 11-bit for standard frames and 29-bit for extended frames. 0 Local Priority disabled 1 Local Priority enabled
19 AEN	Abort Enable This bit is supplied for backwards compatibility reasons. When asserted, it enables the Tx abort feature. This feature guarantees a safe procedure for aborting a pending transmission, so that no frame is sent in the CAN bus without notification. 0 Abort disabled 1 Abort enabled
22:23 IDAM	ID Acceptance Mode This 2-bit field identifies the format of the elements of the Rx FIFO filter table, as shown in *Table 478*.. **Note:** All elements of the table are configured at the same time by this field (they are all the same format). See *Section 40.3.3*..
26:31 MAXMB	Maximum Number of Message Buffers This 6-bit field defines the maximum number of message buffers that will take part in the matching and arbitration processes. The reset value (0x0F) is equivalent to 16 MB configuration. This field should be changed only while the module is in Freeze Mode. Maximum MBs in use = MAXMB + 1 **Note:** MAXMB has to be programmed with a value smaller or equal to the number of available Message Buffers, otherwise FlexCAN will not transmit or receive frames.

Table 478. IDAM coding

IDAM	Format	Explanation
00	A	One full ID (standard or extended) per filter element
01	B	Two full standard IDs or two partial 14-bit extended IDs per filter element
10	C	Four partial 8-bit IDs (standard or extended) per filter element
11	D	All frames rejected

40.3.4.2 Control Register (CTRL)

This register is defined for specific FlexCAN control features related to the CAN bus, such as bit-rate, programmable sampling point within an Rx bit, Loop Back Mode, Listen Only Mode, Bus Off recovery behavior and interrupt enabling (Bus-Off, Error, Warning). It also determines the Division Factor for the clock prescaler. Most of the fields in this register should only be changed while the module is in Disable Mode or in Freeze Mode. Exceptions are the BOFF_MSK, ERR_MSK, TWRN_MSK, RWRN_MSK and BOFF_REC bits, that can be accessed at any time.

Address: 0x0004 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	PRESDIV								RJW		PSEG1			PSEG2		
Reset	0	0	0	0	0	0	0	0	0	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	BOFF_MSK	ERR_MSK	CLK_SRC	LPB	TWRN_MSK	RWRN_MSK	0	0	SMP	BOFF_REC	TSYN	LBUF	LOM	PROPSEG		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 402. Control Register (CTRL)

Table 479. CTRL field descriptions

Field	Description
0:7 PRESDIV	Prescaler Division Factor This 8-bit field defines the ratio between the CPI clock frequency and the Serial Clock (Sclock) frequency . The Sclock period defines the time quantum of the CAN protocol. For the reset value, the Sclock frequency is equal to the CPI clock frequency. The Maximum value of this register is 0xFF, that gives a minimum Sclock frequency equal to the CPI clock frequency divided by 256. For more information refer to *Section 40.4.8.4: Protocol timing.* Sclock frequency = CPI clock frequency / (PRESDIV + 1)
8:9 RJW	Resync Jump Width This 2-bit field defines the maximum number of time quanta(1) that a bit time can be changed by one resynchronization. The valid programmable values are 0–3. Resync Jump Width = RJW + 1.
10:12 PSEG1	PSEG1 — Phase Segment 1 This 3-bit field defines the length of Phase Buffer Segment 1 in the bit time. The valid programmable values are 0–7. Phase Buffer Segment 1(PSEG1 + 1) x Time-Quanta.
13:15 PSEG2	PSEG2: Phase Segment 2 This 3-bit field defines the length of Phase Buffer Segment 2 in the bit time. The valid programmable values are 1–7. Phase Buffer Segment 2 = (PSEG2 + 1) x Time-Quanta.
16 BOFF_MSK	Bus Off Mask This bit provides a mask for the Bus Off Interrupt. 0 Bus Off interrupt disabled. 1 Bus Off interrupt enabled.
17 ERR_MSK	Error Mask This bit provides a mask for the Error Interrupt. 0 Error interrupt disabled. 1 Error interrupt enabled.

Table 479. CTRL field descriptions(Continued)

Field	Description
18 CLK_SRC	CAN Engine Clock Source This bit selects the clock source to the CAN Protocol Interface (CPI) to be either the peripheral clock (driven by the PLL) or the crystal oscillator clock . The selected clock is the one fed to the prescaler to generate the Serial Clock (Sclock) . In order to guarantee reliable operation, this bit should only be changed while the module is in Disable Mode. See *Section 40.4.8.4: Protocol timing* for more information. 0 The CAN engine clock source is the oscillator clock 1 The CAN engine clock source is the bus clock **Note:** This bit does not have any effect as clock source selection (PLL or oscillator) is managed by AUX CLK SEL1 in the Clock Generation Module (CGM) **Note:**
19 LPB	Loop Back This bit configures FlexCAN to operate in Loop-Back Mode . In this mode, FlexCAN performs an internal loop back that can be used for self test operation . The bit stream output of the transmitter is fed back internally to the receiver input . The Rx CAN input pin is ignored and the Tx CAN output goes to the recessive state (logic 1). FlexCAN behaves as it normally does when transmitting, and treats its own transmitted message as a message received from a remote node. In this mode, FlexCAN ignores the bit sent during the ACK slot in the CAN frame acknowledge field, generating an internal acknowledge bit to ensure proper reception of its own message. Both transmit and receive interrupts are generated. 0 Loop Back disabled 1 Loop Back enabled
20 TWRN_MSK	Tx Warning Interrupt Mask This bit provides a mask for the Tx Warning Interrupt associated with the TWRN_INT flag in the Error and Status Register. This bit has no effect if the WRN_EN bit in MCR is negated and it is read as zero when WRN_EN is negated. 0 Tx Warning Interrupt disabled. 1 Tx Warning Interrupt enabled.
21 RWRN_MSK	Rx Warning Interrupt Mask This bit provides a mask for the Rx Warning Interrupt associated with the RWRN_INT flag in the Error and Status Register. This bit has no effect if the WRN_EN bit in MCR is negated and it is read as zero when WRN_EN is negated. 0 Rx Warning Interrupt disabled. 1 Rx Warning Interrupt enabled.
24 SMP	Sampling Mode This bit defines the sampling mode of CAN bits at the Rx input . 0 Just one sample determines the bit value. 1 Three samples are used to determine the value of the received bit: the regular one (sample point) and two preceding samples, a majority rule is used.

Table 479. CTRL field descriptions(Continued)

Field	Description
25 BOFF_REC	Bus Off Recovery Mode This bit defines how FlexCAN recovers from Bus Off state. If this bit is negated, automatic recovering from Bus Off state occurs according to the CAN Specification 2.0B . If the bit is asserted, automatic recovering from Bus Off is disabled and the module remains in Bus Off state until the bit is negated by the user. If the negation occurs before 128 sequences of 11 recessive bits are detected on the CAN bus, then Bus Off recovery happens as if the BOFF_REC bit had never been asserted. If the negation occurs after 128 sequences of 11 recessive bits occurred, then FlexCAN will resynchronize to the bus by waiting for 11 recessive bits before joining the bus. After negation, the BOFF_REC bit can be re-asserted again during Bus Off, but it will only be effective the next time the module enters Bus Off. If BOFF_REC was negated when the module entered Bus Off, asserting it during Bus Off will not be effective for the current Bus Off recovery. 0 Automatic recovering from Bus Off state enabled, according to CAN Spec 2.0 part B. 1 Automatic recovering from Bus Off state disabled.
26 TSYN	Timer Sync Mode This bit enables a mechanism that resets the free-running timer each time a message is received in Message Buffer 0. This feature provides means to synchronize multiple FlexCAN stations with a special “SYNC” message (that is, global network time). If the FEN bit in MCR is set (FIFO enabled), MB8 is used for timer synchronization instead of MB0. 0 Timer Sync feature disabled 1 Timer Sync feature enabled
27 LBUF	Lowest Buffer Transmitted First This bit defines the ordering mechanism for Message Buffer transmission. When asserted, the LPRIO_EN bit does not affect the priority arbitration. 0 Buffer with highest priority is transmitted first. 1 Lowest number buffer is transmitted first.
28 LOM	Listen-Only Mode This bit configures FlexCAN to operate in Listen Only Mode. In this mode, transmission is disabled, all error counters are frozen and the module operates in a CAN Error Passive mode. Only messages acknowledged by another CAN station will be received. If FlexCAN detects a message that has not been acknowledged, it will flag a BIT0 error (without changing the REC), as if it was trying to acknowledge the message. 0 Listen Only Mode is deactivated 1 FlexCAN module operates in Listen Only Mode
29:31 PROPSEG	Propagation Segment This 3-bit field defines the length of the Propagation Segment in the bit time. The valid programmable values are 0–7. Propagation Segment Time = (PROPSEG + 1) × Time-Quanta. Time-Quantum = one Sclock period.

1. One time quantum is equal to the Sclock period.

40.3.4.3 Free Running Timer (TIMER)

This register represents a 16-bit free running counter that can be read and written by the CPU. The timer starts from 0x0000 after Reset, counts linearly to 0xFFFF, and wraps around.

The timer is clocked by the FlexCAN bit-clock (which defines the baud rate on the CAN bus). During a message transmission/reception, it increments by one for each bit that is received or transmitted. When there is no message on the bus, it counts using the previously programmed baud rate. During Freeze Mode, the timer is not incremented.

The timer value is captured at the beginning of the identifier field of any frame on the CAN bus. This captured value is written into the Time Stamp entry in a message buffer after a successful reception or transmission of a message.

Writing to the timer is an indirect operation. The data is first written to an auxiliary register and then an internal request/acknowledge procedure across clock domains is executed. All this is transparent to the user, except for the fact that the data will take some time to be actually written to the register. If desired, software can poll the register to discover when the data was actually written.

Address: 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	TIMER															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 403. Free Running Timer (TIMER)

Table 480. TIMER field descriptions

Field	Description
16:31 TIMER	Holds the value for this timer

40.3.4.4 Rx Global Mask register (RXGMASK)

This register is provided for legacy support and for low cost MCUs that do not have the individual masking per Message Buffer feature. For MCUs supporting individual masks per MB, setting the BCC bit in the MCR causes the RXGMASK register to have no effect on the module operation. For MCUs not supporting individual masks per MB, this register is always effective.

RXGMASK is used as an acceptance mask for all Rx MBs, excluding MBs 14–15, which have individual mask registers. When the FEN bit in the MCR is set (FIFO enabled), the RXGMASK also applies to all elements of the ID filter table, except elements 67, which have individual masks.

The contents of this register must be programmed while the module is in Freeze Mode, and must not be modified when the module is transmitting or receiving frames.

Address: 0x0010 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	MI31	MI30	MI29	MI28	MI27	MI26	MI25	MI24	MI23	MI22	MI21	MI20	MI19	MI18	MI17	MI16
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	MI15	MI14	MI13	MI12	MI11	MI10	MI9	MI8	MI7	MI6	MI5	MI4	MI3	MI2	MI1	MI0
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Figure 404. Rx Global Mask register (RXGMASK)

Table 481. RXGMASK field description

Field	Description
0:31 MI[31:0]	Mask Bits For normal Rx MBs, the mask bits affect the ID filter programmed on the MB. For the Rx FIFO, the mask bits affect all bits programmed in the filter table (ID, IDE, RTR) . 0 The corresponding bit in the filter is “don’t care” 1 The corresponding bit in the filter is checked against the one received

40.3.4.5 Rx 14 Mask (RX14MASK)

This register is provided for legacy support and for low cost MCUs that do not have the individual masking per Message Buffer feature. For MCUs supporting individual masks per MB, setting the BCC bit in the MCR causes the RX14MASK register to have no effect on the module operation.

RX14MASK is used as an acceptance mask for the Identifier in Message Buffer 14. When the FEN bit in the MCR is set (FIFO enabled), the RX14MASK also applies to element 6 of the ID filter table. This register has the same structure as the Rx Global Mask Register. It must be programmed while the module is in Freeze Mode, and must not be modified when the module is transmitting or receiving frames.

Address: Base + 0x0014 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	MI31	MI30	MI29	MI28	MI27	MI26	MI25	MI24	MI23	MI22	MI21	MI20	MI19	MI18	MI17	MI16
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	MI15	MI14	MI13	MI12	MI11	MI10	MI9	MI8	MI7	MI6	MI5	MI4	MI3	MI2	MI1	MI0
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Figure 405. Rx Buffer 14 Mask register (RX14MASK)

Table 482. RX14MASK field description

Field	Description
0:31 MI[31:0]	Mask Bits For normal Rx MBs, the mask bits affect the ID filter programmed on the MB. For the Rx FIFO, the mask bits affect all bits programmed in the filter table (ID, IDE, RTR) . 0 The corresponding bit in the filter is “don’t care” 1 The corresponding bit in the filter is checked against the one received

40.3.4.6 Rx 15 Mask (RX15MASK)

This register is provided for legacy support and for low cost MCUs that do not have the individual masking per Message Buffer feature. For MCUs supporting individual masks per MB, setting the BCC bit in the MCR causes the RX15MASK register to have no effect on the module operation .

When the BCC bit is negated, RX15MASK is used as acceptance mask for the Identifier in Message Buffer 15. When the FEN bit in the MCR is set (FIFO enabled), the RX15MASK also applies to element 7 of the ID filter table. This register has the same structure as the Rx Global Mask Register. It must be programmed while the module is in Freeze Mode, and must not be modified when the module is transmitting or receiving frames.

Address: 0x0018 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	MI31	MI30	MI29	MI28	MI27	MI26	MI25	MI24	MI23	MI22	MI21	MI20	MI19	MI18	MI17	MI16
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	MI15	MI14	MI13	MI12	MI11	MI10	MI9	MI8	MI7	MI6	MI5	MI4	MI3	MI2	MI1	MI0
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Figure 406. Rx Buffer 15 Mask register (RX15MASK)

Table 483. RX15MASK field description

Field	Description
0:31 MI[31:0]	Mask Bits For normal Rx MBs, the mask bits affect the ID filter programmed on the MB. For the Rx FIFO, the mask bits affect all bits programmed in the filter table (ID, IDE, RTR). 0 The corresponding bit in the filter is “don’t care” 1 The corresponding bit in the filter is checked against the one received

40.3.4.7 Error Counter Register (ECR)

This register has two 8-bit fields that reflect the value of two FlexCAN error counters:

- Transmit Error Counter (Tx_Err_Counter field)
- Receive Error Counter (Rx_Err_Counter field)

The rules for increasing and decreasing these counters are described in the CAN protocol and are completely implemented in the FlexCAN module. Both counters are read only except in Freeze Mode, where they can be written by the CPU.

Writing to the Error Counter Register while in Freeze Mode is an indirect operation . The data is first written to an auxiliary register and then an internal request/acknowledge procedure across clock domains is executed . All this is transparent to the user, except for the fact that the data will take some time to be actually written to the register . If desired, software can poll the register to discover when the data was actually written.

FlexCAN responds to any bus state as described in the protocol , e.g., transmit ‘Error Active’ or ‘Error Passive’ flag, delay its transmission start time (‘Error Passive’) and avoid any influence on the bus when in ‘Bus Off’ state . The following are the basic rules for FlexCAN bus state transitions.

- If the value of Tx_Err_Counter or Rx_Err_Counter increases to be greater than or equal to 128, the FLT_CONF field in the Error and Status Register is updated to reflect ‘Error Passive’ state.
- If the FlexCAN state is ‘Error Passive’, and either Tx_Err_Counter or Rx_Err_Counter decrements to a value less than or equal to 127 while the other already satisfies this condition, the FLT_CONF field in the Error and Status Register is updated to reflect ‘Error Active’ state.
- If the value of Tx_Err_Counter increases to be greater than 255, the FLT_CONF field in the Error and Status Register is updated to reflect ‘Bus Off’ state, and an interrupt may be issued. The value of Tx_Err_Counter is then reset to 0.
- If FlexCAN is in ‘Bus Off’ state, then Tx_Err_Counter is cascaded together with another internal counter to count the 128th occurrences of 11 consecutive recessive bits on the bus. Hence, Tx_Err_Counter is reset to 0 and counts in a manner where the internal counter counts 11 such bits and then wraps around while incrementing the Tx_Err_Counter. When Tx_Err_Counter reaches the value of 128, the FLT_CONF field in the Error and Status Register is updated to be ‘Error Active’ and both error counters are reset to ‘0’ . At any instance of dominant bit following a stream of less than 11 consecutive recessive bits, the internal counter resets itself to ‘0’ without affecting the Tx_Err_Counter value.
- If during system start-up, only one node is operating, then its Tx_Err_Counter increases in each message it is trying to transmit, as a result of acknowledge errors (indicated by the ACK_ERR bit in the Error and Status Register). After the transition to

'Error Passive' state, the Tx_Err_Counter does not increment anymore by acknowledge errors. Therefore the device never goes to the 'Bus Off' state.

- If the Rx_Err_Counter increases to a value greater than 127, it is not incremented further, even if more errors are detected while being a receiver. At the next successful message reception, the counter is set to a value between 119 and 127 to resume to 'Error Active' state.

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	Rx_Err_Counter								Tx_Err_Counter							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 407. Error Counter Register (ECR)

Table 484. Error Counter Register (ECR) field descriptions

Field	Description
16:23 Rx_Err_Counter	Receive Error Counter
24:31 Tx_Err_Counter	Transmit Error Counter

40.3.4.8 Error and Status Register (ESR)

This register reflects various error conditions, some general status of the device and it is the source of four interrupts to the CPU . The reported error conditions (bits 16 to 21) are those that occurred since the last time the CPU read this register. The CPU read action clears bits 16 to 21. Bits 22 to 28 are status bits.

Most bits in this register are read only, except TWRN_INT, RWRN_INT, BOFF_INT and ERR_INT, that are interrupt flags that can be cleared by writing 1 to them (writing 0 has no effect). See *Section 40.4.10* for more details.

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	TWRN_INT	RWRN_INT
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	BIT1_ERR	BIT0_ERR	ACK_ERR	CRC_ERR	FRM_ERR	STF_ERR	TX_WRN	RX_WRN	IDLE	TXRX	FLT_CONF		0	BOFF_INT	ERR_INT	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 408. Error and Status Register (ESR)

Table 485. Error and Status Register (ESR) field description

Field	Description
14 TWRN_INT	Tx Warning Interrupt Flag If the WRN_EN bit in MCR is asserted, the TWRN_INT bit is set when the TX_WRN flag transition from 0 to 1, meaning that the Tx error counter reached 96. If the corresponding mask bit in the Control Register (TWRN_MSK) is set, an interrupt is generated to the CPU. This bit is cleared by writing it to 1. Writing 0 has no effect. 0 No such occurrence 1 The Tx error counter transitioned from < 96 to >= 96
15 RWRN_INT	Rx Warning Interrupt Flag If the WRN_EN bit in MCR is asserted, the RWRN_INT bit is set when the RX_WRN flag transition from 0 to 1, meaning that the Rx error counters reached 96. If the corresponding mask bit in the Control Register (RWRN_MSK) is set, an interrupt is generated to the CPU. This bit is cleared by writing it to 1. Writing 0 has no effect. 0 No such occurrence 1 The Rx error counter transitioned from < 96 to >= 96
16 BIT1_ERR	Bit1 Error This bit indicates when an inconsistency occurs between the transmitted and the received bit in a message. 0 No such occurrence 1 At least one bit sent as recessive is received as dominant **Note:** This bit is not set by a transmitter in case of arbitration field or ACK slot, or in case of a node sending a passive error flag that detects dominant bits.
17 BIT0_ERR	Bit0 Error This bit indicates when an inconsistency occurs between the transmitted and the received bit in a message. 0 No such occurrence 1 At least one bit sent as dominant is received as recessive

Table 485. Error and Status Register (ESR) field description(Continued)

Field	Description
18 ACK_ERR	Acknowledge Error This bit indicates that an Acknowledge Error has been detected by the transmitter node, that is, a dominant bit has not been detected during the ACK SLOT. 0 No such occurrence 1 An ACK error occurred since last read of this register
19 CRC_ERR	Cyclic Redundancy Check Error This bit indicates that a CRC Error has been detected by the receiver node, that is, the calculated CRC is different from the received. 0 No such occurrence 1 A CRC error occurred since last read of this register
20 FRM_ERR	Form Error This bit indicates that a Form Error has been detected by the receiver node, that is, a fixed-form bit field contains at least one illegal bit. 0 No such occurrence 1 A Form Error occurred since last read of this register
21 STF_ERR	Stuffing Error This bit indicates that a Stuffing Error has been detected. 0 No such occurrence 1 A Stuffing Error occurred since last read of this register
22 TX_WRN	TX Error Counter This bit indicates when repetitive errors are occurring during message transmission. 0 No such occurrence 1 TX_Err_Counter >= 96
23 RX_WRN	Rx Error Counter This bit indicates when repetitive errors are occurring during message reception. 0 No such occurrence 1 Rx_Err_Counter >= 96
24 IDLE	CAN bus IDLE state This bit indicates when the CAN bus is in IDLE state. 0 No such occurrence 1 CAN bus is now IDLE
25 TXRX	Current FlexCAN status (transmitting/receiving) This bit indicates if FlexCAN is transmitting or receiving a message when the CAN bus is not in IDLE state. This bit has no meaning when IDLE is asserted. 0 FlexCAN is receiving a message (IDLE = 0) 1 FlexCAN is transmitting a message (IDLE = 0)
26:27 FLT_CONF	Fault Confinement State This 2-bit field indicates the Confinement State of the FlexCAN module, as shown in *Table 486*. If the LOM bit in the Control Register is asserted, the FLT_CONF field will indicate “Error Passive”. Since the Control Register is not affected by soft reset, the FLT_CONF field will not be affected by soft reset if the LOM bit is asserted.

Table 485. Error and Status Register (ESR) field description(Continued)

Field	Description
29 BOFF_INT	Bus Off Interrupt This bit is set when FlexCAN enters 'Bus Off' state. If the corresponding mask bit in the Control Register (BOFF_MSK) is set, an interrupt is generated to the CPU. This bit is cleared by writing it to 1. Writing 0 has no effect. 0 No such occurrence 1 FlexCAN module entered 'Bus Off' state
30 ERR_INT	Error Interrupt This bit indicates that at least one of the Error Bits (bits 16–21) is set. If the corresponding mask bit in the Control Register (ERR_MSK) is set, an interrupt is generated to the CPU . This bit is cleared by writing it to 1 .Writing 0 has no effect. 0 No such occurrence 1 Indicates setting of any Error Bit in the Error and Status Register

Table 486. Fault confinement state

Value	Meaning
00	Error Active
01	Error Passive
1X	Bus Off

40.3.4.9 Interrupt Masks 1 Register (IMASK1)

This register allows to enable or disable any number of a range of 32 Message Buffer Interrupts. It contains one interrupt mask bit per buffer, enabling the CPU to determine which buffer generates an interrupt after a successful transmission or reception (that is, when the corresponding IFLAG1 bit is set).

Address: 0x0028 Access: User read/write

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

Figure 409. Interrupt Masks 1 Register (IMASK1)

Table 487. IMASK1 field descriptions

Field	Description
0:31 BUF[31:0]M	Buffer MB_i Mask Each bit enables or disables the respective FlexCAN Message Buffer (MB0 to MB31) Interrupt. 0 The corresponding buffer Interrupt is disabled 1 The corresponding buffer Interrupt is enabled **Note:** Setting or clearing a bit in the IMASK1 Register can assert or negate an interrupt request, if the corresponding IFLAG1 bit is set.

40.3.4.10 Interrupt Flags 1 Register (IFLAG1)

This register defines the flags for 32 Message Buffer interrupts and FIFO interrupts. It contains one interrupt flag bit per buffer. Each successful transmission or reception sets the corresponding IFLAG1 bit. If the corresponding IMASK1 bit is set, an interrupt will be generated. The Interrupt flag must be cleared by writing it to 1. Writing 0 has no effect.

When the AEN bit in the MCR is set (Abort enabled), while the IFLAG1 bit is set for a MB configured as Tx, the writing access done by CPU into the corresponding MB will be blocked.

When the FEN bit in the MCR is set (FIFO enabled), the function of the eight least significant interrupt flags (BUF7I – BUF0I) is changed to support the FIFO operation. BUF7I, BUF6I and BUF5I indicate operating conditions of the FIFO, while BUF4I to BUF0I are not used.

Address: 0x0030 Access: User read/write

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

Figure 410. Interrupt Flags 1 Register (IFLAG1)

Table 488. IFLAG1 field descriptions

Field	Description
0:23 BUF[31:8]I	Buffer MB_i Interrupt Each bit flags the respective FlexCAN Message Buffer (MB8 to MB31) interrupt. 0 No such occurrence 1 The corresponding MB has successfully completed transmission or reception
24 BUF7I	Buffer MB7 Interrupt or “FIFO Overflow” If the FIFO is not enabled, this bit flags the interrupt for MB7. If the FIFO is enabled, this flag indicates an overflow condition in the FIFO (frame lost because FIFO is full). 0 No such occurrence 1 MB7 completed transmission/reception or FIFO overflow
25 BUF6I	Buffer MB6 Interrupt or “FIFO Warning” If the FIFO is not enabled, this bit flags the interrupt for MB6. If the FIFO is enabled, this flag indicates that 5 out of 6 buffers of the FIFO are already occupied (FIFO almost full). 0 No such occurrence 1 MB6 completed transmission/reception or FIFO almost full
26 BUF5I	Buffer MB5 Interrupt or “Frames available in FIFO” If the FIFO is not enabled, this bit flags the interrupt for MB5. If the FIFO is enabled, this flag indicates that at least one frame is available to be read from the FIFO. 0 No such occurrence. 1 MB5 completed transmission/reception or frames available in the FIFO.
27:31 BUF[4:0]I	Buffer MB_i Interrupt or “reserved” If the FIFO is not enabled, these bits flag the interrupts for MB0 to MB4. If the FIFO is enabled, these flags are not used and must be considered as reserved locations. 0 No such occurrence. 1 Corresponding MB completed transmission/reception.

40.3.4.11 Rx Individual Mask Registers (RXIMRn)

These registers are used as acceptance masks for ID filtering in Rx MBs and the FIFO. If the FIFO is not enabled, one mask register is provided for each available Message Buffer, providing ID masking capability on a per Message Buffer basis. When the FIFO is enabled (FEN bit in MCR is set), the first eight Mask Registers apply to the eight elements of the FIFO filter table (on a one-to-one correspondence), while the rest of the registers apply to the regular MBs, starting from MB8.

The Individual Rx Mask Registers are implemented in RAM, so they are not affected by reset and must be explicitly initialized prior to any reception . Furthermore, they can only be accessed by the CPU while the module is in Freeze Mode. Out of Freeze Mode, write accesses are blocked and read accesses will return “all zeros”. Furthermore, if the BCC bit in the MCR is negated, any read or write operation to these registers results in access error.

Address: 0x0880 + n*0x4 (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	MI31	MI30	MI29	MI28	MI27	MI26	MI25	MI24	MI23	MI22	MI21	MI20	MI19	MI18	MI17	MI16
Reset	0	0	0	0	0	0	0	0	0	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	MI15	MI14	MI13	MI12	MI11	MI10	MI9	MI8	MI7	MI6	MI5	MI4	MI3	MI2	MI1	MI0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 411. Rx Individual Mask Registers (RXIMRn)

Table 489. RXIMRn field descriptions

Field	Description
0:31 MI[31:0]	Mask Bits For normal Rx MBs, the mask bits affect the ID filter programmed on the MB. For the Rx FIFO, the mask bits affect all bits programmed in the filter table (ID, IDE, RTR). 0 The corresponding bit in the filter is “don’t care” 1 The corresponding bit in the filter is checked against the one received

40.4 Functional description

40.4.1 Overview

The FlexCAN module is a CAN protocol engine with a very flexible mailbox system for transmitting and receiving CAN frames. The mailbox system is composed by a set of as many as 32 Message Buffers (MB) that store configuration and control data, time stamp, message ID and data (see *Section 40.3.2*). The memory corresponding to the first eight Message Buffers can be configured to support a FIFO reception scheme with a powerful ID filtering mechanism, capable of checking incoming frames against a table of IDs (as many as 8 extended IDs or 16 standard IDs or 32 eight-bit ID slices), each one with its own individual mask register. Simultaneous reception through FIFO and mailbox is supported. For mailbox reception, a matching algorithm makes it possible to store received frames only into MBs that have the same ID programmed on its ID field. A masking scheme makes it possible to match the ID programmed on the MB with a range of IDs on received CAN frames. For transmission, an arbitration algorithm decides the prioritization of MBs to be transmitted based on the message ID (optionally augmented by 3 local priority bits) or the MB ordering.

Before proceeding with the functional description, an important concept must be explained. A Message Buffer is said to be “active” at a given time if it can participate in the matching and arbitration algorithms that are happening at that time. An Rx MB with a 0000 code is inactive (refer to *Table 473*). Similarly, a Tx MB with a 1000 or 1001 code is also inactive (refer to *Table 474*). An MB not programmed with 0000, 1000, or 1001 will be temporarily deactivated (will not participate in the current arbitration or matching run) when the CPU writes to the C/S field of that MB (see *Section 40.4.6.2: Message Buffer deactivation*).

40.4.2 Transmit process

In order to transmit a CAN frame, the CPU must prepare a Message Buffer for transmission by executing the following procedure:

1. If the message buffer is active (transmission pending), write '1000' to the Code field to inactivate the message buffer. The deactivated message buffer can transmit without setting IFLAG and without updating the CODE field (see *Section 40.4.6.2: Message Buffer deactivation*)
2. Write the ID word
3. Write the data bytes
4. Write the Length, Control and Code fields of the Control and Status word to activate the MB

Once the MB is activated in the fourth step, it will participate into the arbitration process and eventually be transmitted according to its priority. At the end of the successful transmission, the value of the Free Running Timer is written into the Time Stamp field, the Code field in the Control and Status word is updated, a status flag is set in the Interrupt Flag Register and an interrupt is generated if allowed by the corresponding Interrupt Mask Register bit. The new Code field after transmission depends on the code that was used to activate the MB in step 4 (see *Table 473* and *Table 474* in *Section 40.3.2*). When the Abort feature is enabled (AEN in MCR is asserted), after the Interrupt Flag is asserted for a MB configured as transmit buffer, the MB is blocked, therefore the CPU is not able to update it until the Interrupt Flag be negated by CPU. It means that the CPU must clear the corresponding IFLAG before starting to prepare this MB for a new transmission or reception.

40.4.3 Arbitration process

The arbitration process is an algorithm executed by the MBM that scans the whole MB memory looking for the highest priority message to be transmitted. All MBs programmed as transmit buffers will be scanned to find the lowest ID[q] or the lowest MB number or the highest priority, depending on the LBUF and LPRIO_EN bits on the Control Register. The arbitration process is triggered in the following events:

- During transmission of a CRC field of the CAN frame
- During transmission of the error delimiter field of the CAN frame
- During Intermission, if the winner MB defined in a previous arbitration was deactivated, or if there was no MB to transmit, but the CPU wrote to the C/S word of any MB after the previous arbitration finished
- When MBM is in Idle or Bus Off state and the CPU writes to the C/S word of any MB
- Upon leaving Freeze Mode

When LBUF is asserted, the LPRIO_EN bit has no effect and the lowest number buffer is transmitted first. When LBUF and LPRIO_EN are both negated, the MB with the lowest ID is transmitted first but. If LBUF is negated and LPRIO_EN is asserted, the PRIO bits augment the ID used during the arbitration process. With this extended ID concept, arbitration is done based on the full 32-bit ID and the PRIO bits define which MB should be transmitted first, therefore MBs with PRIO = 000 have higher priority. If two or more MBs have the same priority, the regular ID will determine the priority of transmission. If two or more MBs have

q. Actually, if LBUF is negated, the arbitration considers not only the ID, but also the RTR and IDE bits placed inside the ID at the same positions they are transmitted in the CAN frame.

the same priority (3 extra bits) and the same regular ID, the lowest MB will be transmitted first.

Once the highest priority MB is selected, it is transferred to a temporary storage space called Serial Message Buffer (SMB), which has the same structure as a normal MB but is not user accessible. This operation is called “move-out” and after it is done, write access to the corresponding MB is blocked (if the AEN bit in the MCR is asserted). The write access is released in the following events:

- After the MB is transmitted
- FlexCAN enters in HALT or BUS OFF
- FlexCAN loses the bus arbitration or there is an error during the transmission

At the first opportunity window on the CAN bus, the message on the SMB is transmitted according to the CAN protocol rules. FlexCAN transmits as many as 8 data bytes, even if the DLC (Data Length Code) value is bigger.

40.4.4 Receive process

To be able to receive CAN frames into the mailbox MBs, the CPU must prepare one or more Message Buffers for reception by executing the following steps:

1. If the MB has a pending transmission, write an ABORT code (‘1001’) to the Code field of the Control and Status word to request an abortion of the transmission, then read back the Code field and the IFLAG register to check if the transmission was aborted (see *Section 40.4.6.1: Transmission abort mechanism*). If backwards compatibility is desired (AEN in MCR negated), just write ‘1000’ to the Code field to inactivate the MB, but then the pending frame may be transmitted without notification (see *Section 40.4.6.2: Message Buffer deactivation*). If the MB already programmed as a receiver, just write ‘0000’ to the Code field of the Control and Status word to keep the MB inactive
2. Write the ID word
3. Write ‘0100’ to the Code field of the Control and Status word to activate the MB

Once the MB is activated in the third step, it will be able to receive frames that match the programmed ID. At the end of a successful reception, the MB is updated by the MBM as follows:

- The value of the Free Running Timer is written into the Time Stamp field
- The received ID, Data (8 bytes at most) and Length fields are stored
- The Code field in the Control and Status word is updated (see *Table 473* and *Table 474* in *Section 40.3.2*)
- A status flag is set in the Interrupt Flag Register and an interrupt is generated if allowed by the corresponding Interrupt Mask Register bit

Upon receiving the MB interrupt, the CPU should service the received frame using the following procedure:

1. Read the Control and Status word (mandatory – activates an internal lock for this buffer)
2. Read the ID field (optional: needed only if a mask was used)
3. Read the Data field
4. Read the Free Running Timer (optional: releases the internal lock)

Upon reading the Control and Status word, if the BUSY bit is set in the Code field, then the CPU should defer the access to the MB until this bit is negated. Reading the Free Running Timer is not mandatory. If not executed the MB remains locked, unless the CPU reads the C/S word of another MB. Note that only a single MB is locked at a time. The only mandatory CPU read operation is the one on the Control and Status word to assure data coherency (see *Section 40.4.6*).

The CPU should synchronize to frame reception by the status flag bit for the specific MB in one of the IFLAG Registers and not by the Code field of that MB. Polling the Code field does not work because once a frame was received and the CPU services the MB (by reading the C/S word followed by unlocking the MB), the Code field will not return to EMPTY. It will remain FULL, as explained in *Table 473*. If the CPU tries to work around this behavior by writing to the C/S word to force an EMPTY code after reading the MB, the MB is actually deactivated from any currently ongoing matching process. As a result, a newly received frame matching the ID of that MB may be lost. In summary: never perform polling by directly reading the C/S word of the MBs; Instead, read the IFLAG registers.

Note that the received ID field is always stored in the matching MB, thus the contents of the ID field in an MB may change if the match was due to masking. Note also that FlexCAN does receive frames transmitted by itself if there exists an Rx matching MB, provided the SRX_DIS bit in the MCR is not asserted. If SRX_DIS is asserted, FlexCAN will not store frames transmitted by itself in any MB, even if it contains a matching MB, and no interrupt flag or interrupt signal will be generated due to the frame reception.

To be able to receive CAN frames through the FIFO, the CPU must enable and configure the FIFO during Freeze Mode (see *Section 40.4.7*). Upon receiving the frames available interrupt from FIFO, the CPU should service the received frame using the following procedure:

1. Read the Control and Status word (optional: needed only if a mask was used for IDE and RTR bits)
2. Read the ID field (optional: needed only if a mask was used)
3. Read the Data field
4. Clear the frames available interrupt (mandatory: release the buffer and allow the CPU to read the next FIFO entry)

40.4.5 Matching process

The matching process is an algorithm executed by the MBM that scans the MB memory looking for Rx MBs programmed with the same ID as the one received from the CAN bus. If the FIFO is enabled, the 8-entry ID table from FIFO is scanned first and then, if a match is not found within the FIFO table, the other MBs are scanned. In the event that the FIFO is full, the matching algorithm will always look for a matching MB outside the FIFO region.

When the frame is received, it is temporarily stored in a hidden auxiliary MB called Serial Message Buffer (SMB). The matching process takes place during the CRC field of the received frame. If a matching ID is found in the FIFO table or in one of the regular MBs, the contents of the SMB will be transferred to the FIFO or to the matched MB during the 6th bit of the End-Of-Frame field of the CAN protocol. This operation is called “move-in”. If any protocol error (such as CRC, ACK) is detected, than the move-in operation does not happen.

For the regular mailbox MBs, an MB is said to be “free to receive” a new frame if the following conditions are satisfied:

- The MB is not locked (see *Section 40.4.6.3: Message Buffer lock mechanism*)
- The Code field is either EMPTY or else it is FULL or OVERRUN but the CPU has already serviced the MB (read the C/S word and then unlocked the MB)

If the first MB with a matching ID is not “free to receive” the new frame, then the matching algorithm keeps looking for another free MB until it finds one. If it cannot find one that is free, then it will overwrite the last matching MB (unless it is locked) and set the Code field to OVERRUN (refer to *Table 473* and *Table 474*). If the last matching MB is locked, then the new message remains in the SMB, waiting for the MB to be unlocked (see *Section 40.4.6.3: Message Buffer lock mechanism*).

Suppose, for example, that the FIFO is disabled and there are two MBs with the same ID, and FlexCAN starts receiving messages with that ID. Let us say that these MBs are the second and the fifth in the array. When the first message arrives, the matching algorithm will find the first match in MB number 2. The code of this MB is EMPTY, so the message is stored there. When the second message arrives, the matching algorithm will find MB number 2 again, but it is not “free to receive”, so it will keep looking and find MB number 5 and store the message there. If yet another message with the same ID arrives, the matching algorithm finds out that there are no matching MBs that are “free to receive”, so it decides to overwrite the last matched MB, which is number 5. In doing so, it sets the Code field of the MB to indicate OVERRUN.

The ability to match the same ID in more than one MB can be exploited to implement a reception queue (in addition to the full featured FIFO) to allow more time for the CPU to service the MBs. By programming more than one MB with the same ID, received messages will be queued into the MBs. The CPU can examine the Time Stamp field of the MBs to determine the order in which the messages arrived.

The matching algorithm described above can be changed to be the same one used in previous versions of the FlexCAN module. When the BCC bit in the MCR is negated, the matching algorithm stops at the first MB with a matching ID that it founds, whether this MB is free or not. As a result, the message queueing feature does not work if the BCC bit is negated.

Matching to a range of IDs is possible by using ID Acceptance Masks. FlexCAN supports individual masking per MB. Please refer to *Section 40.3.4.11: Rx Individual Mask Registers (RXIMRn).* During the matching algorithm, if a mask bit is asserted, then the corresponding ID bit is compared. If the mask bit is negated, the corresponding ID bit is “don’t care”. Please note that the Individual Mask Registers are implemented in RAM, so they are not initialized out of reset. Also, they can only be programmed if the BCC bit is asserted and while the module is in Freeze Mode.

FlexCAN also supports an alternate masking scheme with only three mask registers (RGXMASK, RX14MASK and RX15MASK) for backwards compatibility. This alternate masking scheme is enabled when the BCC bit in the MCR is negated.

40.4.6 Data coherence

In order to maintain data coherency and FlexCAN proper operation, the CPU must obey the rules described in *Section 40.4.2: Transmit process* and *Section 40.4.4.* Any form of CPU accessing an MB structure within FlexCAN other than those specified may cause FlexCAN to behave in an unpredictable way.

40.4.6.1 Transmission abort mechanism

The abort mechanism provides a safe way to request the abortion of a pending transmission. A feedback mechanism is provided to inform the CPU if the transmission was aborted or if the frame could not be aborted and was transmitted instead. In order to maintain backwards compatibility, the abort mechanism must be explicitly enabled by asserting the AEN bit in the MCR.

In order to abort a transmission, the CPU must write a specific abort code (1001) to the Code field of the Control and Status word. When the abort mechanism is enabled, the active MBs configured as transmission must be aborted first and then they may be updated. If the abort code is written to an MB that is currently being transmitted, or to an MB that was already loaded into the SMB for transmission, the write operation is blocked and the MB is not deactivated, but the abort request is captured and kept pending until one of the following conditions are satisfied:

- The module loses the bus arbitration
- There is an error during the transmission
- The module is put into Freeze Mode

If none of conditions above are reached, the MB is transmitted correctly, the interrupt flag is set in the IFLAG register and an interrupt to the CPU is generated (if enabled). The abort request is automatically cleared when the interrupt flag is set. In the other hand, if one of the above conditions is reached, the frame is not transmitted, therefore the abort code is written into the Code field, the interrupt flag is set in the IFLAG and an interrupt is (optionally) generated to the CPU.

If the CPU writes the abort code before the transmission begins internally, then the write operation is not blocked, therefore the MB is updated and no interrupt flag is set. In this way the CPU just needs to read the abort code to make sure the active MB was deactivated. Although the AEN bit is asserted and the CPU wrote the abort code, in this case the MB is deactivated and not aborted, because the transmission did not start yet. One MB is only aborted when the abort request is captured and kept pending until one of the previous conditions are satisfied.

The abort procedure can be summarized as follows:

1. CPU writes 1001 into the code field of the C/S word
2. CPU reads the CODE field and compares it to the value that was written
3. If the CODE field that was read is different from the value that was written, the CPU must read the corresponding IFLAG to check if the frame was transmitted or it is being currently transmitted. If the corresponding IFLAG is set, the frame was transmitted. If the corresponding IFLAG is reset, the CPU must wait for it to be set, and then the CPU must read the CODE field to check if the MB was aborted (CODE = 1001) or it was transmitted (CODE = 1000)

Note: An abort request to a TxMB can block any write operation into its CODE field. Therefore, the TxMB cannot be aborted or deactivated until it completes a transmission by winning the CAN bus arbitration.

40.4.6.2 Message Buffer deactivation

Deactivation is mechanism provided to maintain data coherence when the CPU writes to the Control and Status word of active MBs out of Freeze Mode. Any CPU write access to the Control and Status word of an MB causes that MB to be excluded from the transmit or

receive processes during the current matching or arbitration round. The deactivation is temporary, affecting only for the current match/arbitration round.

The purpose of deactivation is data coherency. The match/arbitration process scans the MBs to decide which MB to transmit or receive. If the CPU updates the MB in the middle of a match or arbitration process, the data of that MB may no longer be coherent, therefore deactivation of that MB is done.

Even with the coherence mechanism described above, writing to the Control and Status word of active MBs when not in Freeze Mode may produce undesirable results. Examples are:

- Matching and arbitration are one-pass processes. If MBs are deactivated after they are scanned, no re-evaluation is done to determine a new match/winner. If an Rx MB with a matching ID is deactivated during the matching process after it was scanned, then this MB is marked as invalid to receive the frame, and FlexCAN will keep looking for another matching MB within the ones it has not scanned yet. If it cannot find one, then the message will be lost. Suppose, for example, that two MBs have a matching ID to a received frame, and the user deactivated the first matching MB after FlexCAN has scanned the second. The received frame will be lost even if the second matching MB was “free to receive”.
- If a Tx MB containing the lowest ID is deactivated after FlexCAN has scanned it, then FlexCAN will look for another winner within the MBs that it has not scanned yet. Therefore, it may transmit an MB with ID that may not be the lowest at the time because a lower ID might be present in one of the MBs that it had already scanned before the deactivation.
- There is a point in time until which the deactivation of a Tx MB causes it not to be transmitted (end of move-out). After this point, it is transmitted but no interrupt is issued and the Code field is not updated. In order to avoid this situation, the abort procedures described in *Section 40.4.6.1: Transmission abort mechanism* should be used.

40.4.6.3 Message Buffer lock mechanism

Besides MB deactivation, FlexCAN has another data coherence mechanism for the receive process. When the CPU reads the Control and Status word of an “active not empty” Rx MB, FlexCAN assumes that the CPU wants to read the whole MB in an atomic operation, and thus it sets an internal lock flag for that MB. The lock is released when the CPU reads the Free Running Timer (global unlock operation), or when it reads the Control and Status word of another MB. The MB locking is done to prevent a new frame to be written into the MB while the CPU is reading it.

Note: The locking mechanism only applies to Rx MBs that have a code different than INACTIVE (‘0000’) or EMPTY[r] (‘0100’). Also, Tx MBs cannot be locked.

Suppose, for example, that the FIFO is disabled and the second and the fifth MBs of the array are programmed with the same ID, and FlexCAN has already received and stored messages into these two MBs. Suppose now that the CPU decides to read MB number 5 and at the same time another message with the same ID is arriving. When the CPU reads the Control and Status word of MB number 5, this MB is locked. The new message arrives and the matching algorithm finds out that there are no “free to receive” MBs, so it decides to override MB number 5. However, this MB is locked, so the new message cannot be written

r. In previous FlexCAN versions, reading the C/S word locked the MB even if it was EMPTY. In current FlexCAN versions, this behavior is maintained when the BCC bit is negated.

there. It will remain in the SMB waiting for the MB to be unlocked, and only then will be written to the MB. If the MB is not unlocked in time and yet another new message with the same ID arrives, then the new message overwrites the one on the SMB and there will be no indication of lost messages either in the Code field of the MB or in the Error and Status Register.

While the message is being moved-in from the SMB to the MB, the BUSY bit on the Code field is asserted. If the CPU reads the Control and Status word and finds out that the BUSY bit is set, it should defer accessing the MB until the BUSY bit is negated.

Note: If the BUSY bit is asserted or if the MB is empty, then reading the Control and Status word does not lock the MB.

Deactivation takes precedence over locking. If the CPU deactivates a locked Rx MB, then its lock status is negated and the MB is marked as invalid for the current matching round. Any pending message on the SMB will not be transferred anymore to the MB.

40.4.7 Rx FIFO

The receive-only FIFO is enabled by asserting CAN_MCR[FEN]. The reset value of this bit is '0' to maintain software backwards compatibility with previous versions of the module that did not have the FIFO feature. When the FIFO is enabled, the memory region normally occupied by the first eight MBs (0x80–0xFF) is now reserved for use of the FIFO engine (see *Section 40.3.3*). Management of read and write pointers is done internally by the FIFO engine. The CPU can read the received frames sequentially, in the order they were received, by repeatedly accessing a Message Buffer structure at the beginning of the memory.

The FIFO can store as many as six frames pending service by the CPU. An interrupt is sent to the CPU when new frames are available in the FIFO. Upon receiving the interrupt, the CPU must read the frame (accessing an MB in the 0x80 address) and then clear the interrupt. The act of clearing the interrupt triggers the FIFO engine to replace the MB in 0x80 with the next frame in the queue, and then issue another interrupt to the CPU. If the FIFO is full and more frames continue to be received, an OVERFLOW interrupt is issued to the CPU and subsequent frames are not accepted until the CPU creates space in the FIFO by reading one or more frames. A warning interrupt is also generated when five frames are accumulated in the FIFO.

A powerful filtering scheme is provided to accept only frames intended for the target application, thus reducing the interrupt servicing work load. The filtering criteria is specified by programming a table of eight 32-bit registers that can be configured to one of the following formats (see also *Section 40.3.3*):

- Format A: 8 extended or standard IDs (including IDE and RTR)
- Format B: 16 standard IDs or 16 extended 14-bit ID slices (including IDE and RTR)
- Format C: 32 standard or extended 8-bit ID slices

Note: A chosen format is applied to all eight registers of the filter table. It is not possible to mix formats within the table.

The eight elements of the filter table are individually affected by the first eight Individual Mask Registers (RXIMR0–RXIMR7), allowing very powerful filtering criteria to be defined. The rest of the RXIMR, starting from RXIM8, continue to affect the regular MBs, starting from MB8. If the BCC bit is negated (or if the RXIMR are not available for the particular MCU), then the FIFO filter table is affected by the legacy mask registers as follows: element

6 is affected by RX14MASK, element 7 is affected by RX15MASK and the other elements (0 to 5) are affected by RXGMASK.

40.4.7.1 Precautions when using Global Mask and Individual Mask registers

Mask filtering alignment is affected based on the setting of the FEN and BCC of MCR. The following table shows recommended actions depending on FEN and BCC settings.

Table 490. Recommended FEN and BCC settings

Case	MCR[FEN] Rx FIFO	MCR[BCC] Rx Individual Mask	Notes
1	FEN = 0	BCC = 0	RXGMASK, RX14MASK, and RX15MASK can safely be used. This allows backwards compatibility to older devices (e.g., devices without the individual masks feature). In this case, individual masks are not used.
2	FEN = 1	BCC = 0	1st alternative: Do not use RXGMASK, RX14MASK, and RX15MASK in this case, leave the masks in their reset state.
3	FEN = 1	BCC = 0	2nd alternative: Do not configure any MB as Rx (that is let all MBs as either Tx or inactive). In this case, RXGMASK, RX14MASK, and RX15MASK can be used to affect ID Tables without affecting the filtering process for Rx MBs.
4	don't care	BCC = 1	If MCR[BCC] = 1, then the RXIMRs are enabled. Thus, RXGMASK, RX14MASK, and RX15MASK are not used. Particularly, when MCR[FEN] = 0, RxFIFO is disabled; RXGMASK, RX14MASK, and RX15MASK do not affect filtering. Individual masks are used.

40.4.8 CAN protocol related features

40.4.8.1 Remote frames

A remote frame is a special kind of frame. The user can program a MB to be a Request Remote Frame by writing the MB as Transmit with the RTR bit set to '1'. After the Remote Request frame is transmitted successfully, the MB becomes a Receive Message Buffer, with the same ID as before.

When a Remote Request frame is received by FlexCAN, its ID is compared to the IDs of the transmit message buffers with the Code field '1010'. If there is a matching ID, then this MB frame will be transmitted. Note that if the matching MB has the RTR bit set, then FlexCAN will transmit a Remote Frame as a response.

A received Remote Request Frame is not stored in a receive buffer. It is only used to trigger a transmission of a frame in response. The mask registers are not used in remote frame matching, and all ID bits (except RTR) of the incoming received frame should match.

In the case that a Remote Request Frame was received and matched an MB, this message buffer immediately enters the internal arbitration process, but is considered as normal Tx

MB, with no higher priority. The data length of this frame is independent of the DLC field in the remote frame that initiated its transmission.

If the Rx FIFO is enabled (CAN_MCR[FEN] set), FlexCAN will not generate an automatic response for Remote Request Frames that match the FIFO filtering criteria. If the remote frame matches one of the target IDs, it will be stored in the FIFO and presented to the CPU. Note that for filtering formats A and B, it is possible to select whether remote frames are accepted or not. For format C, remote frames are always accepted (if they match the ID).

40.4.8.2 Overload frames

FlexCAN does transmit overload frames due to detection of following conditions on CAN bus:

- Detection of a dominant bit in the first/second bit of Intermission
- Detection of a dominant bit at the 7th bit (last) of End of Frame field (Rx frames)
- Detection of a dominant bit at the 8th bit (last) of Error Frame Delimiter or Overload Frame Delimiter

40.4.8.3 Time stamp

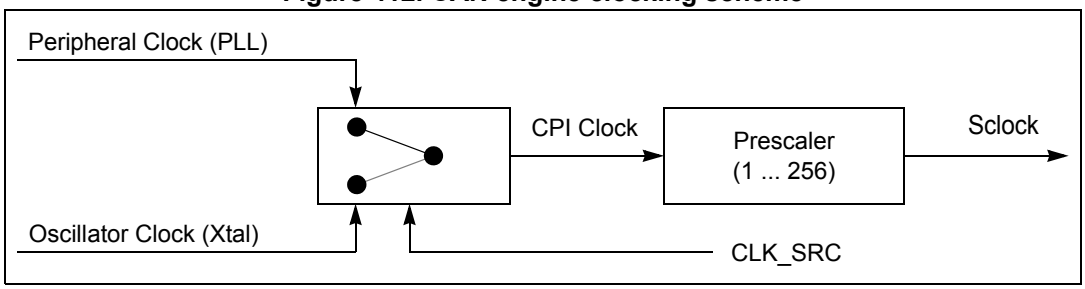
The value of the Free Running Timer is sampled at the beginning of the Identifier field on the CAN bus, and is stored at the end of “move-in” in the TIME STAMP field, providing network behavior with respect to time.

Note that the Free Running Timer can be reset upon a specific frame reception, enabling network time synchronization. Refer to TSYN description in *Section 40.3.4.2: Control Register (CTRL).*

40.4.8.4 Protocol timing

Figure 412 shows the structure of the clock generation circuitry that feeds the CAN Protocol Interface (CPI) submodule. The clock source bit (CAN_CTRL[CLK_SRC]) defines whether the internal clock is connected to the output of a crystal oscillator (Oscillator Clock) or to the Peripheral Clock (generally from a PLL). In order to guarantee reliable operation, the clock source should be selected while the module is in Disable Mode (bit MDIS set in the Module Configuration Register).

Figure 412. CAN engine clocking scheme



The crystal oscillator clock should be selected whenever a tight tolerance (to 0.1%) is required in the CAN bus timing. The crystal oscillator clock has better jitter performance than PLL generated clocks.

Note: This clock selection feature may not be available in all MCUs. A particular MCU may not have a PLL, in which case it would have only the oscillator clock, or it may use only the PLL

clock feeding the FlexCAN module. In these cases, bit CAN_CTRL[CLK_SRC] has no effect on the module operation.

The FlexCAN module supports a variety of means to set up bit timing parameters that are required by the CAN protocol. The Control Register has various fields used to control bit timing parameters: PRESDIV, PROPSEG, PSEG1, PSEG2 and RJW. See *Section 40.3.4.2: Control Register (CTRL).*

The PRESDIV field controls a prescaler that generates the Serial Clock (Sclock), whose period defines the 'time quantum' used to compose the CAN waveform. A time quantum is the atomic unit of time handled by the CAN engine.

Equation 31

$$f_{Tq} = \frac{f_{CANCLK}}{(Prescaler\ \ value)}$$

A bit time is subdivided into three segments[s] (reference *Figure 413* and *Table 491*):

- SYNC_SEG: This segment has a fixed length of one time quantum. Signal edges are expected to happen within this section
- Time Segment 1: This segment includes the Propagation Segment and the Phase Segment 1 of the CAN standard. It can be programmed by setting the PROPSEG and the PSEG1 fields of the CTRL Register so that their sum (plus 2) is in the range of 4 to 16 time quanta
- Time Segment 2: This segment represents the Phase Segment 2 of the CAN standard. It can be programmed by setting the PSEG2 field of the CTRL Register (plus 1) to be 2 to 8 time quanta long

Equation 32

$$Bit\ \ Rate = \frac{f_{Tq}}{(number\ \ of\ \ Time\ \ Quanta)}$$

s. For further explanation of the underlying concepts please refer to ISO/DIS 11519–1, Section 10.3.

Figure 413. Segments within the bit time

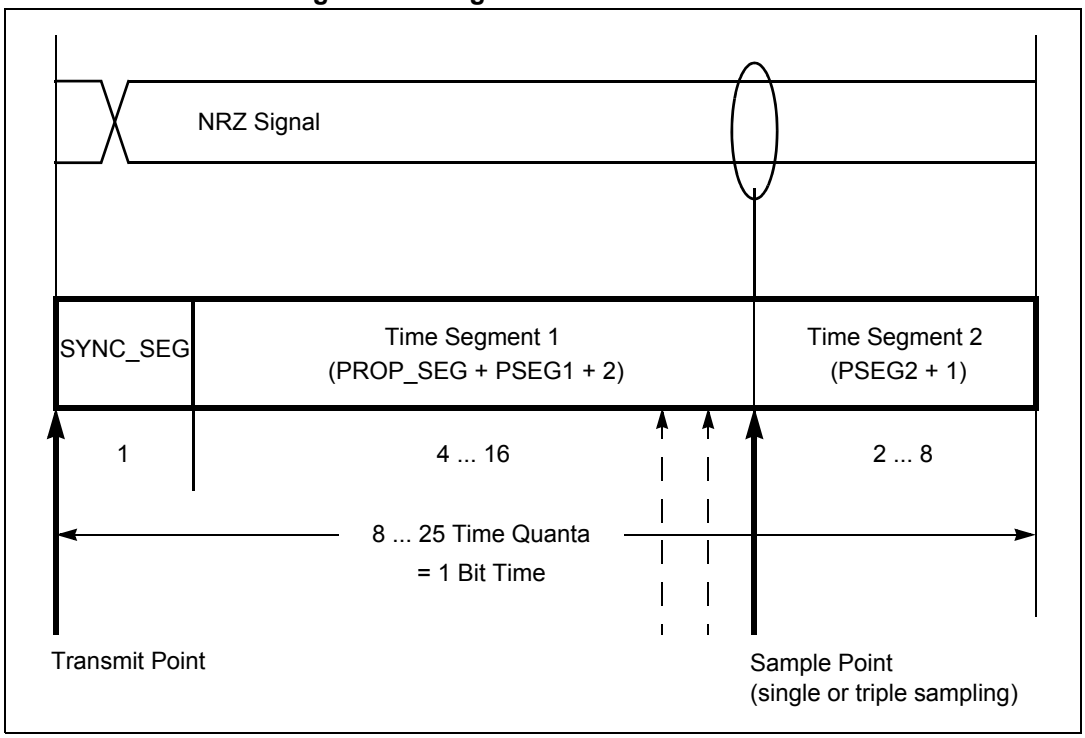


Table 491. Time segment syntax

Syntax	Description
SYNC_SEG	System expects transitions to occur on the bus during this period.
Transmit Point	A node in transmit mode transfers a new value to the CAN bus at this point.
Sample Point	A node samples the bus at this point. If the three samples per bit option is selected, then this point marks the position of the third sample.

Table 492 gives an overview of the CAN compliant segment settings and the related parameter values.

Table 492. CAN standard compliant bit time segment settings

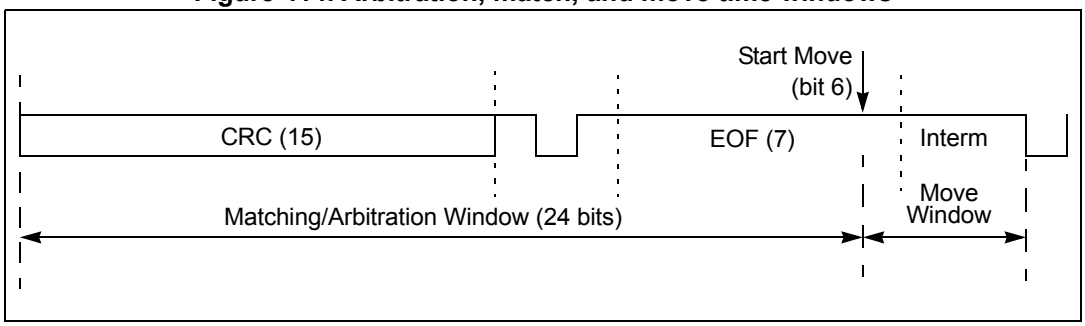
Time segment 1	Time segment 2	Resynchronization jump width
5 ... 10	2	1 ... 2
4 ... 11	3	1 ... 3
5 ... 12	4	1 ... 4
6 ... 13	5	1 ... 4
7 ... 14	6	1 .. 4
8 ... 15	7	1 ... 4
9 ... 16	8	1 ... 4

Note: It is the user's responsibility to ensure the bit time settings are in compliance with the CAN standard. For bit time calculations, use an IPT (Information Processing Time) of 2, which is the value implemented in the FlexCAN module.

40.4.8.5 Arbitration and matching timing

During normal transmission or reception of frames, the arbitration, matching, move-in and move-out processes are executed during certain time windows inside the CAN frame, as shown in *Figure 414*.

Figure 414. Arbitration, match, and move time windows



When doing matching and arbitration, FlexCAN needs to scan the whole Message Buffer memory during the available time slot. In order to have sufficient time to do that, the following requirements must be observed:

- A valid CAN bit timing must be programmed, as indicated in *Table 492*
- The peripheral clock frequency cannot be smaller than the oscillator clock frequency, that is, the PLL cannot be programmed to divide down the oscillator clock
- There must be a minimum ratio between the peripheral clock frequency and the CAN bit rate, as specified in *Table 493*

Table 493. Minimum ratio between peripheral clock frequency and CAN bit rate

Number of message buffers	Minimum ratio
16	8
32	8

A direct consequence of the first requirement is that the minimum number of time quanta per CAN bit must be 8, so the oscillator clock frequency should be at least 8 times the CAN bit rate. The minimum frequency ratio specified in *Table 493* can be achieved by choosing a high enough peripheral clock frequency when compared to the oscillator clock frequency, or by adjusting one or more of the bit timing parameters (PRESDIV, PROPSEG, PSEG1, PSEG2). As an example, taking the case of 32 MBs, if the oscillator and peripheral clock frequencies are equal and the CAN bit timing is programmed to have 8 time quanta per bit, then the prescaler factor (PRESDIV + 1) should be at least 2. For prescaler factor equal to 1 and CAN bit timing with 8 time quanta per bit, the ratio between peripheral and oscillator clock frequencies should be at least 2.

40.4.9 Modes of operation details

40.4.9.1 Freeze mode

This mode is entered by asserting CAN_MCR[HALT] or when the MCU is put into Debug Mode. In both cases it is also necessary that CAN_MCR[FRZ] is asserted and the module is not in any of the low power modes. When Freeze Mode is requested during transmission or reception, FlexCAN does the following:

- Waits to be in either Intermission, Passive Error, Bus Off or Idle state
- Waits for all internal activities like arbitration, matching, move-in and move-out to finish
- Ignores the Rx input pin and drives the Tx pin as recessive
- Stops the prescaler, thus halting all CAN protocol activities
- Grants write access to the Error Counters Register, which is read-only in other modes
- Sets the NOT_RDY and FRZ_ACK bits in MCR

After requesting Freeze Mode, the user must wait for CAN_MCR[FRZ_ACK] to be asserted before executing any other action, otherwise FlexCAN may operate in an unpredictable way. In Freeze mode, all memory mapped registers are accessible.

Exiting Freeze Mode is done in one of the following ways:

- CPU negates CAN_MCR[FRZ]
- The MCU is removed from Debug Mode and/or the HALT bit is negated

Once out of Freeze Mode, FlexCAN tries to resynchronize to the CAN bus by waiting for 11 consecutive recessive bits.

40.4.9.2 Module disable mode

This low power mode is entered when CAN_MCR[MDIS] is asserted by the CPU. When the FlexCAN module is disabled during Freeze Mode, the module sends a request to disable the clocks to the CPI and MBM submodules, sets CAN_MCR[MDISACK] and negates CAN_MCR[FRZ_ACK]. If the module is disabled during transmission or reception, FlexCAN does the following:

- Waits to be in either Idle or Bus Off state, or else waits for the third bit of Intermission and then checks it to be recessive
- Waits for all internal activities like arbitration, matching, move-in and move-out to finish
- Ignores its Rx input pin and drives its Tx pin as recessive
- Shuts down the clocks to the CPI and MBM submodules
- Sets the NOT_RDY and LPM_ACK bits in MCR

The Bus Interface Unit continues to operate, enabling the CPU to access memory mapped registers, except the Free Running Timer, the Error Counter Register, and the Message Buffers, which cannot be accessed when the module is in Disable Mode. Exiting from this mode is done by negating the MDIS bit, which will resume the clocks and negate CAN_MCR[LPM_ACK].

40.4.9.3 Stop mode

This is a system low power mode in which all MCU clocks are stopped for maximum power savings. If FlexCAN receives the global Stop Mode request during Freeze Mode, it sets CAN_MCR[LPM_ACK], negates CAN_MCR[FRZ_ACK] and then sends a Stop

Acknowledge signal to the CPU, in order to shut down the clocks globally. If Stop Mode is requested during transmission or reception, FlexCAN does the following:

- Waits to be in either Idle or Bus Off state, or else waits for the third bit of Intermission and checks it to be recessive
- Waits for all internal activities like arbitration, matching, move-in and move-out to finish
- Ignores its Rx input pin and drives its Tx pin as recessive
- Sets the NOT_RDY and LPM_ACK bits in MCR
- Sends a Stop Acknowledge signal to the CPU, so that it can shut down the clocks globally

Stop Mode is exited by the CPU resuming the clocks and removing the Stop Mode request.

40.4.10 Interrupts

The module can generate as many as 38 interrupt sources (32 interrupts due to message buffers and 6 interrupts due to ORed interrupts from MBs, Bus Off, Error, Tx Warning, Rx Warning, and Wake Up[t]). The number of actual sources depends on the configured number of Message Buffers.

Each one of the message buffers can be an interrupt source if its corresponding IMASK bit is set. There is no distinction between Tx and Rx interrupts for a particular buffer, under the assumption that the buffer is initialized for either transmission or reception. Each of the buffers has assigned a flag bit in the IFLAG Registers. The bit is set when the corresponding buffer completes a successful transmission/reception and is cleared when the CPU writes it to 1 (unless another interrupt is generated at the same time).

Note: It must be guaranteed that the CPU only clears the bit causing the current interrupt. For this reason, bit manipulation instructions (BSET) must not be used to clear interrupt flags. These instructions may cause accidental clearing of interrupt flags that are set after entering the current interrupt service routine.

If the Rx FIFO is enabled (CAN_MCR[FEN] set), the interrupts corresponding to MBs 0 to 7 have a different behavior. Bit 7 of the IFLAG1 becomes the FIFO Overflow flag; bit 6 becomes the FIFO Warning flag, bit 5 becomes the Frames Available in FIFO flag and bits 4:0 are unused. See *Section 40.3.4.10: Interrupt Flags 1 Register (IFLAG1)* for more information.

A combined interrupt for all MBs is also generated by an OR of all the interrupt sources from MBs. This interrupt is generated when any of the MBs generates an interrupt. In this case the CPU must read the IFLAG Registers to determine which MB caused the interrupt.

The other five interrupt sources (Bus Off, Error, Tx Warning, Rx Warning, and Wakeup) generate interrupts like the MB ones, and can be read from the ESR register. The Bus Off, Error, Tx Warning, and Rx Warning interrupt mask bits are located in the CTRL register[t].

t. Wakeup interrupt never occurs because self-wakeup mechanism is not implemented.

40.4.11 Bus interface

The CPU access to FlexCAN registers is subject to the following rules:

- Read and write access to supervisor registers in User Mode results in access error.
- Read and write access to unimplemented or reserved address space also results in access error. Any access to unimplemented MB or Rx Individual Mask Register locations results in access error. Any access to the Rx Individual Mask Register space when the BCC bit in the MCR is negated results in access error.
- If MAXMB is programmed with a value smaller than the available number of MBs, then the unused memory space can be used as general purpose RAM space. Note that the Rx Individual Mask Registers can only be accessed in Freeze Mode, and this is still true for unused space within this memory. Note also that reserved words within RAM cannot be used. As an example, suppose FlexCAN is configured with 32 MBs and MAXMB is programmed with 0. The maximum number of MBs in this case becomes 1. The MB memory starts at 0x0060, but the space from 0x0060 to 0x007F is reserved (for SMB usage), and the space from 0x0080 to 0x008F is used by the one MB. This leaves us with the available space from 0x0090 to 0x027F. The available memory in the Mask Registers space would be from 0x0884 to 0x08FF.

Note: Unused MB space must not be used as general purpose RAM while FlexCAN is transmitting and receiving CAN frames.

40.5 Initialization/application information

This section provide instructions for initializing the FlexCAN module.

40.5.1 FlexCAN initialization sequence

The FlexCAN module may be reset in two ways:

- MCU level hard reset, which resets all memory mapped registers asynchronously
- SOFT_RST bit in MCR, which has the same effect as the MCU level soft reset

Soft reset is synchronous and has to follow an internal request/acknowledge procedure across clock domains. Therefore, it may take some time to fully propagate its effects. The SOFT_RST bit remains asserted while soft reset is pending, so software can poll this bit to know when the reset has completed. Also, soft reset cannot be applied while clocks are shut down in any of the low power modes. The low power mode should be exited and the clocks resumed before applying soft reset.

The clock source (CLK_SRC bit) should be selected while the module is in Disable Mode. After the clock source is selected and the module is enabled (MDIS bit negated), FlexCAN automatically goes to Freeze Mode. In Freeze Mode, FlexCAN is unsynchronized to the CAN bus, the HALT and FRZ bits in the MCR are set, the internal state machines are disabled and the FRZ_ACK and NOT_RDY bits in the MCR are set. The Tx pin is in recessive state and FlexCAN does not initiate any transmission or reception of CAN frames. Note that the Message Buffers and the Rx Individual Mask Registers are not affected by reset, so they are not automatically initialized.

For any configuration change/initialization it is required that FlexCAN is put into Freeze Mode (see *Section 40.4.9.1: Freeze mode*). The following is a generic initialization sequence applicable to the FlexCAN module:

1. Initialize the Module Configuration Register
   a) Enable the individual filtering per MB and reception queue features by setting the BCC bit
   b) Enable the warning interrupts by setting the WRN_EN bit
   c) If required, disable frame self reception by setting the SRX_DIS bit
   d) Enable the FIFO by setting the FEN bit
   e) Enable the abort mechanism by setting the AEN bit
   f) Enable the local priority feature by setting the LPRIO_EN bit
2. Initialize the Control Register
   a) Determine the bit timing parameters: PROPSEG, PSEG1, PSEG2, RJW
   b) Determine the bit rate by programming the PRESDIV field
   c) Determine the internal arbitration mode (LBUF bit)
3. Initialize the Message Buffers
   a) The Control and Status word of all Message Buffers must be initialized
   b) If FIFO was enabled, the 8-entry ID table must be initialized
   c) Other entries in each Message Buffer should be initialized as required
4. Initialize the Rx Individual Mask Registers
5. Set required interrupt mask bits in the IMASK Registers (for all MB interrupts), in CTRL Register (for Bus Off and Error interrupts)
6. Negate the HALT bit in MCR

Starting with the last event, FlexCAN attempts to synchronize to the CAN bus.

41 Deserial Serial Peripheral Interface (DSPI)

41.1 Introduction

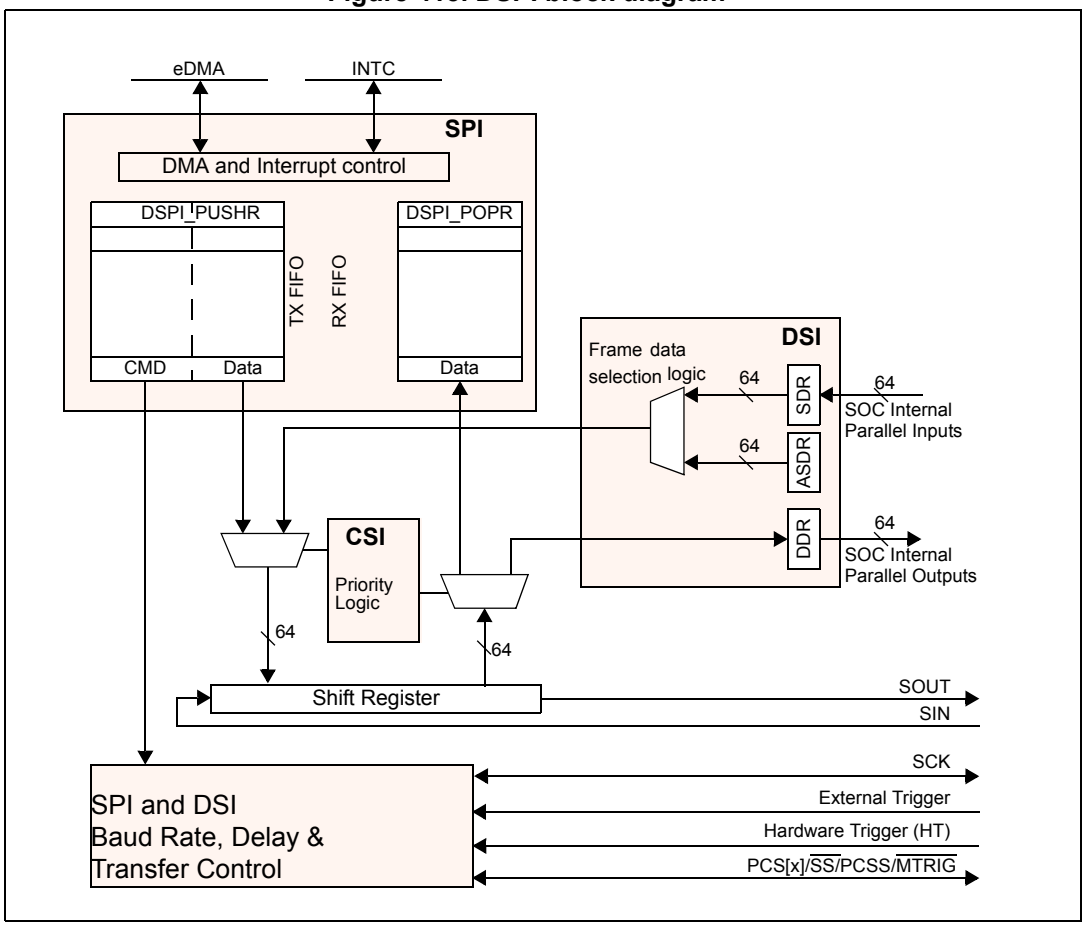
The Deserial Serial Peripheral Interface (DSPI) module provides a synchronous serial bus for communication between an MCU and an external peripheral device. The module supports MCU pin count reduction through serialization and deserialization of MCU internal signals transmitted over the SPI serial link.

Note: For the chip-specific implementation details of this module's instances, see Chapter 7: Device configuration: Section 7.8.2: DSPI configuration.

41.1.1 Block diagram

Figure 415 provides the block diagram of this module.

Figure 415. DSPI block diagram (1)



1. CSI and DSI features are not supported

41.1.2 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.

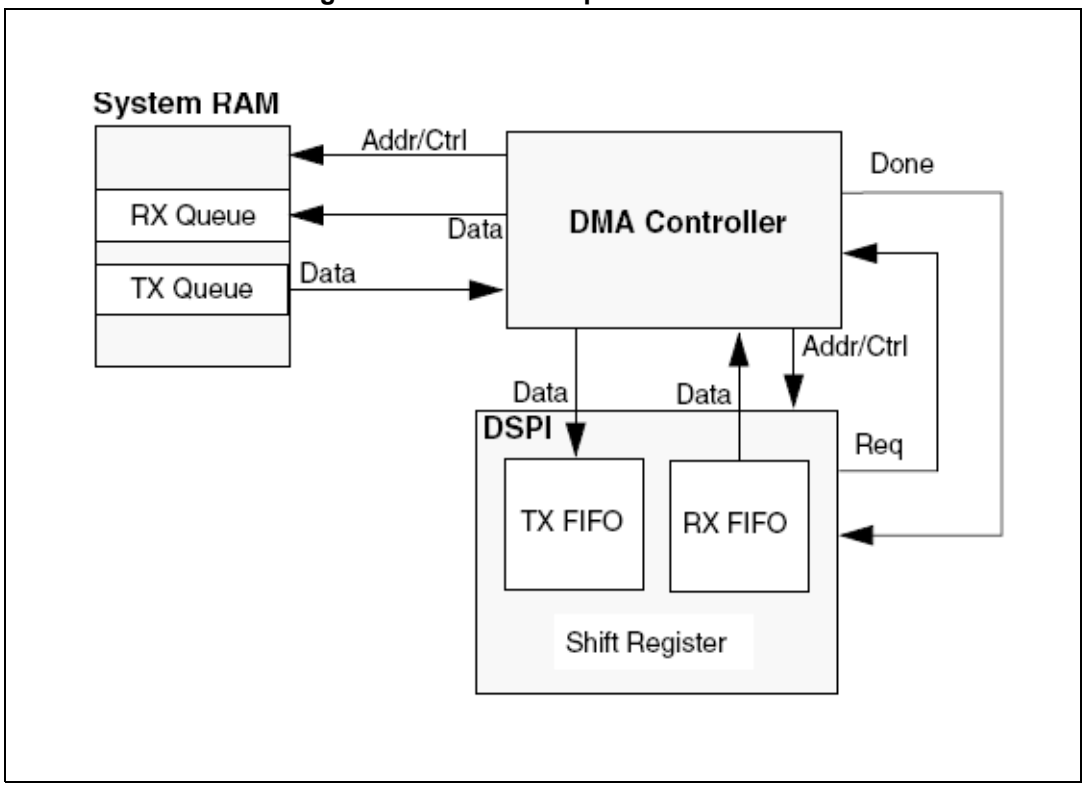
- two sources of serialized data:
  - DSPI memory-mapped register
  - parallel input signals
  - programmable selection of source data on bit basis
- deserialized data provided as Parallel Output signals and as bits in a memory-mapped register
- interrupt conditions:
  - deserialized data matches pre-programmed 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

41.1.3 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 416* shows a system example with DMA, DSPI and external queues in system RAM.

Figure 416. DSPI with queues and DMA



41.1.4 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.

41.1.4.1 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.

41.1.4.2 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.

41.1.4.3 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.

41.1.4.4 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.

41.1.4.5 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.

41.2 DSPI signal description

This section provides the DSPI signals description.

Table 494 lists the signals that may connect off chip depending on device implementation.

Table 494. 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

41.2.1 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.

41.2.2 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.

41.2.3 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.

41.2.4 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.

41.2.5 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.

41.2.6 SIN — Serial Input

SIN is a serial data input signal.

41.2.7 SOUT — Serial Output

SOUT is a serial data output signal.

41.2.8 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.

41.2.9 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 7.8.2: DSPI configuration.

Table 495. DSPI memory map

Offset (hex)	Register name	Location
0x0000	DSPI Module Configuration Register (DSPI*x*_MCR)	*41.2.9.1 on page 864*
0x0004-0x0007	Reserved	
0x0008	DSPI Transfer Count Register (DSPI*x*_TCR)	*41.2.9.2 on page 868*
0x000C	DSPI Clock and Transfer Attributes Register 0 (In Master Mode) (DSPI*x*_CTAR0)	*41.2.9.3 on page 869*
0x000C	DSPI Clock and Transfer Attributes Register 0 (In Slave Mode) (DSPI*x*_CTAR0_SLAVE)	*41.2.9.4 on page 874*
0x0010	DSPI Clock and Transfer Attributes Register 1 (In Master Mode) (DSPI*x*_CTAR1)	*41.2.9.3 on page 869*
0x0010	DSPI Clock and Transfer Attributes Register 1 (In Slave Mode) (DSPI*x*_CTAR1_SLAVE)	*41.2.9.4 on page 874*
0x0014	DSPI Clock and Transfer Attributes Register 2 (In Master Mode) (DSPI*x*_CTAR2)	*41.2.9.3 on page 869*
0x0018	DSPI Clock and Transfer Attributes Register 3 (In Master Mode) (DSPI*x*_CTAR3)	*41.2.9.3 on page 869*
0x001C	DSPI Clock and Transfer Attributes Register 4 (In Master Mode) (DSPI*x*_CTAR4)	*41.2.9.3 on page 869*
0x0020	DSPI Clock and Transfer Attributes Register 5 (In Master Mode) (DSPI*x*_CTAR5)	*41.2.9.3 on page 869*
0x0024	DSPI Clock and Transfer Attributes Register 6 (In Master Mode) (DSPI*x*_CTAR6)	*41.2.9.3 on page 869*
0x0028	DSPI Clock and Transfer Attributes Register 7 (In Master Mode) (DSPI*x*_CTAR7)	*41.2.9.3 on page 869*
0x002C	DSPI Status Register (DSPI*x*_SR)	*41.2.9.6 on page 876*
0x0030	DSPI DMA/Interrupt Request Select and Enable Register (DSPI*x*_RSER)	*41.2.9.7 on page 878*
0x0034	DSPI PUSH FIFO Register In Master Mode (DSPI*x*_PUSHR)	*41.2.9.8 on page 881*
0x0034	DSPI PUSH FIFO Register In Slave Mode (DSPI*x*_PUSHR_SLAVE)	*41.2.9.9 on page 884*
0x0038	DSPI POP FIFO Register (DSPI*x*_POPR)	*41.2.9.10 on page 884*
TX FIFO registers (n=0 to 3)		
0x003C + n*0x4	DSPI Transmit FIFO Register n (DSPI*x*_TXFR*n*)	*41.2.9.11 on page 885*
0x004C–0x007B	Reserved	
RX FIFO registers (n=0 to 3)		

Table 495. DSPI memory map(Continued)

Offset (hex)	Register name	Location
0x007C + n*0x4	DSPI Receive FIFO Register n (DSPI*x*_RXFR*n*)	*41.2.9.12 on page 886*
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*)	*41.2.9.13 on page 886*
0x013C	DSPI Status Register Extended (DSPI*x*_SREX)	*41.2.9.14 on page 887*

41.2.9.1 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 417. DSPI Module Configuration Register (DSPI*x*_MCR)

Table 496. 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.

Table 496. DSPI*x*_MCR field descriptions(Continued)

Field	Description
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.
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.

Table 496. DSPI*x*_MCR field descriptions(Continued)

Field	Description
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.
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.

Table 496. DSPI*x*_MCR field descriptions(Continued)

Field	Description
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.
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

Table 496. DSPI*x*_MCR field descriptions(Continued)

Field	Description
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.

41.2.9.2 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 W	SPI_TCNT															
Reset	0	0	0	0	0	0	0	0	0	0	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 418. DSPI Transfer Count Register (DSPI*x*_TCR)

Table 497. 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.

41.2.9.3 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 419. DSPI Clock and Transfer Attributes Register (In Master Mode) (DSPI*x*_CTAR*n*)

Table 498. 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 499*. See the BR field description for details on how to compute the baud rate. Refer to *Table 499* 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.
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.

Table 498. DSPI*x*_CTAR*n* field descriptions(Continued)

Field	Description
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 41.4.3.2: PCS to SCK delay (tCSC)* 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 41.4.3.4: Delay after transfer (tDT)* 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.
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.

Table 498. DSPI*x*_CTAR*n* field descriptions(Continued)

Field	Description
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 41.4.3.2: PCS to SCK delay (tCSC)* 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 41.4.3.3: After SCK delay (tASC)* for more details.

Table 498. 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 499. 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

41.2.9.4 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 420. DSPI Clock and Transfer Attributes Register (In Slave Mode) (DSPI*x*_CTAR0_SLAVE)

Table 500. 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.

41.2.9.5 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 421. DSPI Clock and Transfer Attributes Register (In Slave Mode) (DSPI*x*_CTAR1_SLAVE)

Table 501. 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.

41.2.9.6 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 422. DSPI Status Register (DSPI*x*_SR)

Table 502. 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 502. 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 41.4.2.5: 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 502. 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.

41.2.9.7 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

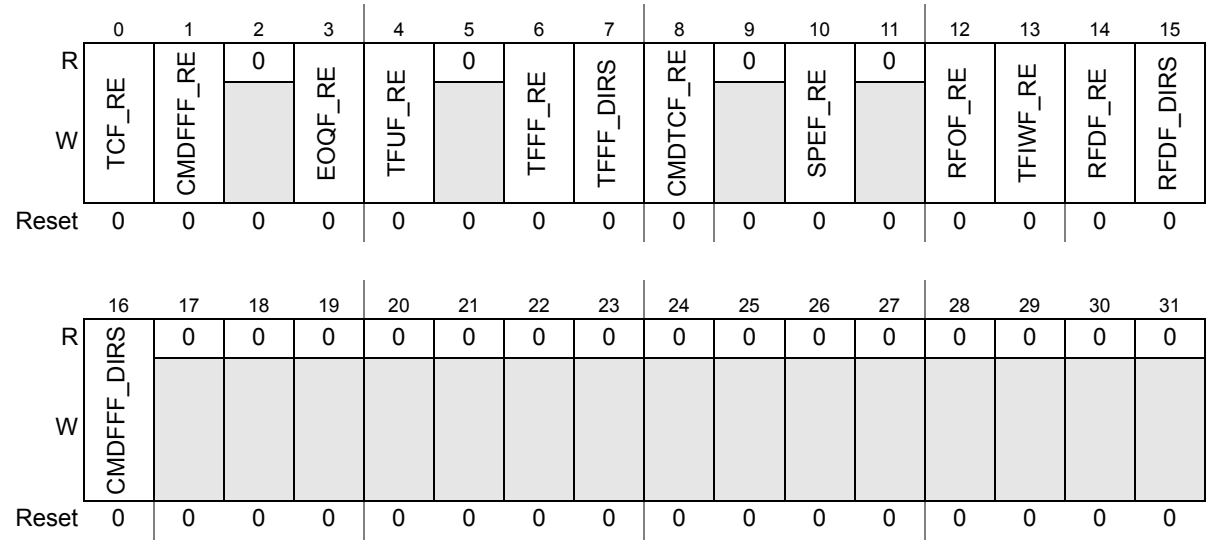


Figure 423. DSPI DMA/Interrupt Request Select and Enable Register (DSPI*x*_RSER)

Table 503. 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 503. 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 503. 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.

41.2.9.8 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 424. DSPI PUSH FIFO Register In Master Mode (DSPI*x*_PUSHR)

Table 504. 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 41.4.4.6: 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 504. 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 41.4.4.6: 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 504. 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.

41.2.9.9 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 W	TXDATA															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 425. DSPI PUSH FIFO Register In Slave Mode (DSPI*x*_PUSHR_SLAVE)

Table 505. 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.

41.2.9.10 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 426. DSPI POP FIFO Register (DSPI*x*_POPR)

Table 506. 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.

41.2.9.11 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 427. DSPI Transmit FIFO Registers (DSPI*x*_TXFRn)

Table 507. 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.

41.2.9.12 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 428. DSPI Receive FIFO Registers (DSPI*x*_RXFRn)

Table 508. DSPI*x*_RXFRn field descriptions

Field	Description
0:31 RXDATA	Receive Data Contains the received SPI data.

41.2.9.13 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 429. DSPI Clock and Transfer Attributes Register Extended (DSPI*x*_CTARE*n*)

Table 509. 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’.

41.2.9.14 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 430. DSPI Status Register Extended (DSPI*x*_SREX)

Table 510. 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.

41.3 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.

41.4 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 41.2.9.1: DSPI Module Configuration Register (DSPIx_MCR)* for the DSPI configuration values.

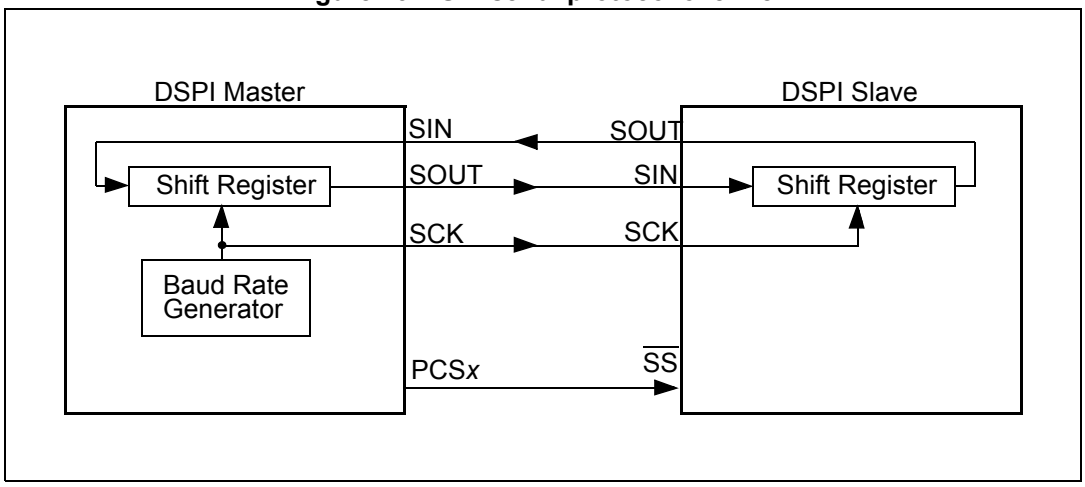
The CTAR*n* registers hold clock and transfer attributes. The SPI configuration allows to select which CTAR to use on a frame by frame basis by setting a field in the SPI command. The Extended SPI Mode (DSPI*x*_MCR[XSPI]) further allows the usage of CTARE*n* (CTAR*n* Extended) registers which allows 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 431*. 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 431. 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.

41.4.1 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.

41.4.2 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 41.4.2.4: Transmit First In First Out (TX FIFO) buffering mechanism*, *Section 41.4.2.5: Command First In First Out (CMD FIFO) buffering mechanism* and *Section 41.4.2.6: Receive First In First Out (RX FIFO) Buffering Mechanism*. The interrupt and DMA request conditions are described in *Section 41.4.8*.

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.

41.4.2.1 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.

41.4.2.2 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.

41.4.2.3 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.

41.4.2.4 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.

41.4.2.4.1 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 41.4.8.2: 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.

41.4.2.4.2 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 41.4.8.7: Transmit FIFO Underflow interrupt request* for details.

41.4.2.5 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.

41.4.2.6 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.

41.4.2.6.1 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.

41.4.2.6.2 Draining the RX FIFO

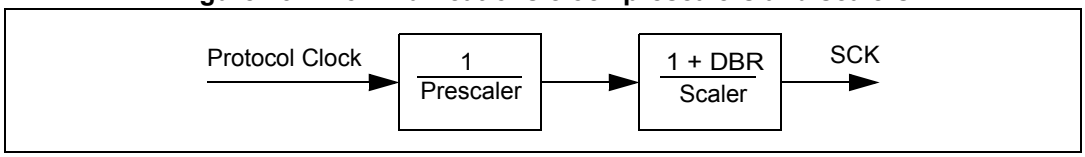
Host CPU or a DMA can remove (pop) entries from the RX FIFO by reading the DSPI POP FIFO Register (POPR). A read of the POPR decrements the RX FIFO Counter by one. Attempts to pop data from an empty RX FIFO are ignored and the RX FIFO Counter remains unchanged. The data, read from the empty RX FIFO, 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.

41.4.3 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 432* shows conceptually how the SCK signal is generated.

Figure 432. Communications clock prescalers and scalers



41.4.3.1 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 511* shows an example of how to compute the baud rate.

Table 511. 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 511 are given as an example. Refer to the clocking chapter for the frequency used to drive this module in the device.

41.4.3.2 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 512* 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 512* shows an example of how to compute the PCS to SCK delay.

Table 512. 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 512 is given as an example. Refer to the clocking chapter for the frequency used to drive this module in the device.

41.4.3.3 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 513* 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 513* shows an example of how to compute the After SCK delay.

Table 513. 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 513 is given as an example. Refer to the clocking chapter for the frequency used to drive this module in the device.

41.4.3.4 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 514* 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 514* shows an example of how to compute the delay after transfer.

Table 514. Delay after transfer computation example

f_p	PDT	Prescaler	DT	Scaler	Delay after transfer
100 MHz	0b01	3	0b1110	32768	0.98 ms

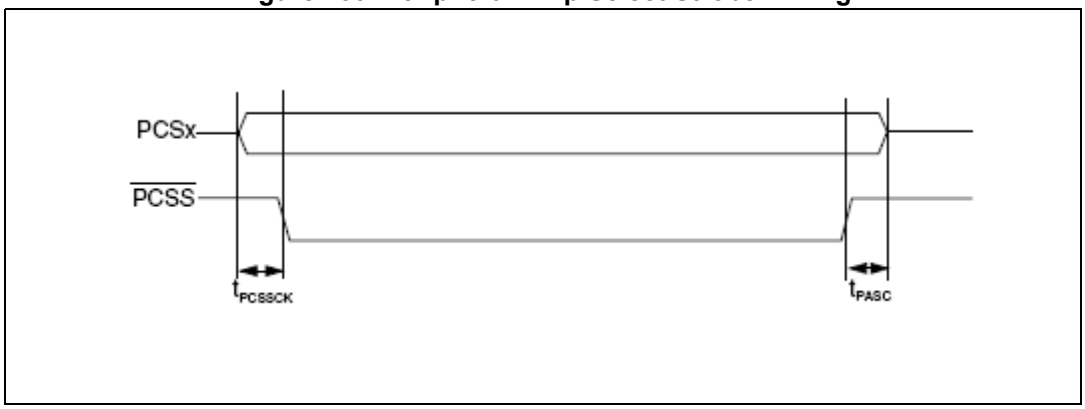
Note: The clock frequency mentioned in Table 514 is given as an example. Refer to the clocking chapter for the frequency used to drive this module in the device.

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.

41.4.3.5 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 433* shows the timing of the $\overline{PCSS}$ signal relative to PCS signals.

Figure 433. 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 33

$$t_{PCSSCK} = \frac{1}{f_P} \times PCSSCK$$

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 34

$$t_{PASC} = \frac{1}{f_P} \times PASC$$

Table 515 shows an example of how to compute the t_{PCSSCK} delay.

Table 515. Peripheral Chip Select Strobe Assert Computation Example

f_p	PCSSCK	Prescaler	Delay before transfer
100 MHz	0b11	7	70.0 ns

Table 516 shows an example of how to compute the t_{PASC} delay.

Table 516. 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 515 and Table 516 is given as an example. Refer to the clocking chapter for the frequency used to drive this module in the device.

41.4.4 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

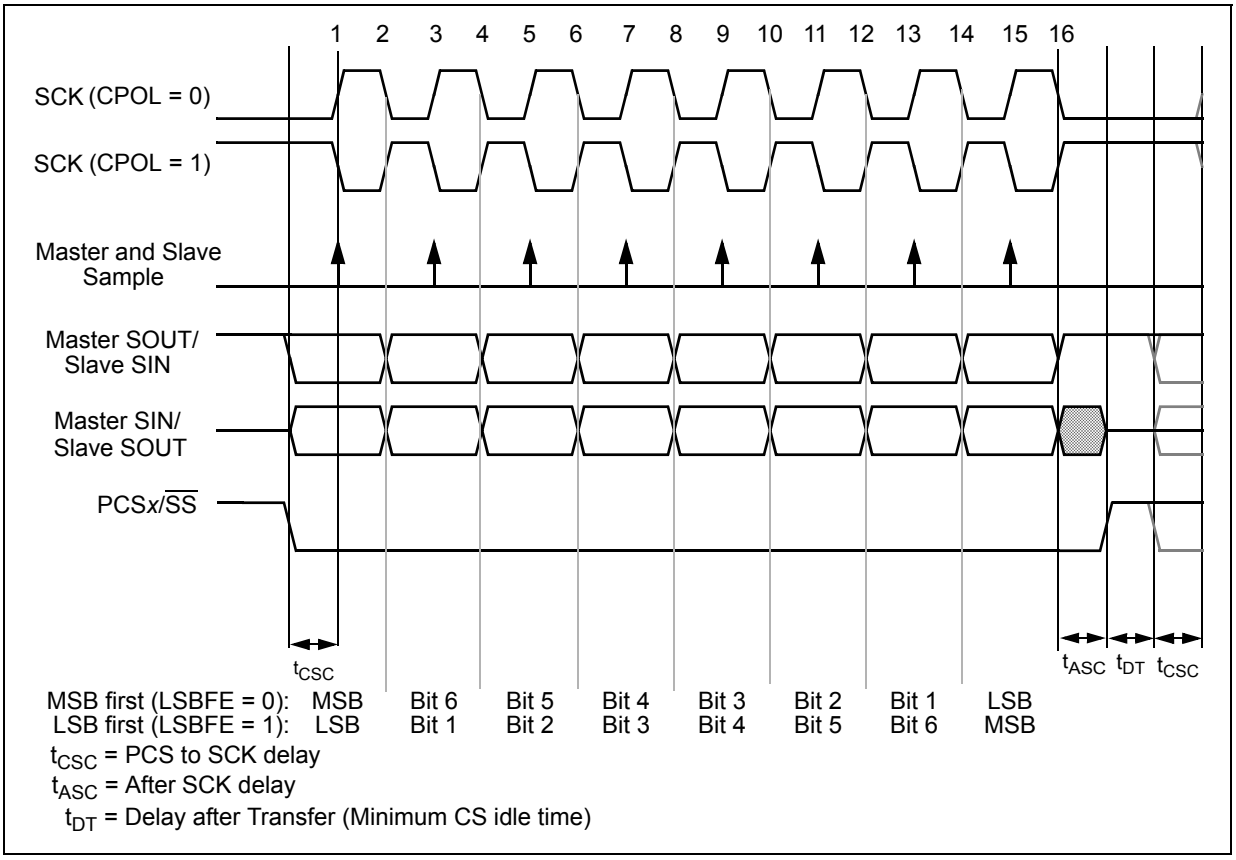
A modified transfer format is supported to allow for high-speed communication with peripherals that require longer setup times. The DSPI can sample the incoming data later than halfway through the cycle to give the peripheral more setup time. The MTFE bit in the MCR selects between Classic SPI Format and Modified Transfer Format.

In the SPI configuration, the DSPI provides the option of keeping the PCS signals asserted between frames. See *Section 41.4.4.5: Continuous Selection Format* for details.

41.4.4.1 Classic SPI Transfer Format (CPHA = 0)

The transfer format shown in *Figure 434* 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 434. DSPI transfer timing diagram (MTFE = 0, CPHA = 0, FMSZ = 8)

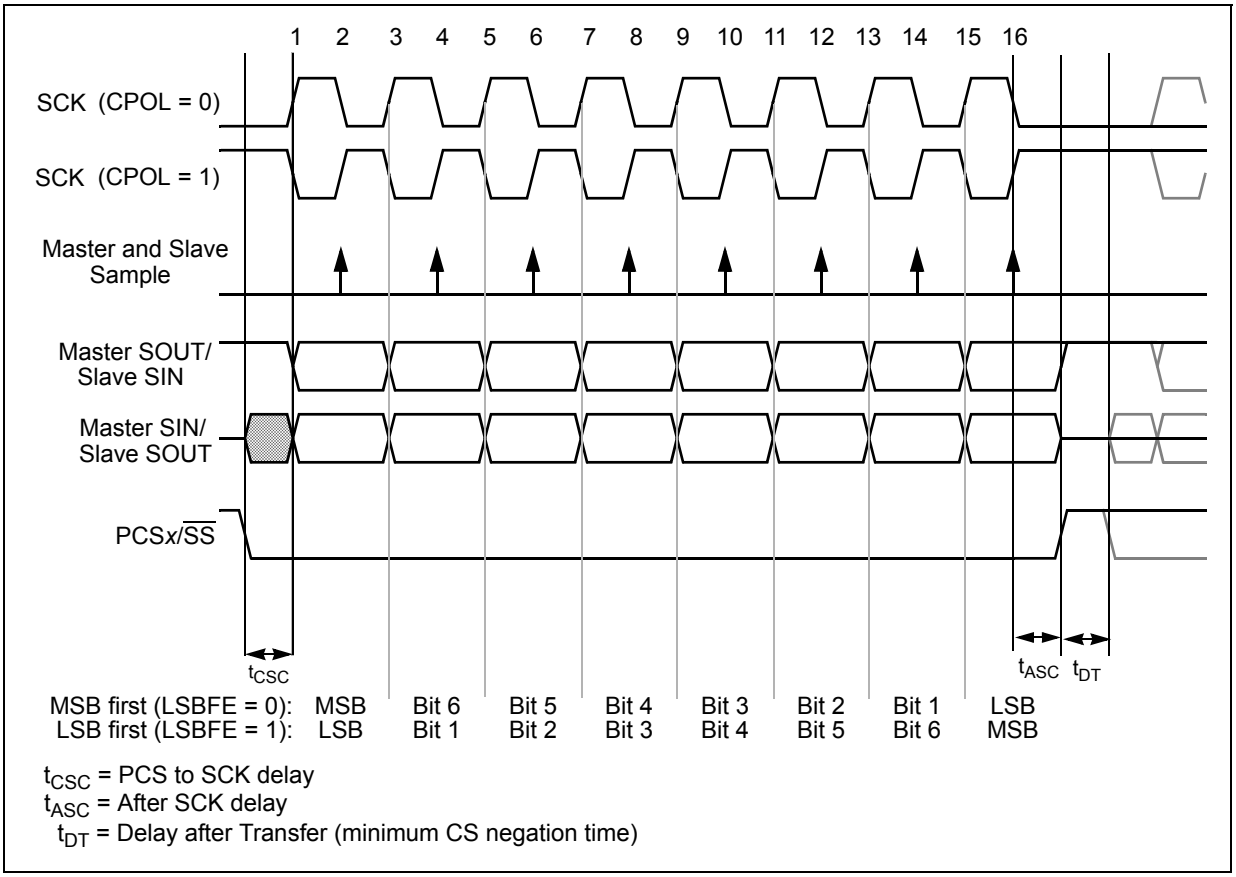


The master initiates the transfer by placing its first data bit on the SOUT pin and asserting the appropriate peripheral chip select signals to the slave device. The slave responds by placing its first data bit on its SOUT pin. After the t_{CSC} delay elapses, the master outputs the first edge of SCK. 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.

41.4.4.2 Classic SPI Transfer Format (CPHA = 1)

This transfer format shown in *Figure 435* 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 435. 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.

41.4.4.3 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 436 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 436. DSPI Modified Transfer Format (MTFE = 1, CPHA = 0, $f_{sck} = f_p/4$)

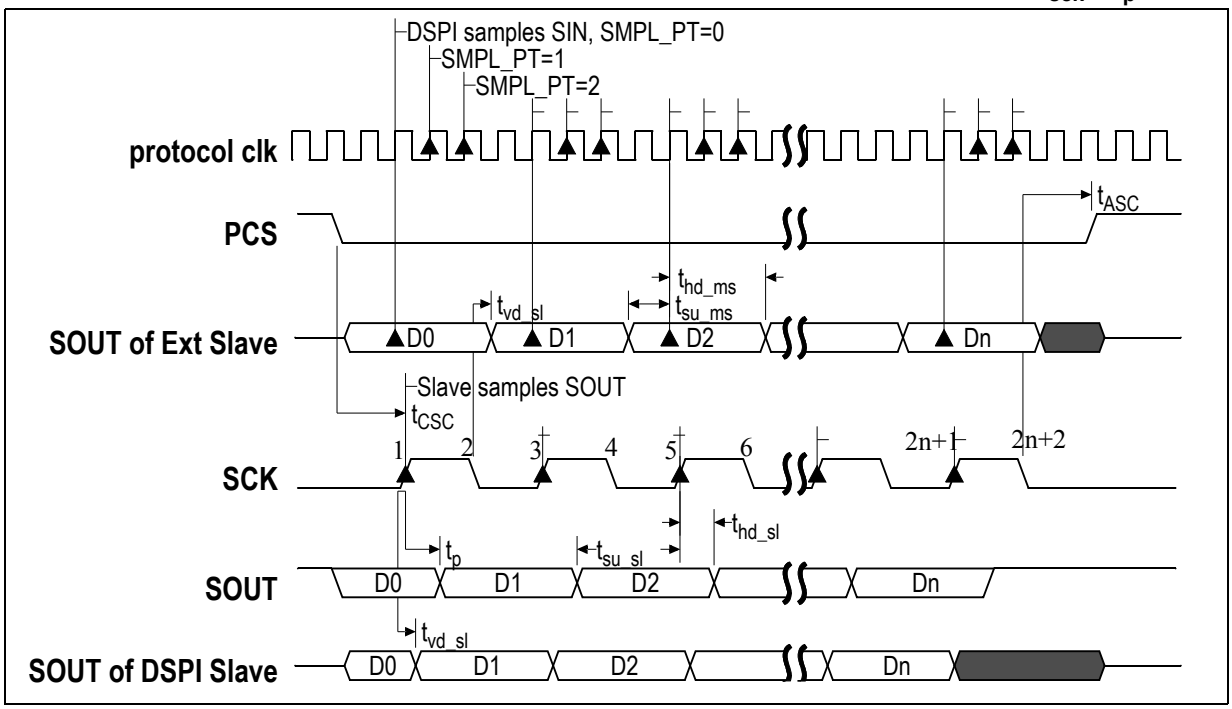


Figure 437. DSPI Modified Transfer Format (MTFE = 1, CPHA = 0, $f_{sck} = f_p/2$)

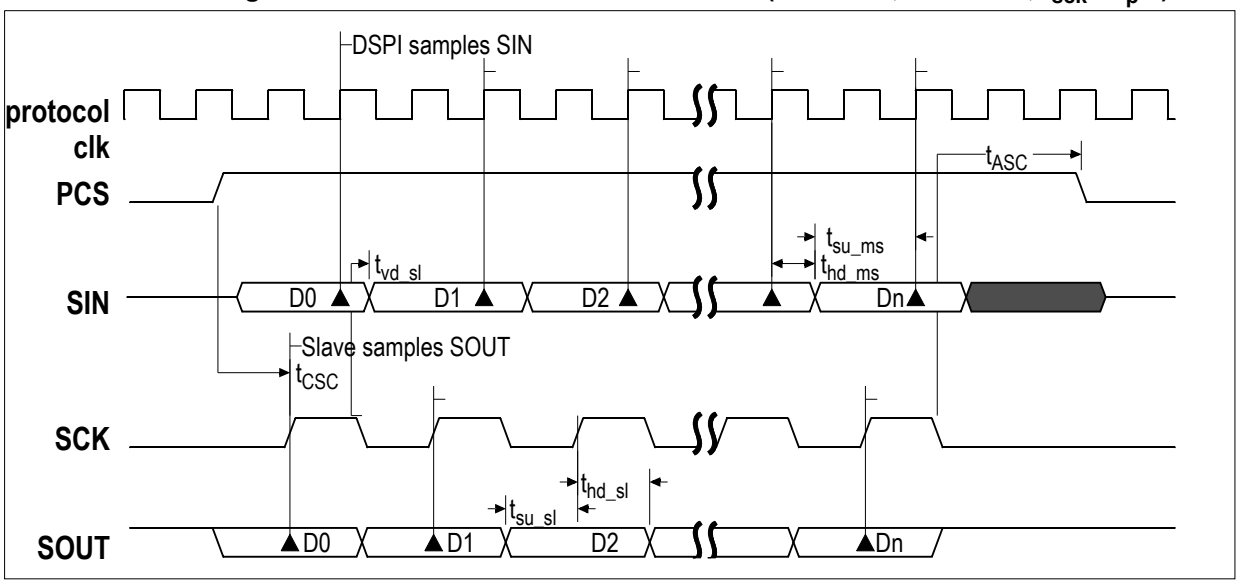
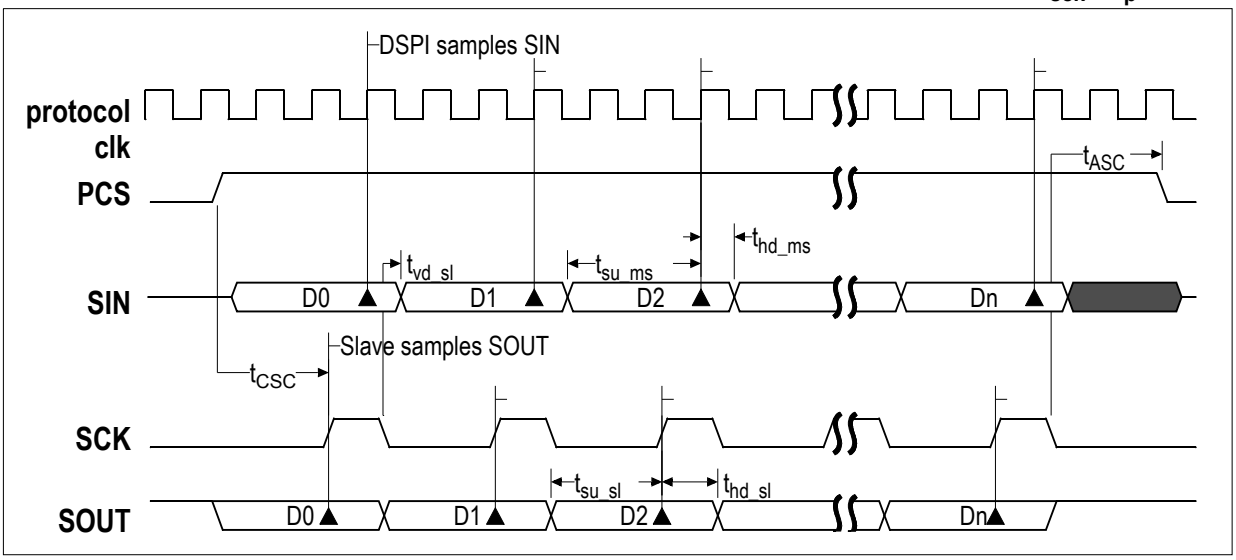


Figure 438. DSPI Modified Transfer Format (MTFE = 1, CPHA = 0, $f_{sck} = f_p/3$)



41.4.4.4 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 439. DSPI Modified Transfer Format (MTFE = 1, CPHA = 1, $f_{sck} = f_p/2$)

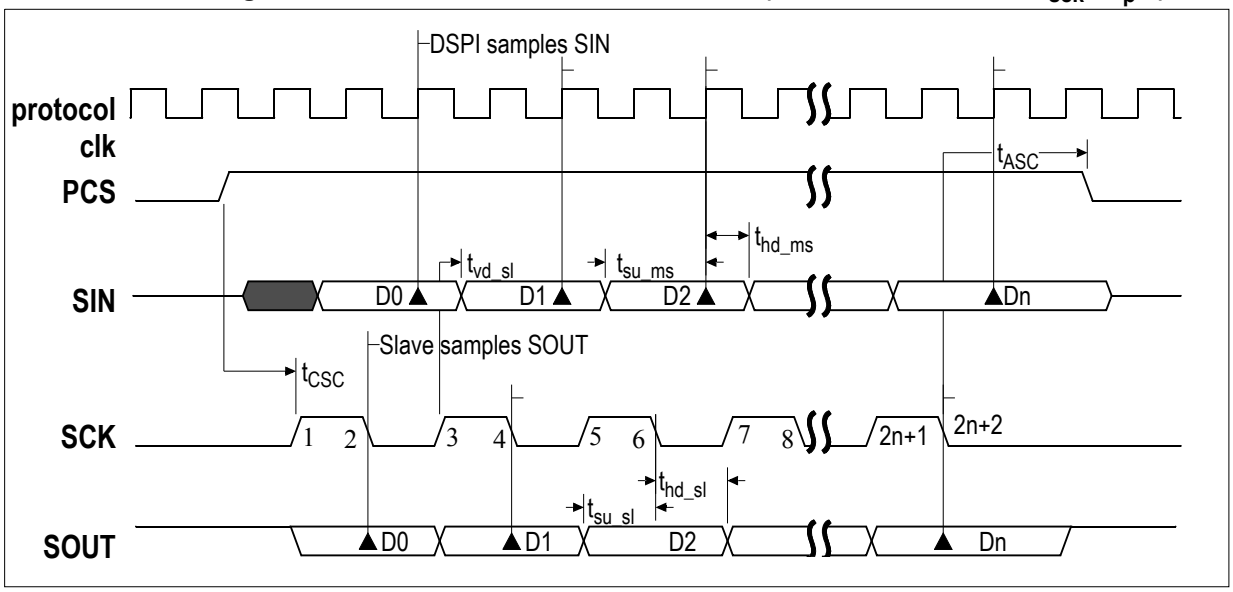


Figure 440. DSPI Modified Transfer Format (MTFE = 1, CPHA = 1, $f_{sck} = f_p/3$)

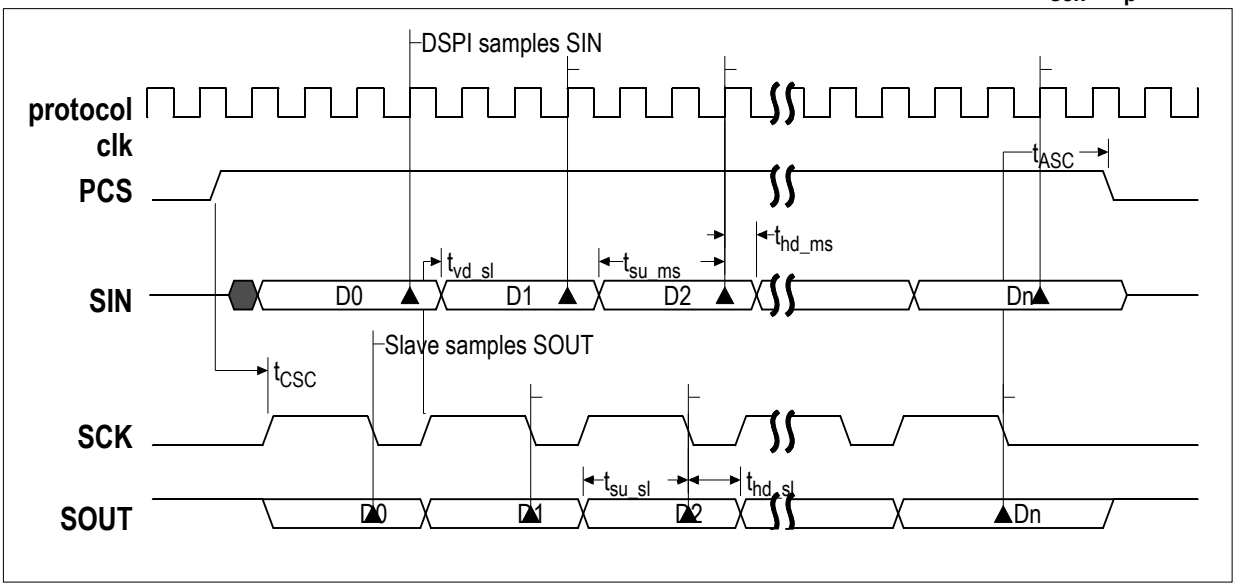
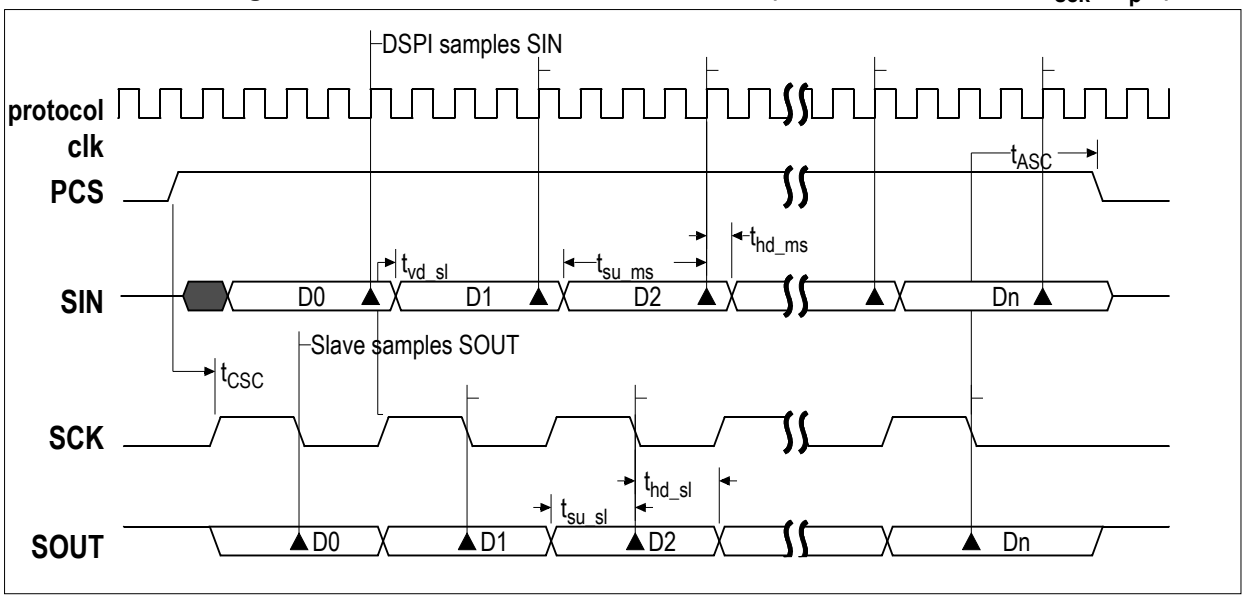


Figure 441. DSPI Modified Transfer Format (MTFE = 1, CPHA = 1, $f_{sck} = f_p/4$)

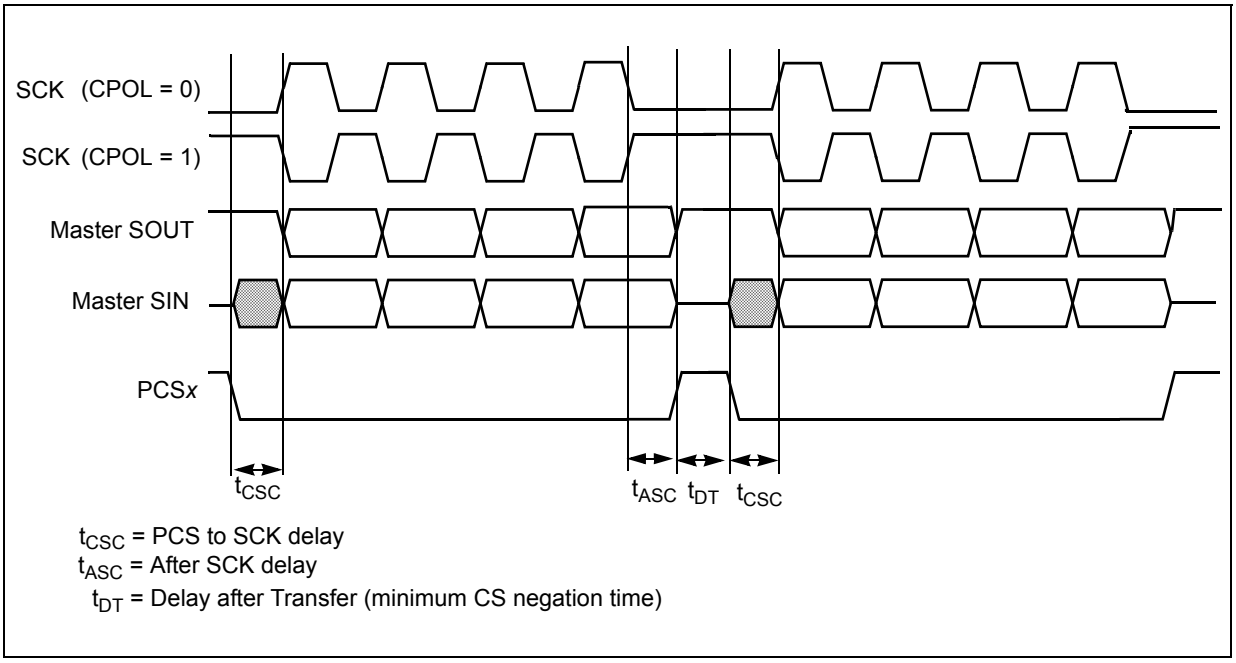


41.4.4.5 Continuous Selection Format

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.

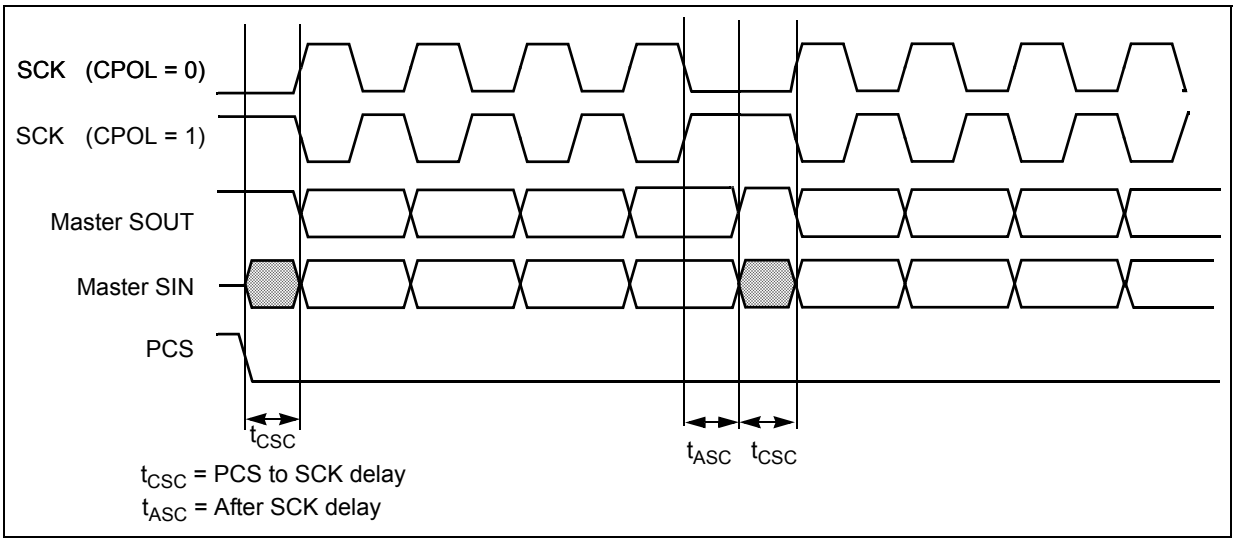
The behavior of the PCS signals in the two configurations is identical so only SPI configuration will be described. When the CONT bit = 0, the DSPI drives the asserted Chip Select signals to their idle states in between frames. The idle states of the Chip Select signals are selected by the PCSISn bits in the MCR. The timing diagram in *Figure 442* is for two four-bit transfers with CPHA = 1 and CONT = 0.

Figure 442. 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 443* shows the timing diagram for two four-bit transfers with CPHA = 1 and CONT = 1.

Figure 443. 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.

41.4.4.6 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.

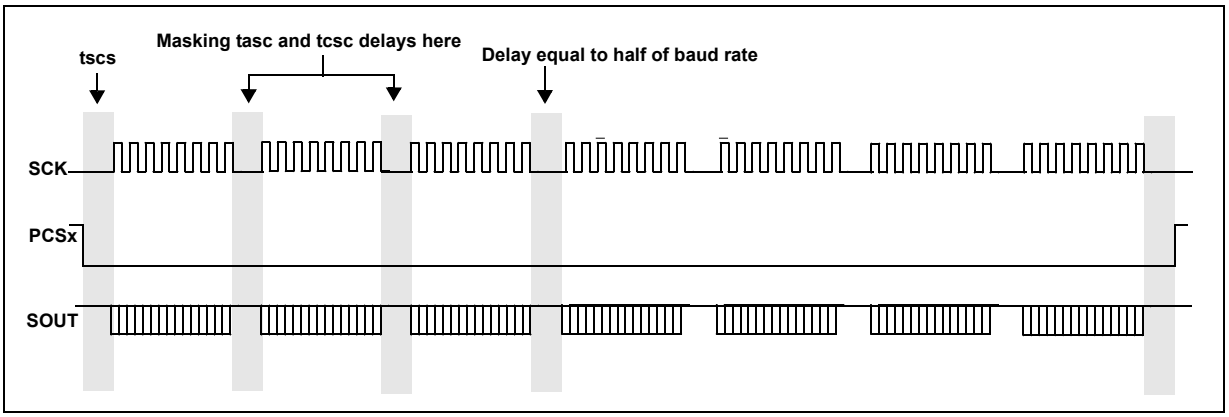
The Fast Continuous Selection Format is enabled by writing ‘1’ to bit MCR[FCPCS]. When this bit is asserted, MASC and MCSC bits of the PUSHR perform the function of mask bits for the transmit frame. These bits individually mask the t_{ASC} and t_{CSC} delays as programmed by the user software. A normal 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 444 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 444. 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.

41.4.5 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.

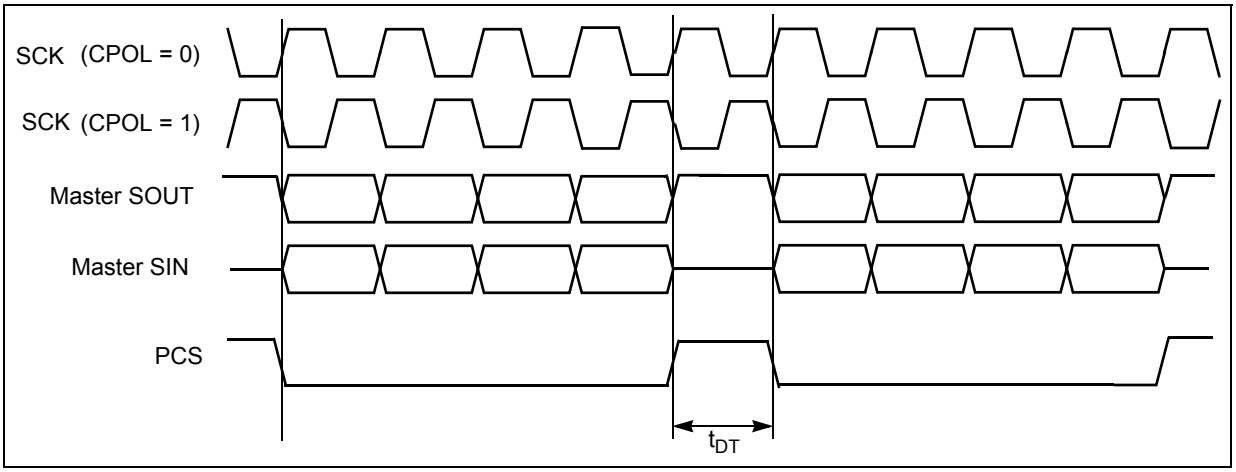
It is recommended to keep the baud rate the same while using the Continuous SCK. Switching clock polarity between frames while using Continuous SCK can cause errors in the transfer. Continuous SCK operation is not guaranteed if the DSPI is put into the External Stop mode or Module Disable mode.

Enabling Continuous SCK disables the PCS to SCK delay and the Delay after Transfer (t_{DT}) is fixed to one SCK cycle.

Figure 445 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 445. Continuous SCK timing diagram (CONT = 0)

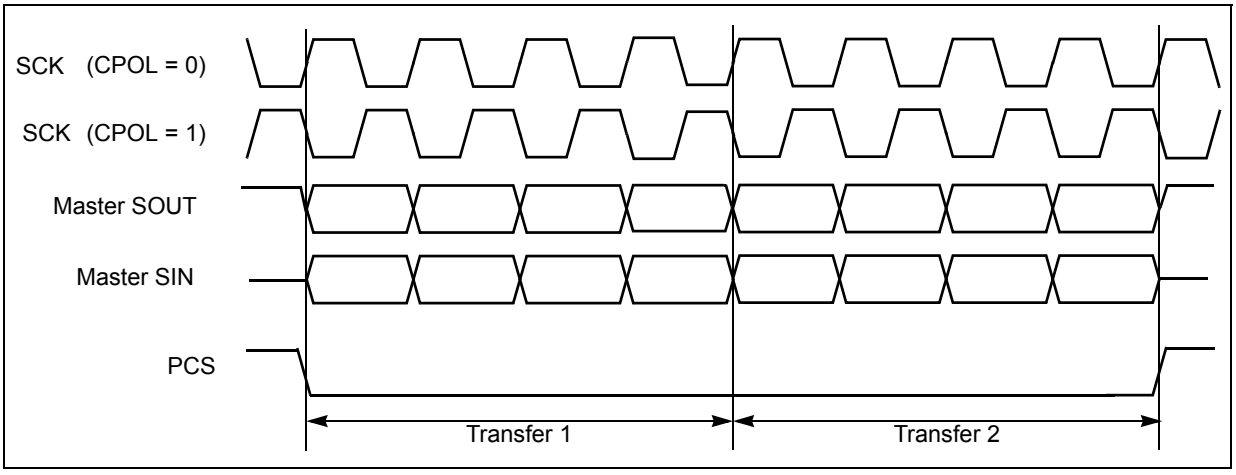


If the CONT bit in the TX FIFO entry is set, PCS remains asserted between the transfers. Under certain conditions, SCK can continue with PCS asserted, but with no data being shifted out of SOUT (SOUT pulled high). This can cause the slave to receive incorrect data. Those conditions include:

- 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 41.4.1*.
- Continuous SCK with CONT bit set and entering Stop mode or Module Disable mode.

Figure 446 shows the timing diagram for Continuous SCK format with Continuous Selection enabled.

Figure 446. Continuous SCK Timing Diagram (CONT = 1)



41.4.6 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.

41.4.7 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.

41.4.7.1 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.

41.4.8 Interrupts/DMA requests

Certain DSPI conditions can generate only interrupt requests and other DSPI conditions can generate either interrupt requests or DMA requests. *Table 517* lists these conditions.

Table 517. 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.

41.4.8.1 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.

41.4.8.2 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.*

41.4.8.3 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.*

41.4.8.4 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.

41.4.8.5 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.

41.4.8.6 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.

41.4.8.7 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.

41.4.8.8 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.

41.4.8.9 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.

41.4.8.10 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.

41.4.9 Power saving features

The DSPI supports following power-saving strategies:

- External Stop mode
- Module Disable mode – Clock gating of non-memory-mapped logic

41.4.9.1 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.

41.4.9.2 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.

41.5 Initialization/application information

This section describes how to initialize the DSPI module.

41.5.1 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.

41.5.2 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].

41.5.3 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.

41.5.4 Baud rate settings

Table 518 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 clocking chapter for the frequency used to drive this module in the device.

Table 518. Baud rate values (bps)

		Baud rate divider prescaler values (DSPI*n*_CTAR[PBR])			
		2	**3**	**5**	**7**
Baud Rate scaler values (DSPI*n*_CTAR[BR])	**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

41.5.5 Delay settings

Table 519 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 clocking chapter for the frequency used to drive this module in the device.

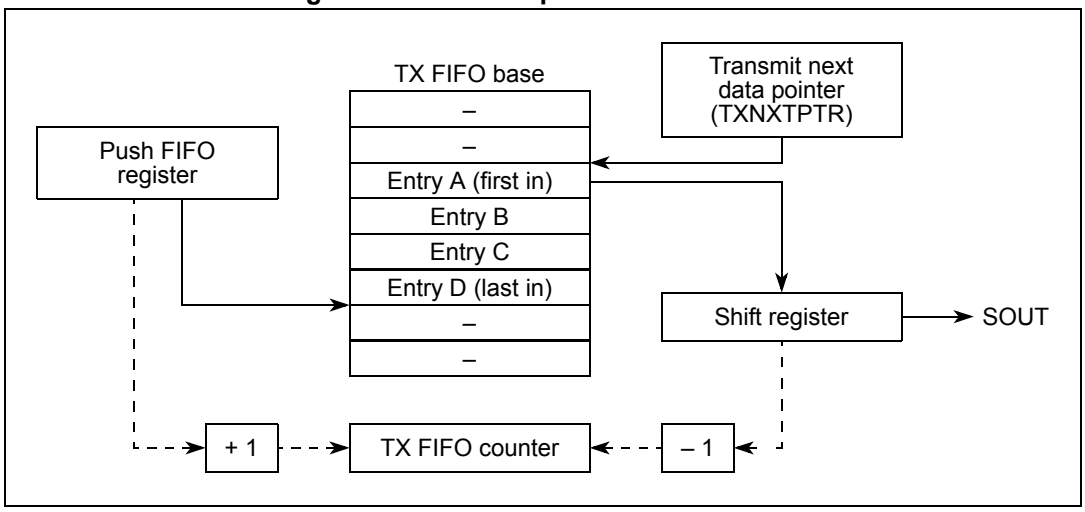
Table 519. Delay values

		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

41.5.6 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 447* 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 41.4.2.4: Transmit First In First Out (TX FIFO) buffering mechanism*, *Section 41.4.2.5: Command First In First Out (CMD FIFO) buffering mechanism* and *Section 41.4.2.6: Receive First In First Out (RX FIFO) Buffering Mechanism* for details on the FIFO operation.

Figure 447. TX FIFO pointers and counter



41.5.6.1 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 35
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 36
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

41.5.6.2 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 37
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 38
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

41.5.6.3 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 39
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 40
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

42 LINFlexD

42.1 Introduction

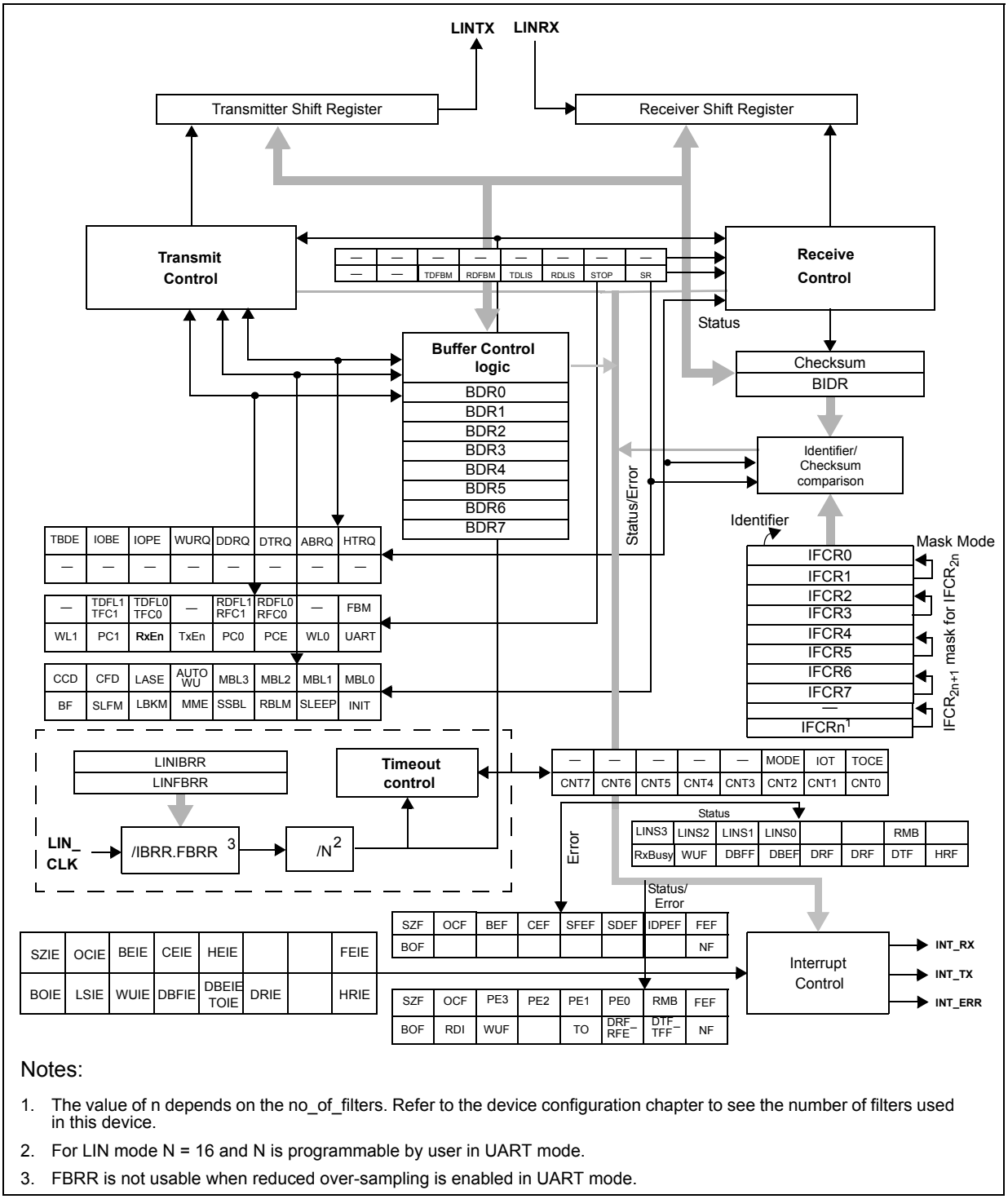
The LINFlexD controller is designed to manage a high number of LIN messages efficiently with a minimum of CPU load. To reduce the CPU loading in Master mode, LINFlexD autonomously handles the LIN messages once software has triggered the header transmission, until the next header transmission request 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 448. Block diagram



Notes:

1. The value of n depends on the no_of_filters. Refer to the device configuration chapter 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.

42.2 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

42.2.1 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 ≥ PBRIDGEx_CLK > 1/3*LIN_CLK” should be maintained)

42.2.2 UART mode features

- Full-duplex communication
- Baud rate is a function of baud clock, LINIBRR and LINFBRR registers — see *Section 42.3.2.8*
- 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

42.2.3 Standard features

DMA2x bus interface

42.3 Functional description

42.3.1 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.

42.3.1.1 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 449. Frames

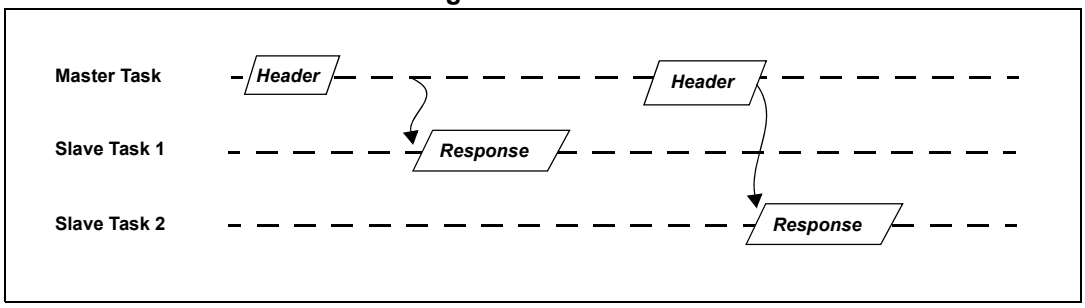
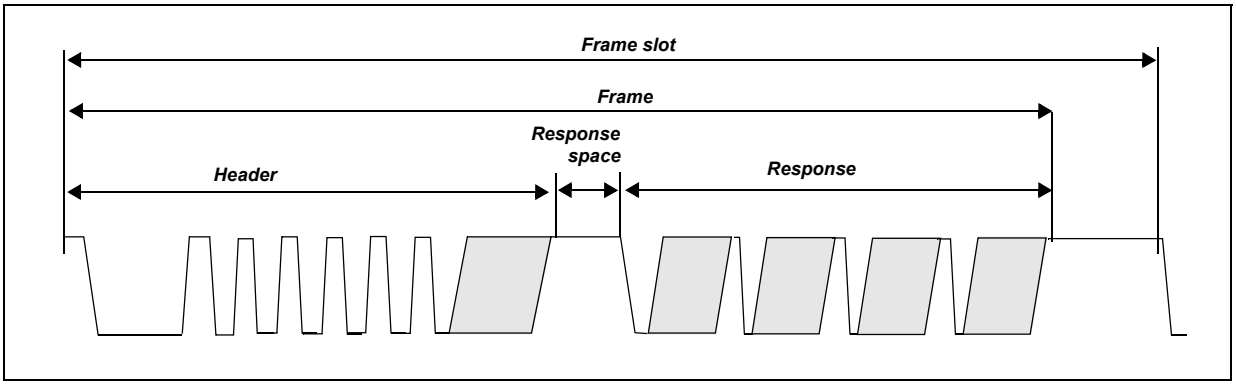


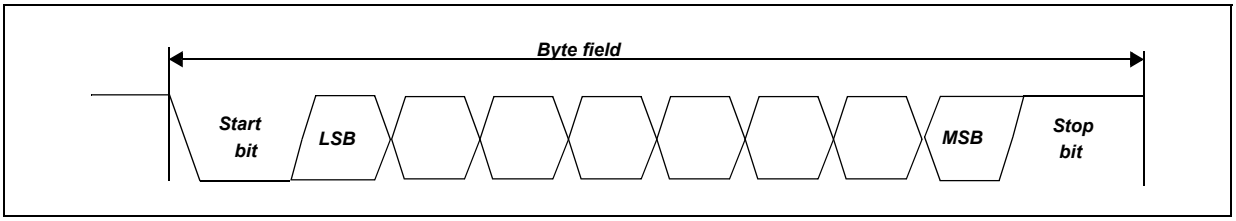
Figure 450. Structure of LIN frame



42.3.1.2 Data field

Each byte is transmitted as shown in *Figure 451*. 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).

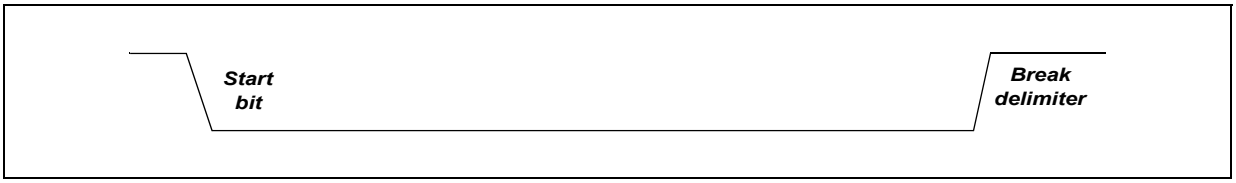
Figure 451. Structure of byte field



42.3.1.3 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 452*. The break delimiter must be of two-bit duration (to be compliant with LIN protocol 2.1).

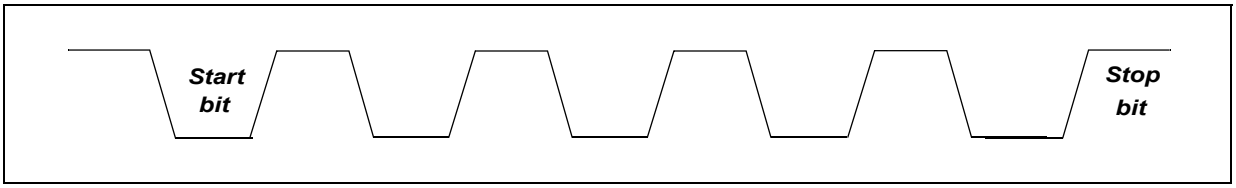
Figure 452. Break field



42.3.1.4 Sync byte

Sync is a byte field with the data value of 0x55.

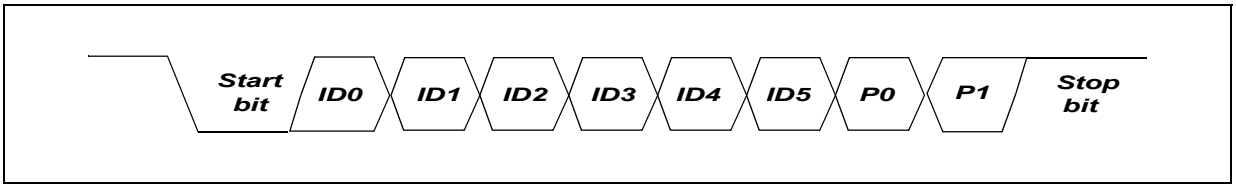
Figure 453. Sync byte field



42.3.1.5 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 454. Identifier field



P0 = ID0 XOR ID1 XOR ID2 XOR ID4

P1 = NOT (ID1 XOR ID3 XOR ID4 XOR ID5)

42.3.1.6 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.

42.3.2 LINFlexD features

42.3.2.1 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.

42.3.2.1.1 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)

42.3.2.1.2 Normal mode (NM)

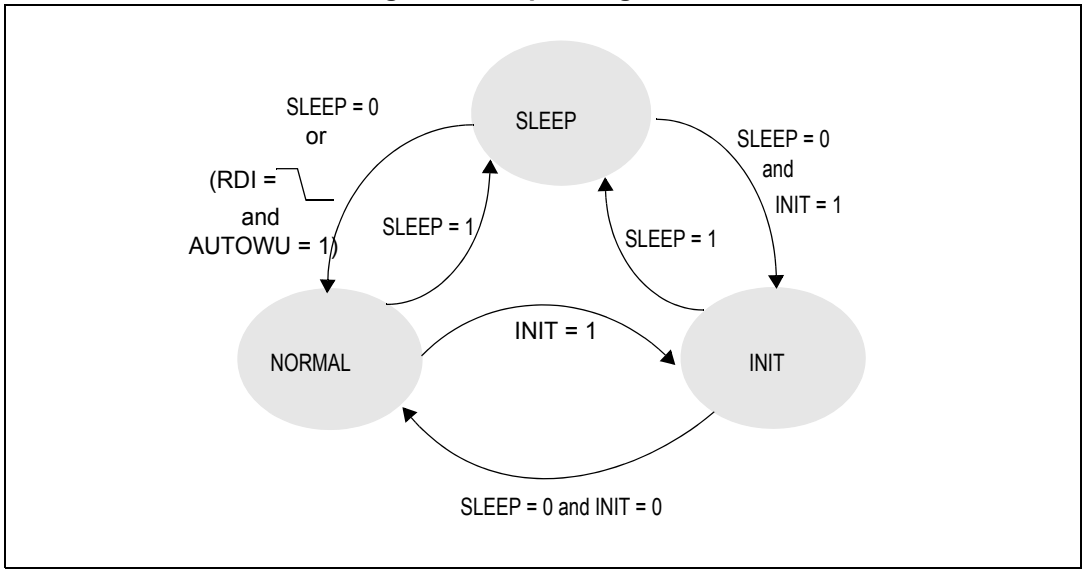
Once software has completed initialization of the LINFlexD controller, it can enter Normal mode by clearing the INIT bit.

42.3.2.1.3 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 42.3.2.10*.

Figure 455. Operating modes



42.3.2.2 Test mode

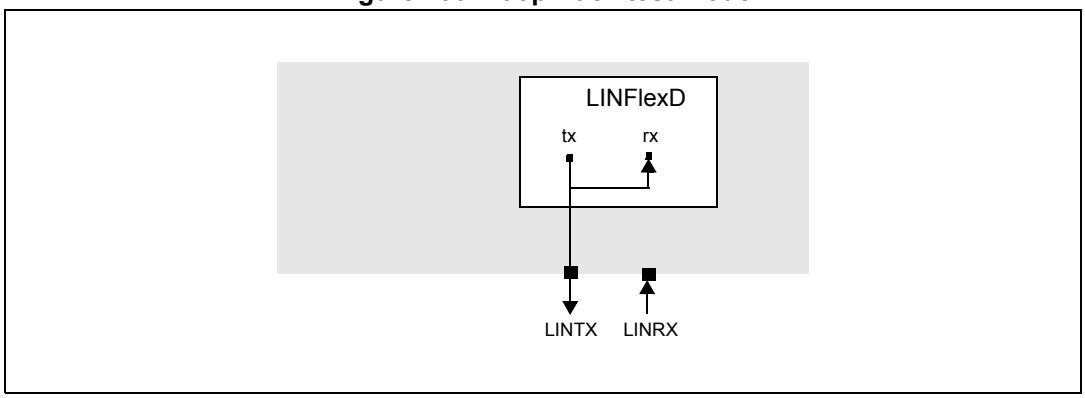
42.3.2.2.1 Loop Back mode

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 456. Loop Back test mode



42.3.2.3 Master mode

Master mode is selected by means of the MME bit in LINCR1 register.

42.3.2.3.1 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.

42.3.2.3.2 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.

42.3.2.3.3 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.

42.3.2.3.4 Data Discard

If the user wants to discard the data after header transmission, then the DDRQ bit in LINCR2 should be set.

42.3.2.4 Slave mode

This mode is selected when the MME bit of the LINCR1 register is cleared.

42.3.2.4.1 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.

42.3.2.4.2 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.

42.3.2.5 Errors

42.3.2.5.1 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.

42.3.2.5.2 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.

42.3.2.5.3 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.

42.3.2.5.4 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.

42.3.2.5.5 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.

42.3.2.5.6 Timeout error

Figure 457. Incomplete response (for example, missing checksum)

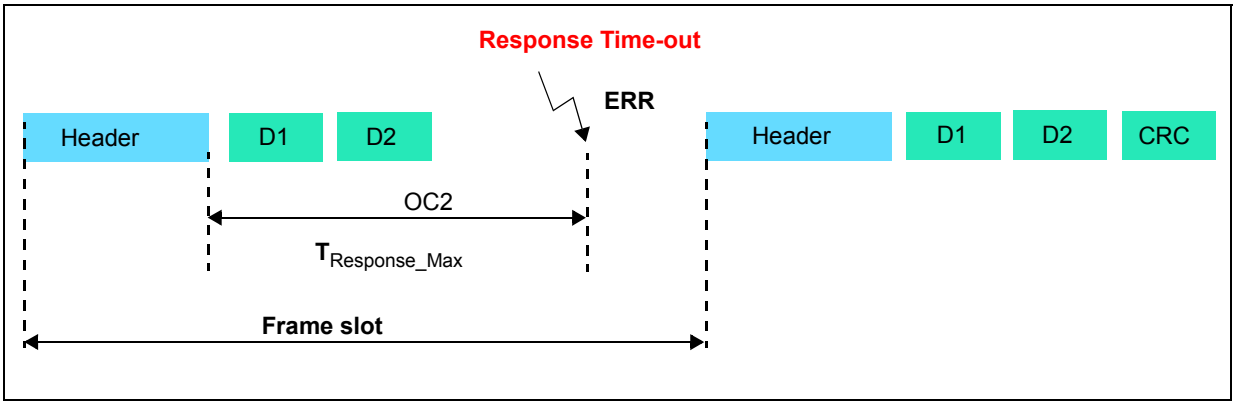
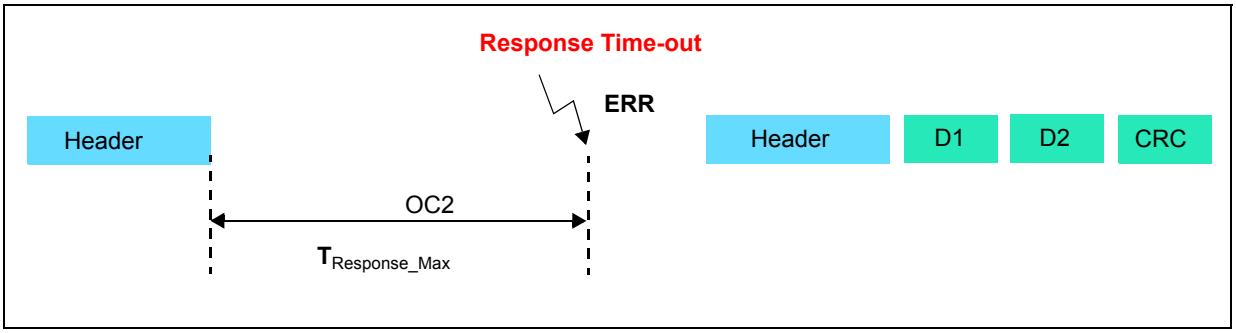


Figure 458. No response



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).

42.3.2.5.7 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 42.4.2.4*).

42.3.2.6 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.

42.3.2.6.1 Identifier list mode

If the nth 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).

42.3.2.6.2 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 nth 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 xth bit of the mask is set, then the xth bit of the received identifier must match the xth bit of filter.

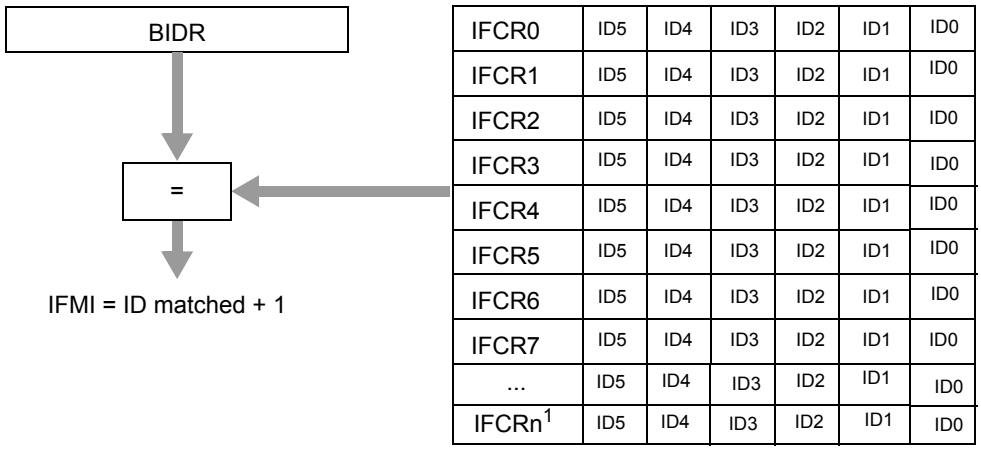
If there is a match of identifier with the *m*th filter in any mode, then *m*+1 is loaded in the IFMI register by hardware. No match condition is denoted by IFMI = 0.

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.

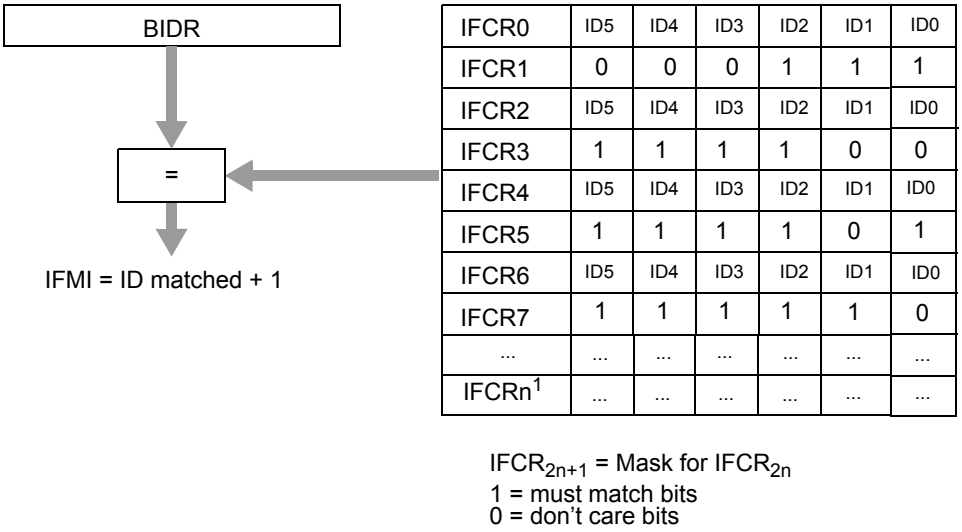
Figure 459. Identifier List mode



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 device configuration chapter to see the number of filters used in this device.

Figure 460. 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 device configuration chapter to see the number of filters used in this device.

42.3.2.7 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 520. 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.

42.3.2.7.1 Start detection mechanism

The following steps are followed for detecting a start bit:

1. Refer to *Table 520* 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.

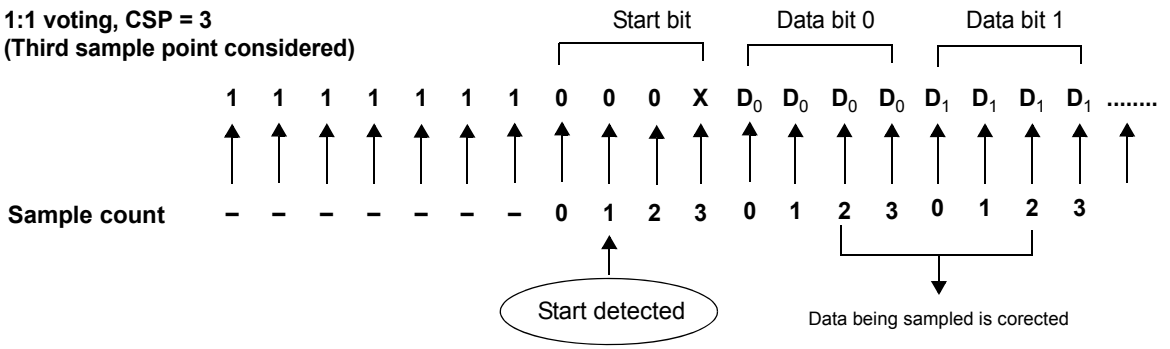
42.3.2.7.2 Delimiter detection mechanism

The following steps are followed for detecting a delimiter bit:

1. Refer to *Table 520* 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.

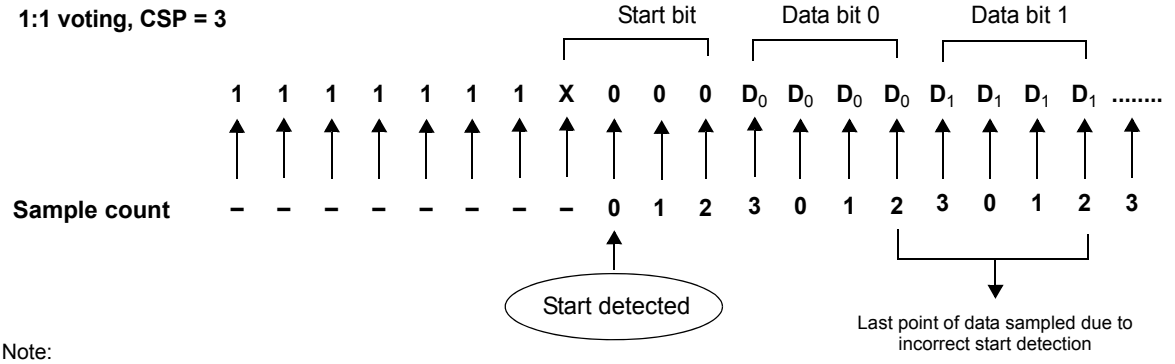
Figure 461. Start detection and sampling for over sampling rate = 4 (Case 1)



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 462. Start detection and sampling for over sampling rate = 4 (Case 2a)



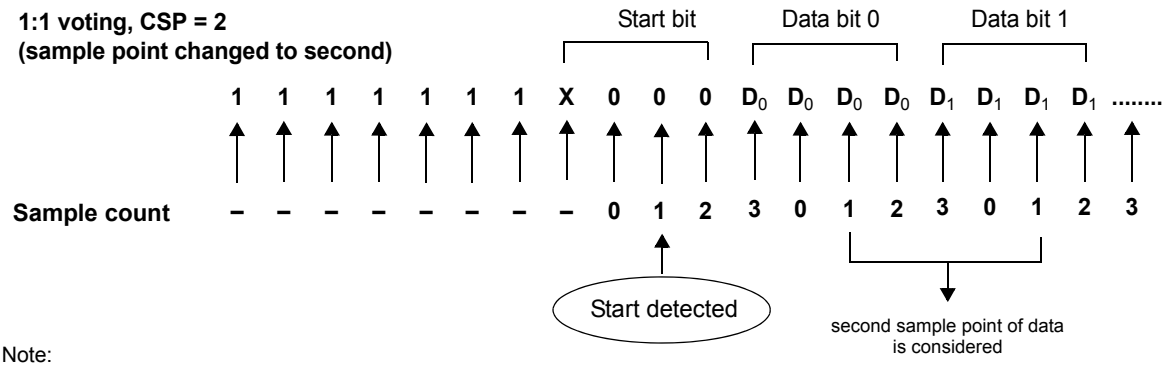
Note:

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:

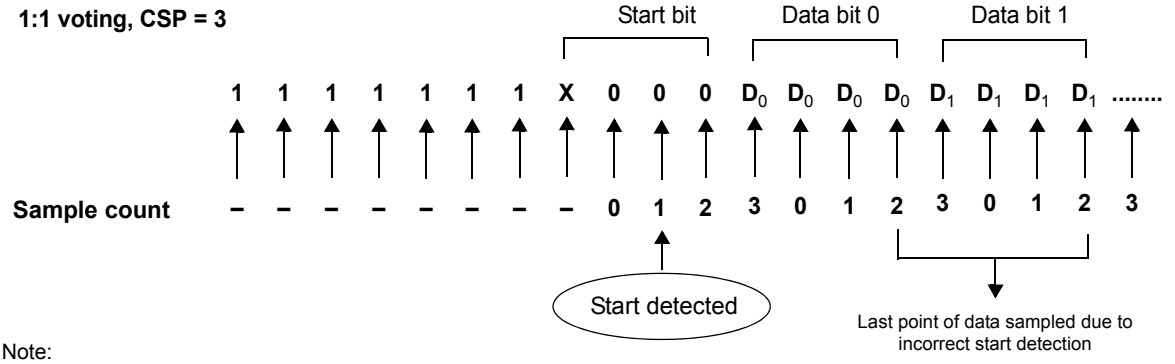
Figure 463. Start detection and sampling for over sampling rate = 4 (Case 2b)



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.

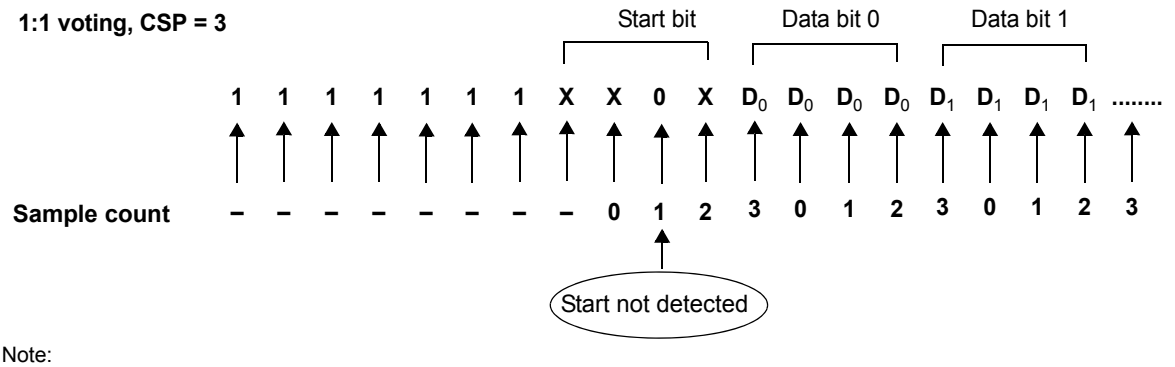
Figure 464. Start detection and sampling for over sampling rate = 4 (Case 3)



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.

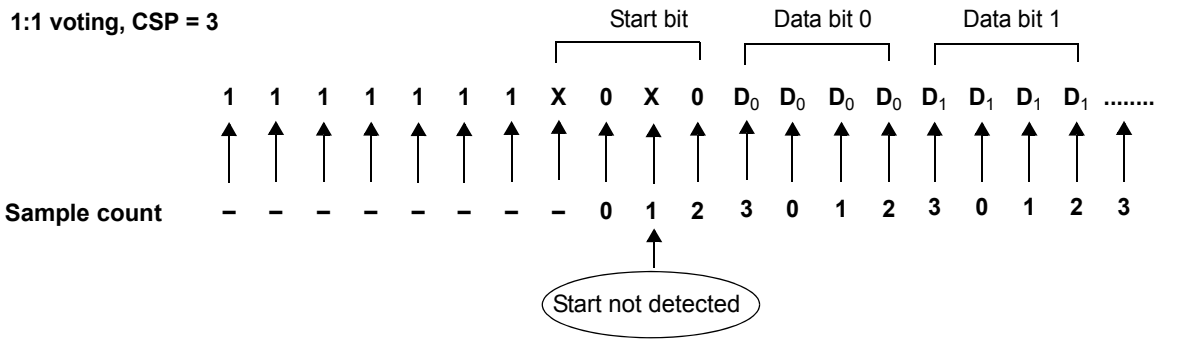
Figure 465. Start detection and sampling for over sampling rate = 4 (Case 4)



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 466. 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.

42.3.2.8 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 41 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)

42.3.2.9 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.

42.3.2.10 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.

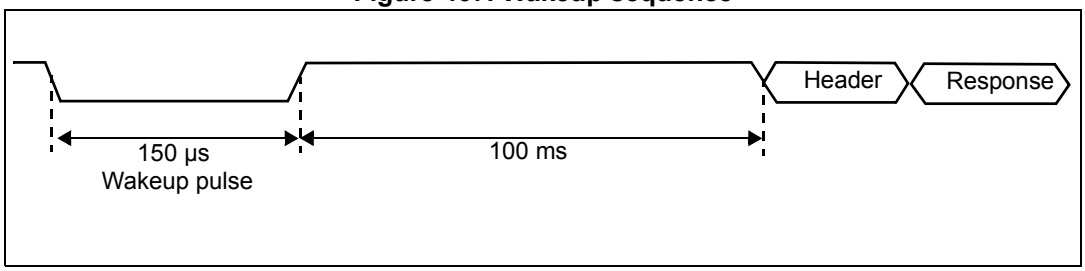
1. AUTOWU = 1

   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 467. Wakeup sequence



42.3.2.11 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.

42.3.3 Timer

There is an 8-bit counter that is free-running (even in Sleep and Initialization modes) if it is configured in Output Compare mode. There are two software-configurable output compare registers that, upon match with the counter value, generate an output compare interrupt provided the TOCE bit in LINTCSR is set.

If this timer is configured in LIN mode, then 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.

42.3.4 UART mode

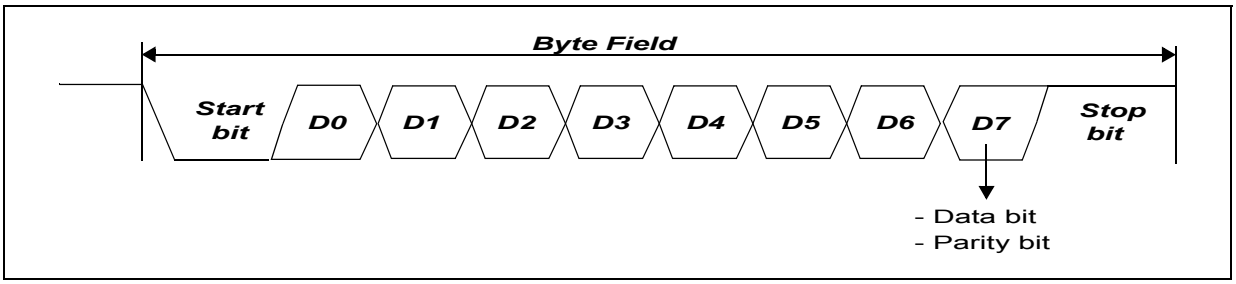
The main features in UART mode are:

- 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

42.3.4.1 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.

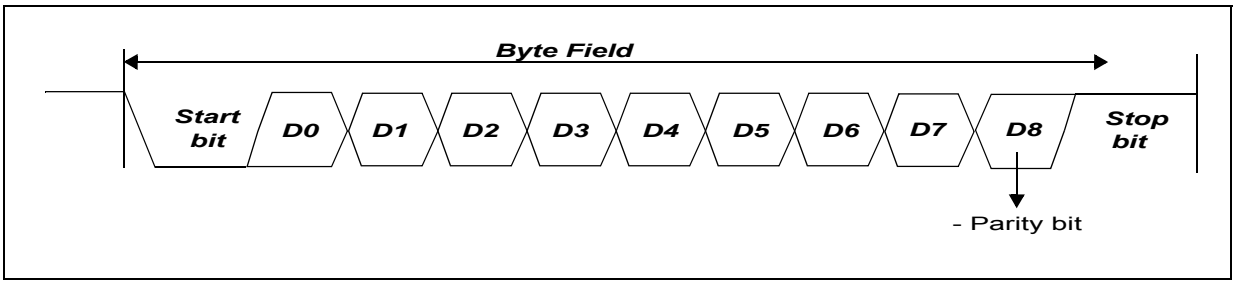
Figure 468. UART mode 8-bit data frame



42.3.4.2 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.

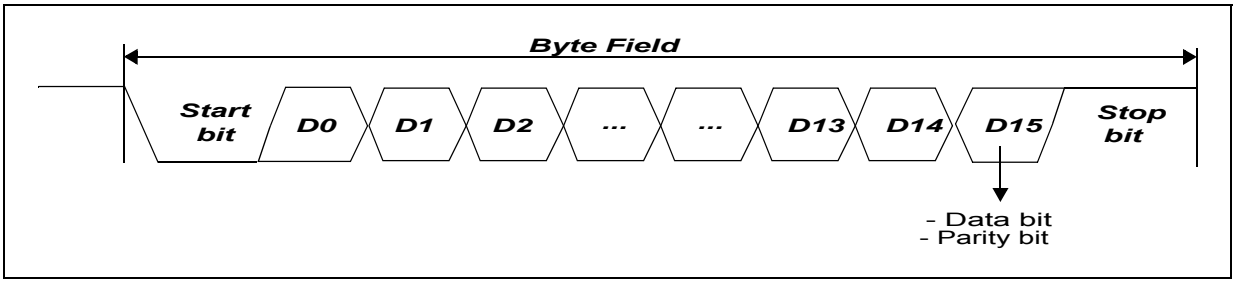
Figure 469. UART mode 9-bit data frame



42.3.4.3 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.

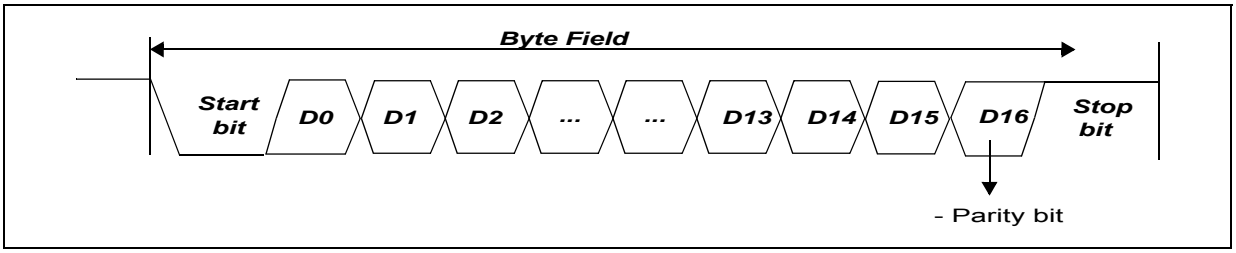
Figure 470. UART mode 16-bit data frame



42.3.4.4 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.

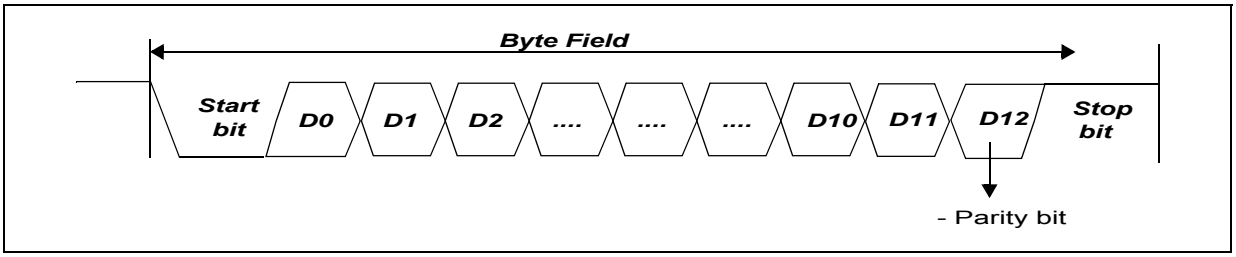
Figure 471. UART mode 17-bit data frame



42.3.4.5 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.

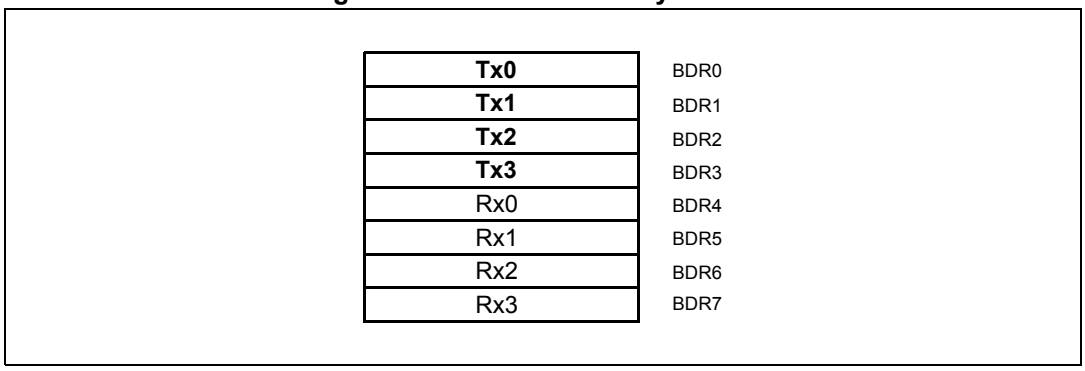
Figure 472. UART mode 13-bit data frame



42.3.4.6 Buffer in UART mode

The 8-byte buffer is divided into two parts: one for receiver and one for transmitter, as shown in *Figure 473*.

Figure 473. 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.

42.3.4.7 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 521. 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 521. 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.

42.3.4.8 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 522. 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

Table 522. BDRM access in UART mode(1),(2)(Continued)

IPS operation	Register	Mode (UARTCR[RFBM])	Word length (UARTCR:WL)	IPS operation result
Read word	BDRM	FIFO	Halfword	IPS transfer error
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.

42.3.5 DMA interface

42.3.5.1 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.

42.3.5.2 Definitions

Control/status register fields of the LINFlexD macrocell are described in *Table 523* and *Table 524*.

Table 523. 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 523. 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 524. 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 525*.

Table 525. 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)

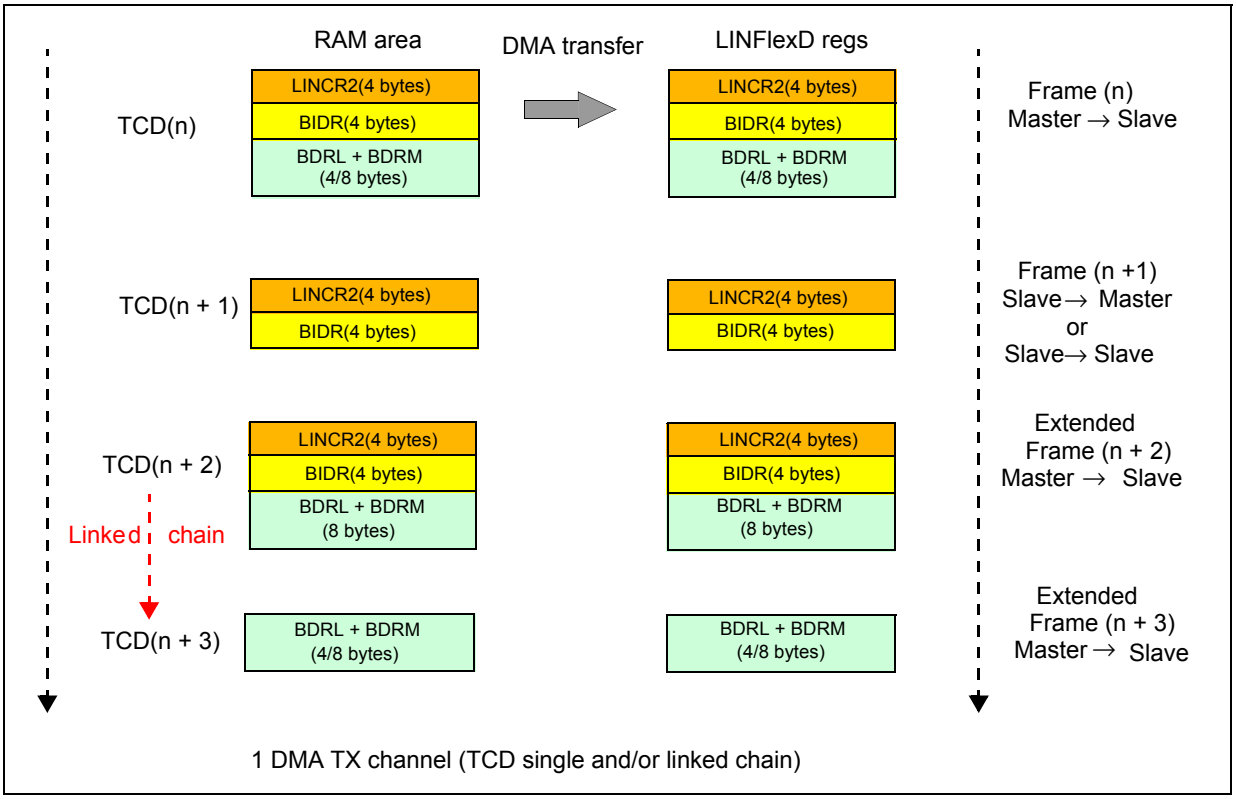
Table 525. TCD control fields description(Continued)

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.

42.3.5.3 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 474*.

Figure 474. 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 526 provides the register settings of the LINCR2 and BIDR for each class of LIN frame.

Table 526. Master node: Tx mode - Register setting

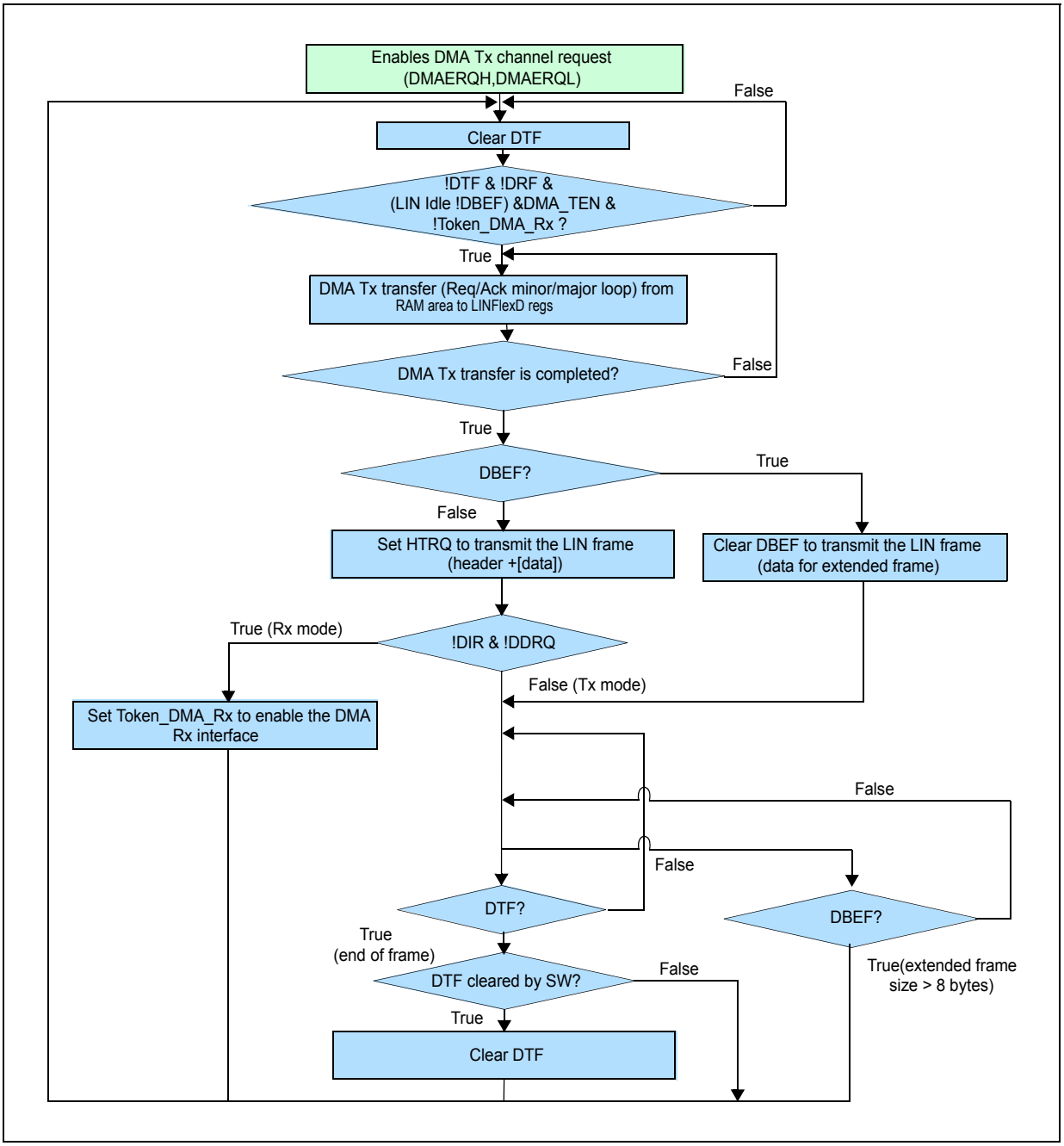
LIN frame	LINCR2	BIDR
Master to Slave	DDRQ = 1 DTRQ = 0 HTRQ = 0	DFL = payload size ID = address CCS = checksum DIR = 1 (TX)
Slave to Master	DDRQ = 0 DTRQ = 0 HTRQ = 0	DFL = payload size ID = address CCS = checksum DIR = 0 (RX)
Slave to Slave	DDRQ = 1 DTRQ = 0 HTRQ = 0	DFL = payload size ID = address CCS = checksum DIR = 0 (RX)

The concept FSM to control the DMA Tx interface is given in *Figure 475*. 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 527*. All other TCD fields = 0. TCD settings based on halfword or byte transfer are allowed.

Table 527. 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

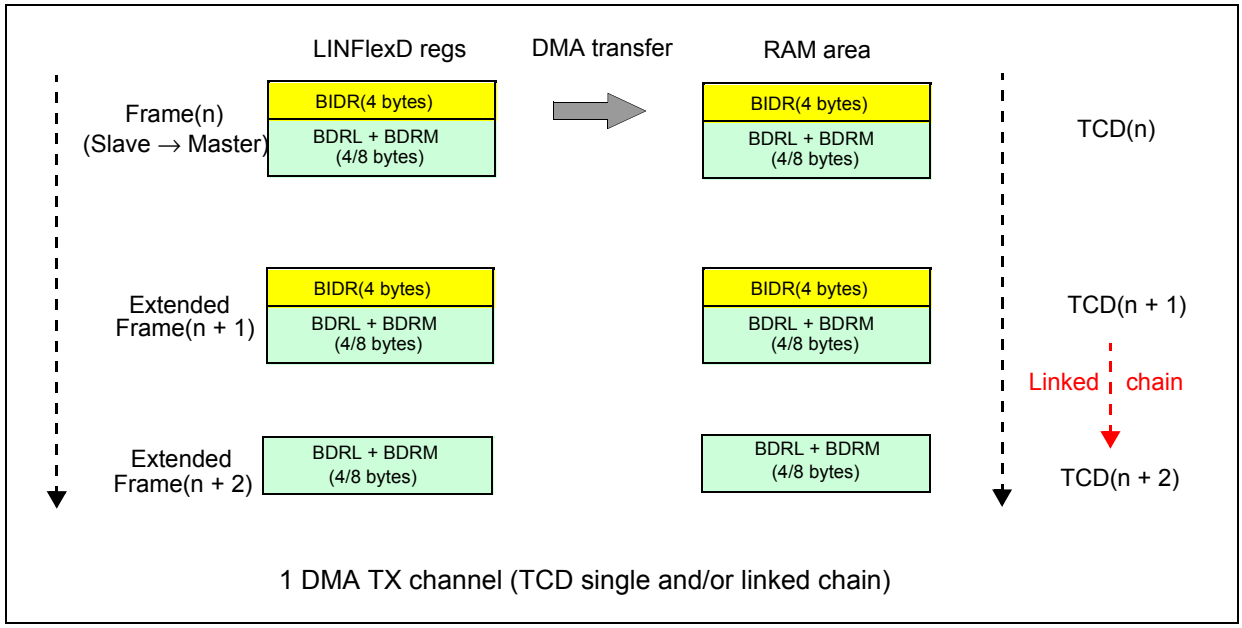
Figure 475. Master node: DMA Tx FSM (concept scheme)



42.3.5.4 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 476*.

Figure 476. 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 477*. 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 528*. All other TCD fields = 0. TCD settings based on halfword or byte transfer are allowed.

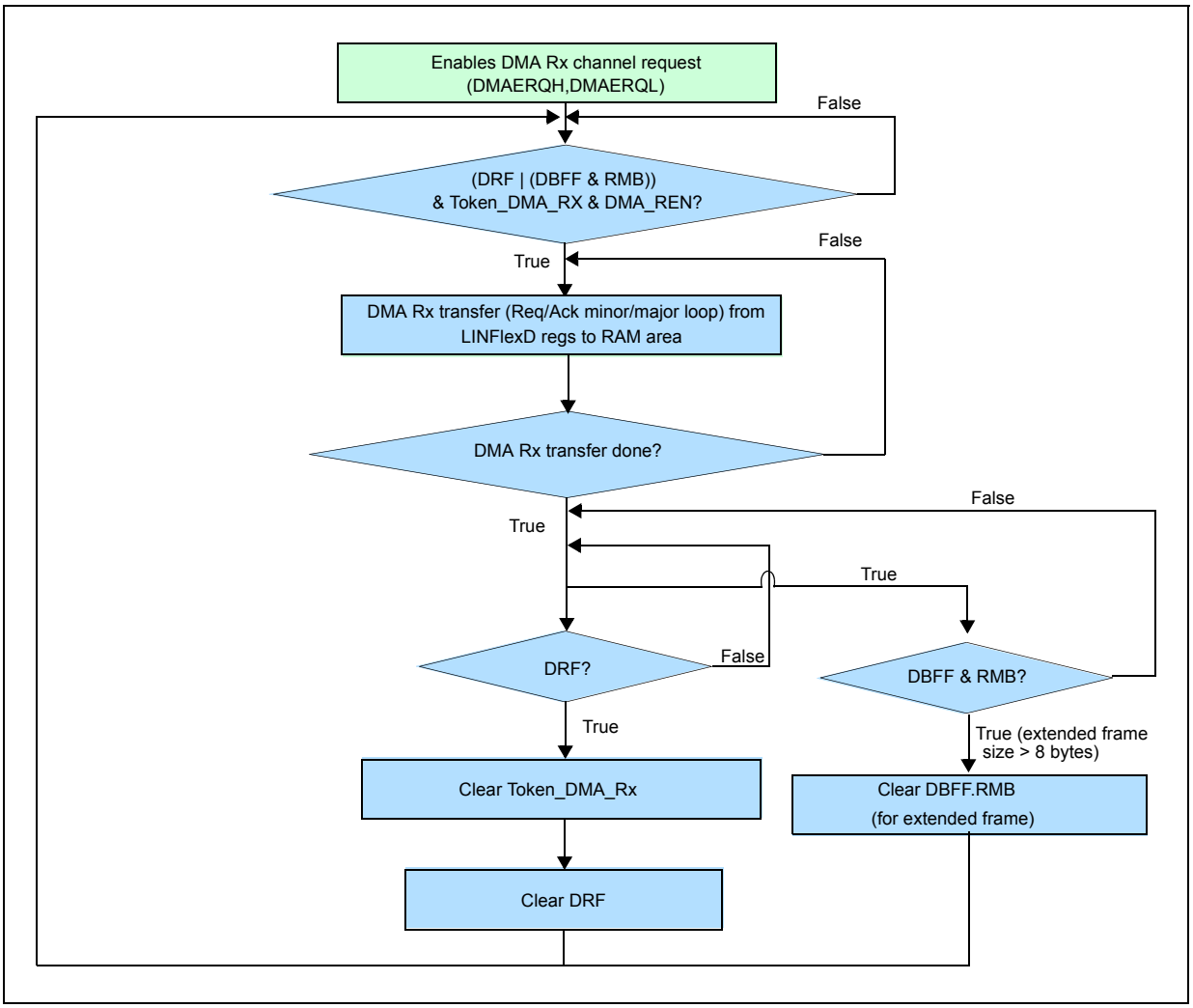
Table 528. TCD setting: Master node - Rx mode

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 528. 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

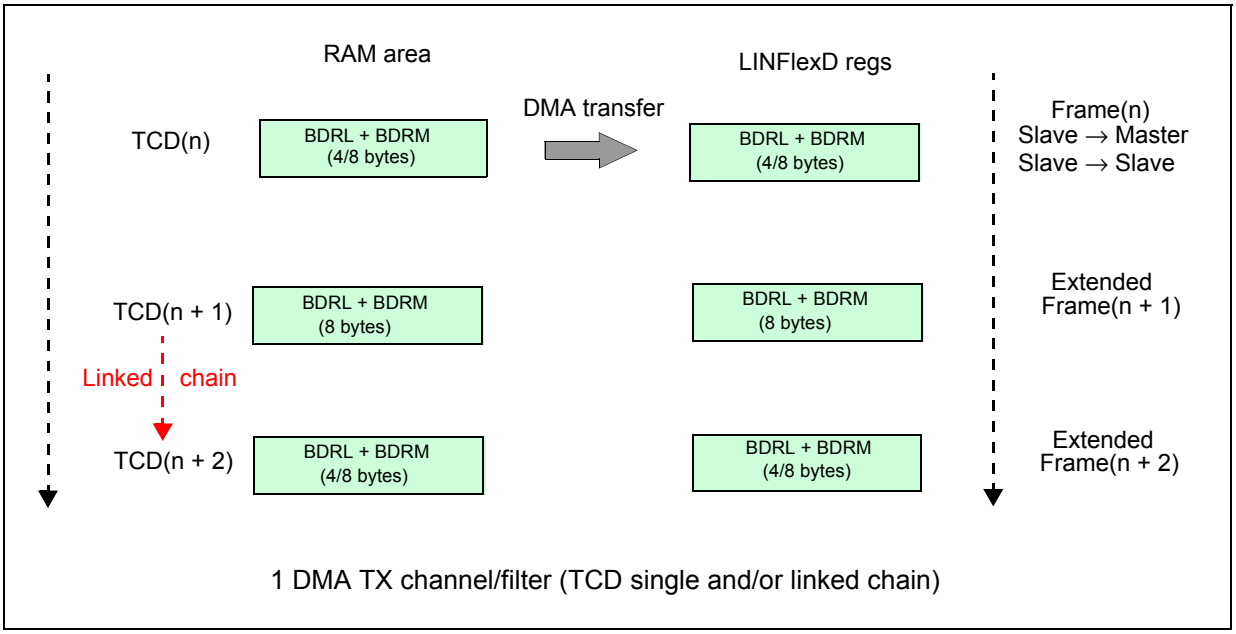
Figure 477. Master node — DMA Rx FSM (concept scheme)



42.3.5.5 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 478*.

Figure 478. 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 529*.

Table 529. 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 479*. DMA TX FSM moves to Idle state if DMATXE[x] = 0 where x = IFMI – 1. The TCD setting (word transfer) is given in *Table 530*. All other TCD fields = 0. TCD settings based on halfword or byte transfer are allowed.

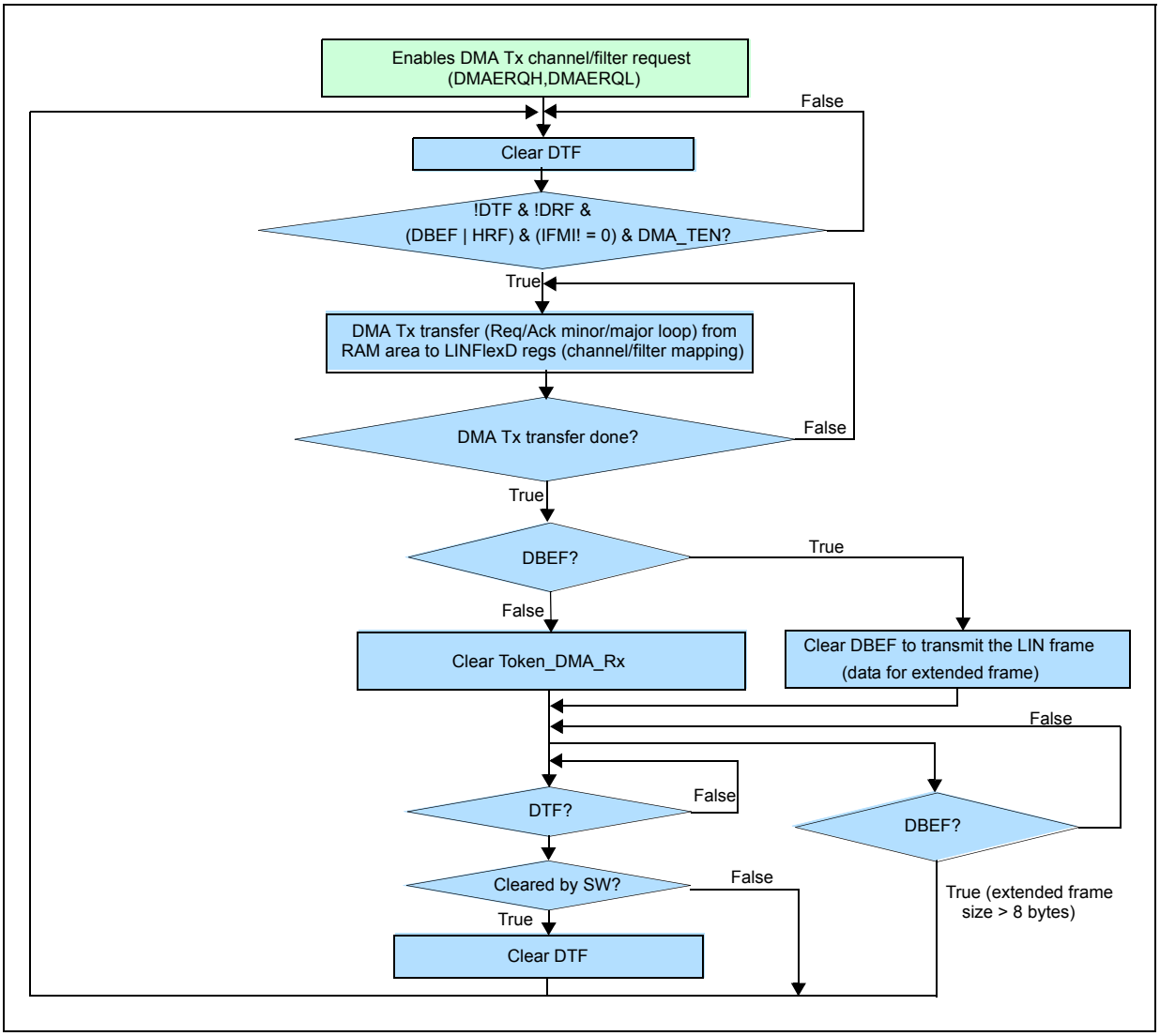
Table 530. TCD setting: Slave node - Tx mode

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 530. 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

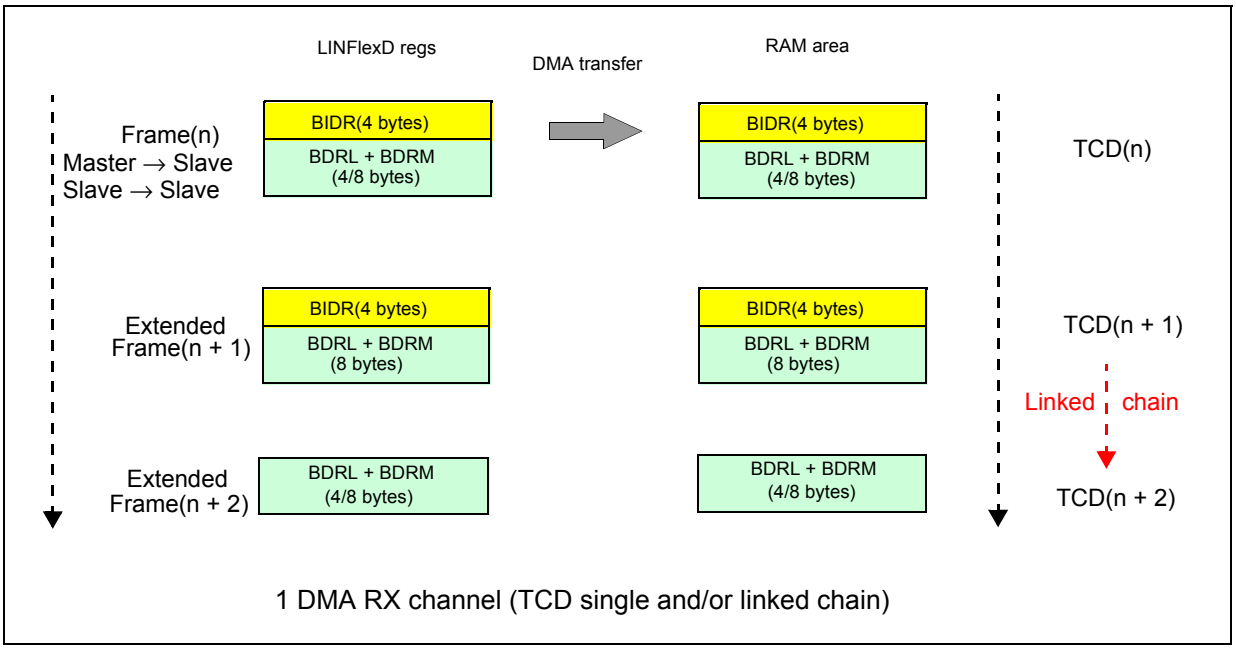
Figure 479. Slave node: DMA Tx FSM (concept scheme)



42.3.5.6 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 480*.

Figure 480. 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 531*.

Table 531. Slave node: Rx mode - Register setting

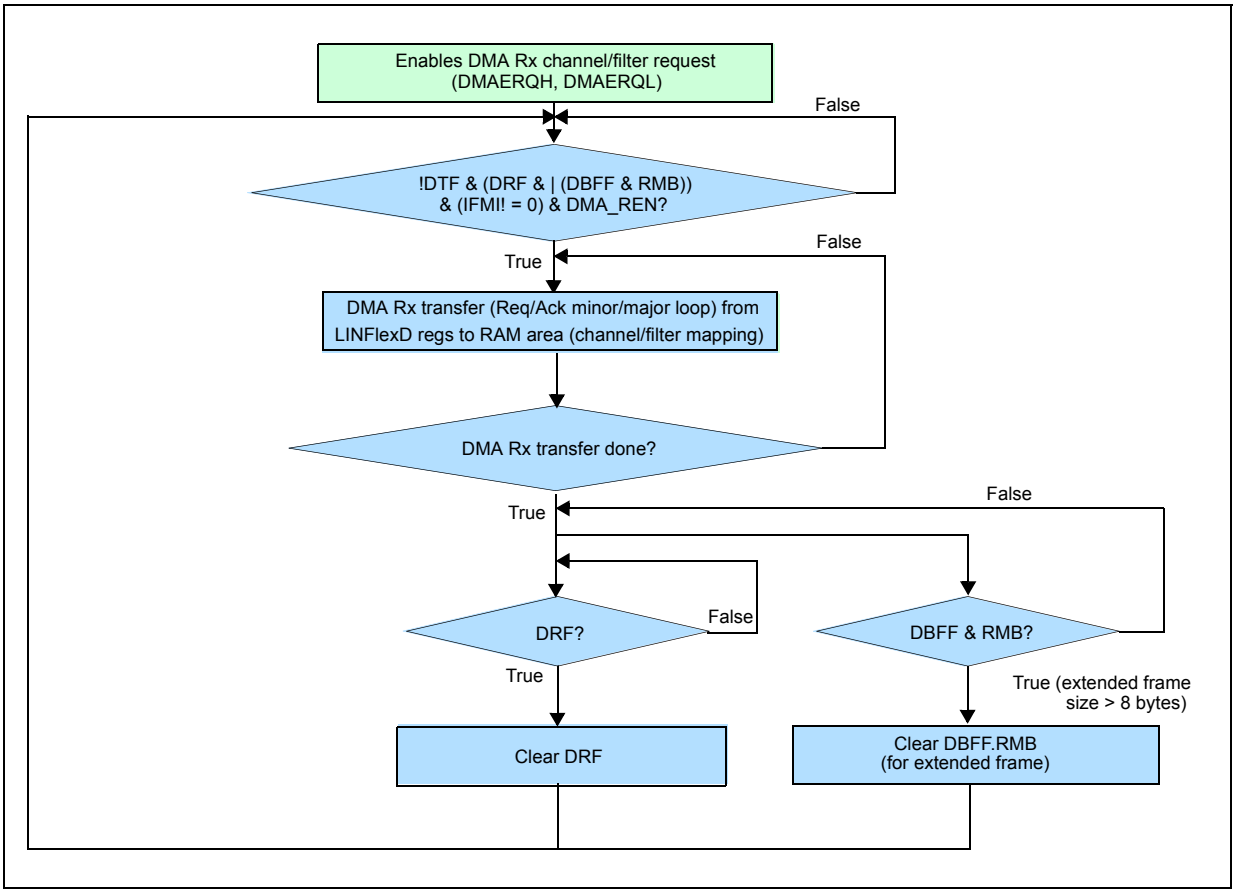
LIN frame	LINCR2	IFER	IFMR	IFCR
Master to Slave or Slave to Slave	DDRQ = 0 DTRQ = 0 HTRQ = 0	To enable an ID filter (Rx mode) for each DMA RX channel	– Identifier list mode Identifier mask mode	DFL = payload size ID = address CCS = checksum DIR = 0 (RX)

The concept FSM to control the DMA Rx interface is given in *Figure 481*. DMA Rx FSM moves to Idle state if DMARXE[x] = 0 where x = IFMI – 1. The TCD setting (word transfer) is given in *Table 532*. All other TCD fields = 0. TCD settings based on halfword or byte transfer are allowed.

Table 532. 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

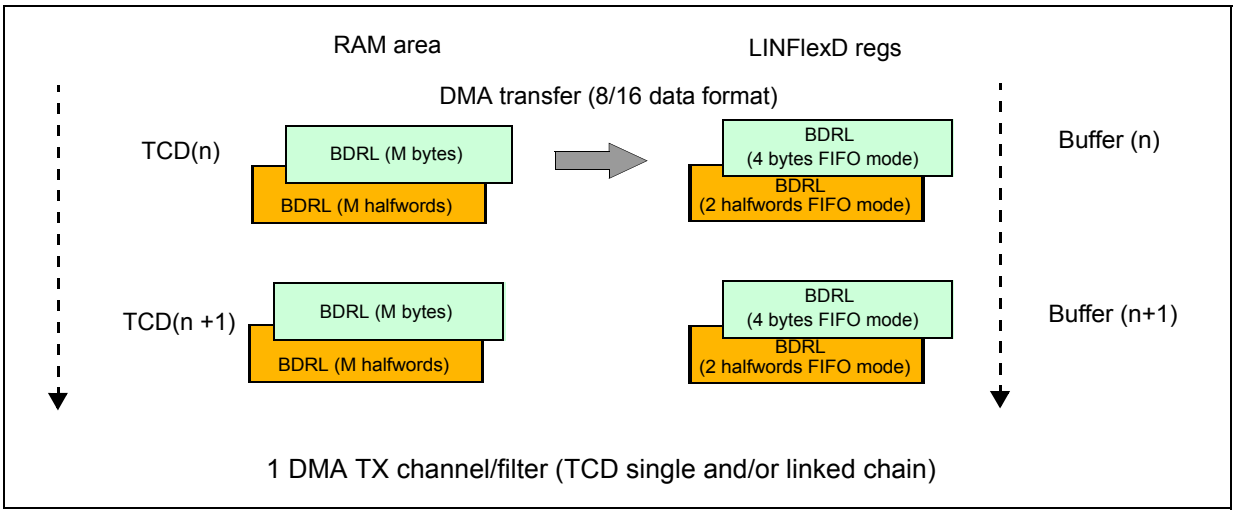
Figure 481. Slave node — DMA Rx FSM (concept scheme)



42.3.5.7 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 482*.

Figure 482. 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.

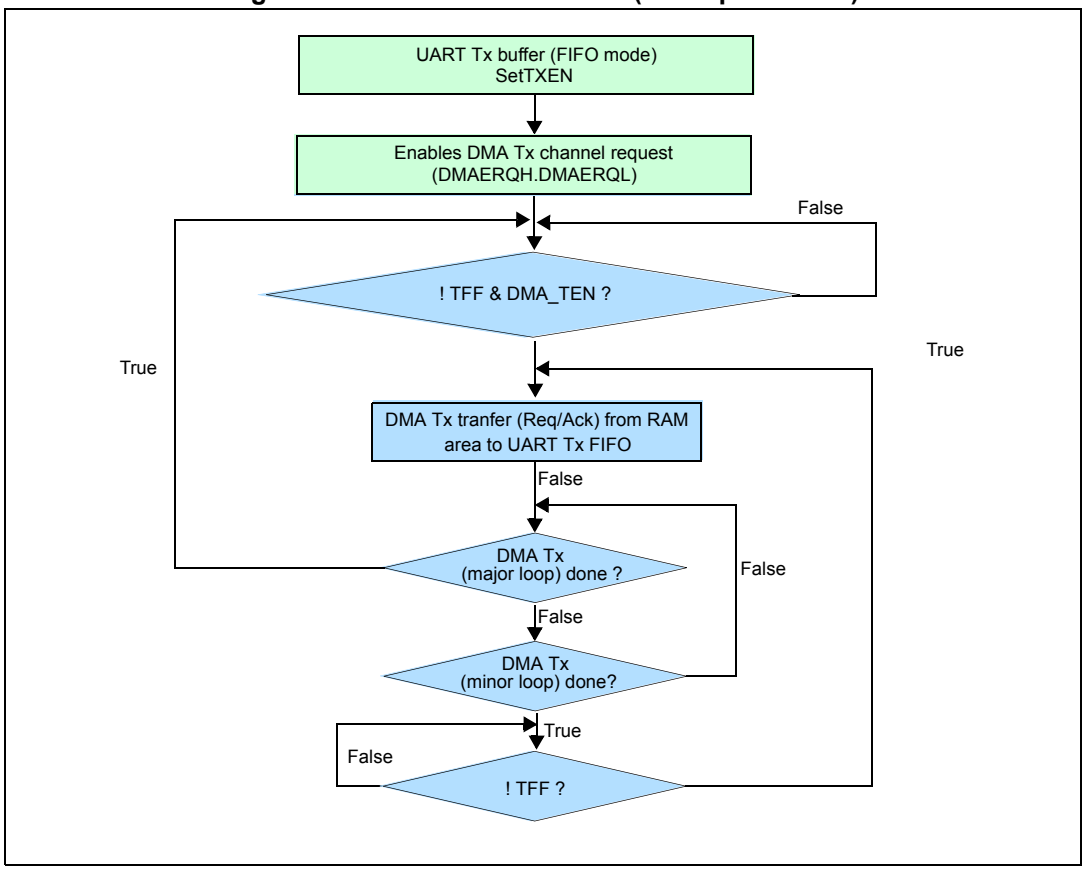
The concept FSM to control the DMA Tx interface is given in *Figure 483*. DMA Tx FSM will move to Idle state if DMATXE[0] = 0. The TCD setting (typical case) is given in *Table 533*. All other TCD fields = 0. The minor loop transfers a single byte/halfword as soon as a free entry is available in the Tx FIFO.

Table 533. 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.

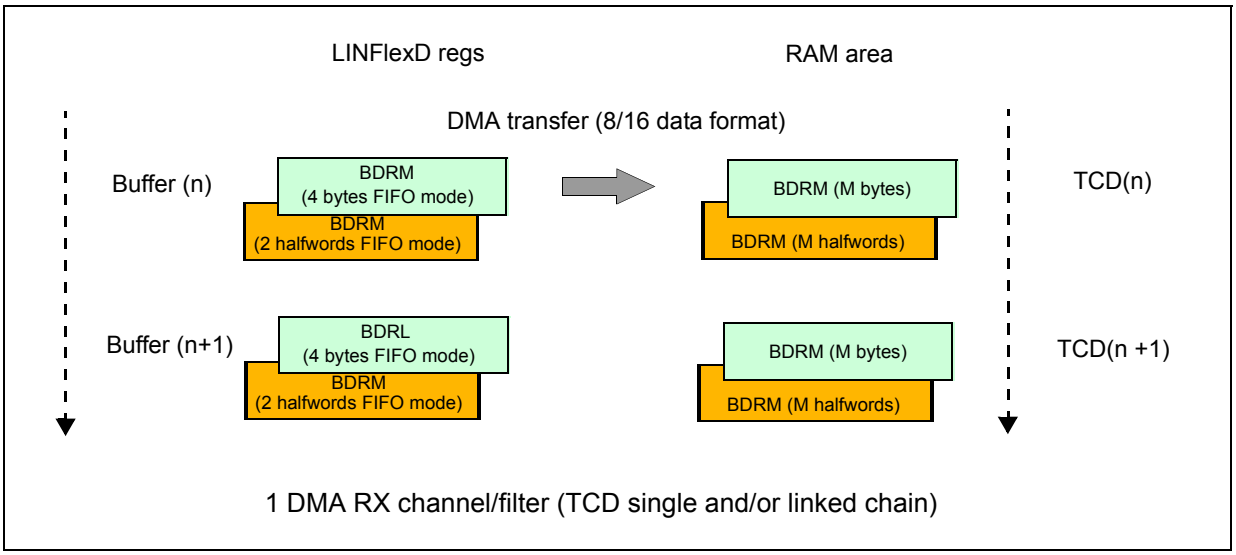
Figure 483. UART: DMA Tx FSM (concept scheme)



42.3.5.8 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 534*.

Figure 484. 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.

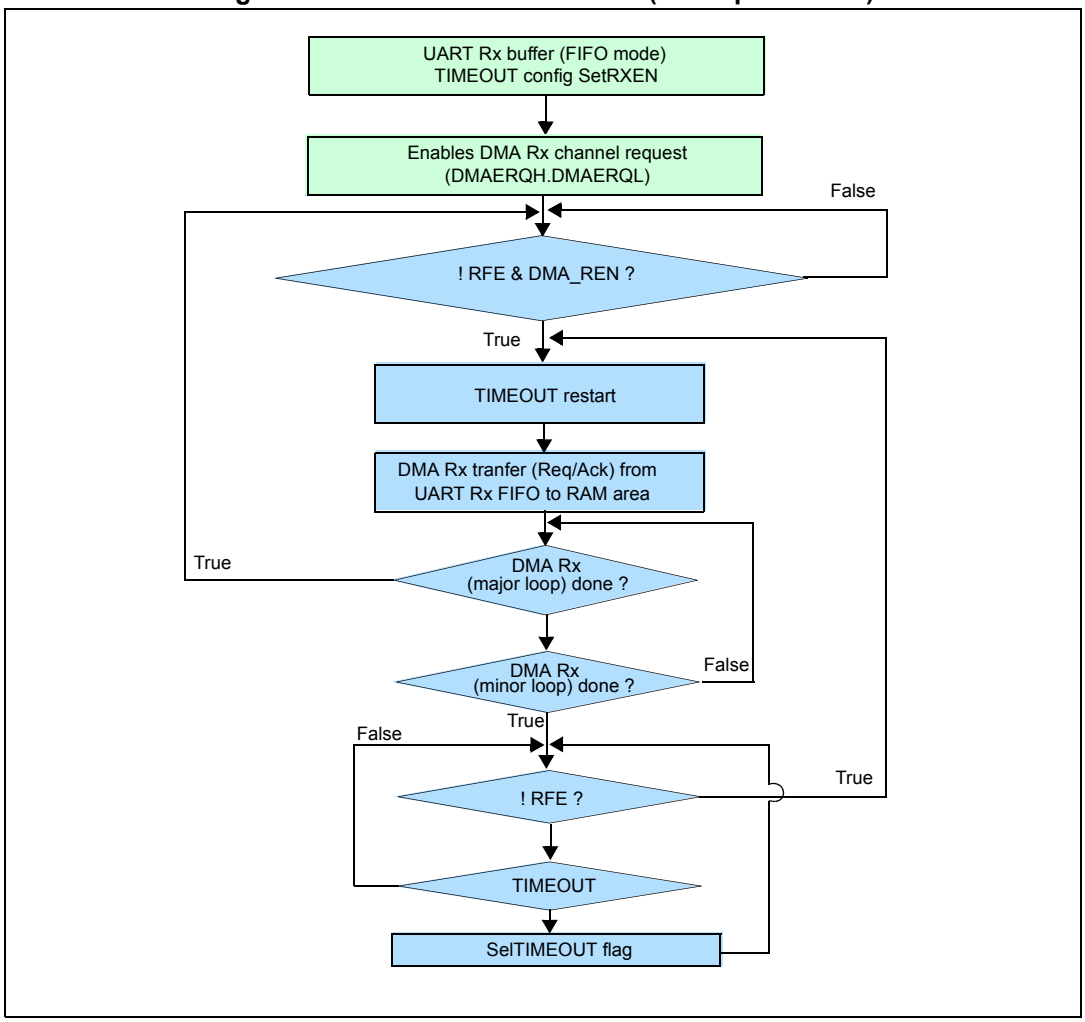
The concept FSM to control the DMA Rx interface is given in *Figure 485*. DMA Rx FSM will move to Idle state if DMARXE[0] = 0. The TCD setting (typical case) is given in the next table. All other TCD fields equal zero. The minor loop transfers a single byte/halfword as soon an entry is available in the 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 534. 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 485. UART — DMA Rx FSM (concept scheme)



42.3.5.9 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 486. DMA interface (major loop — single iteration)

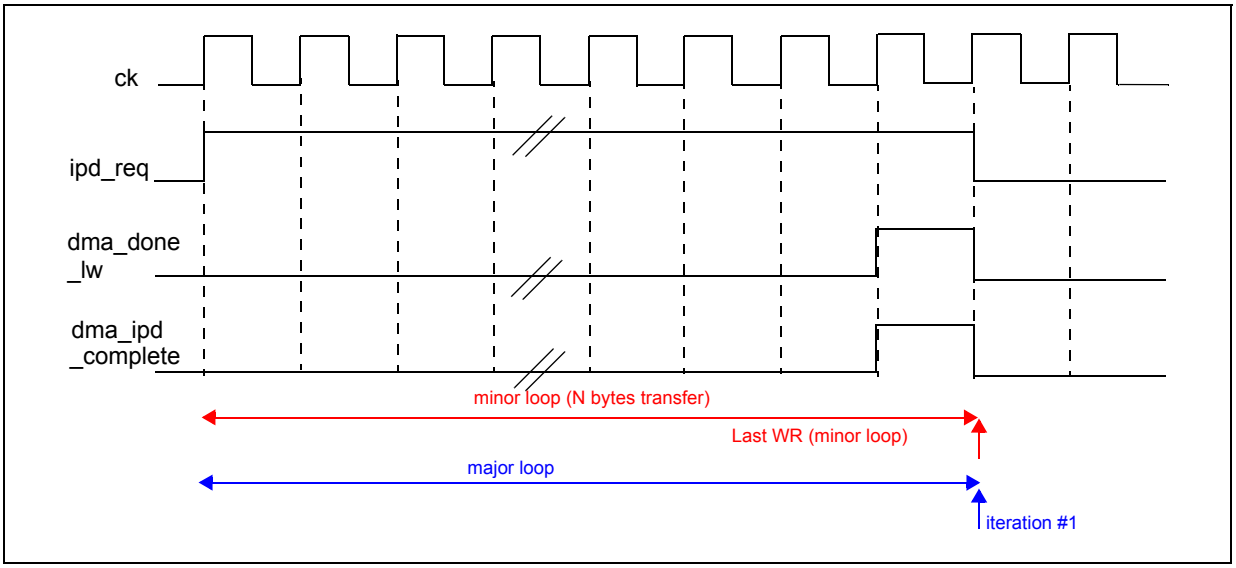
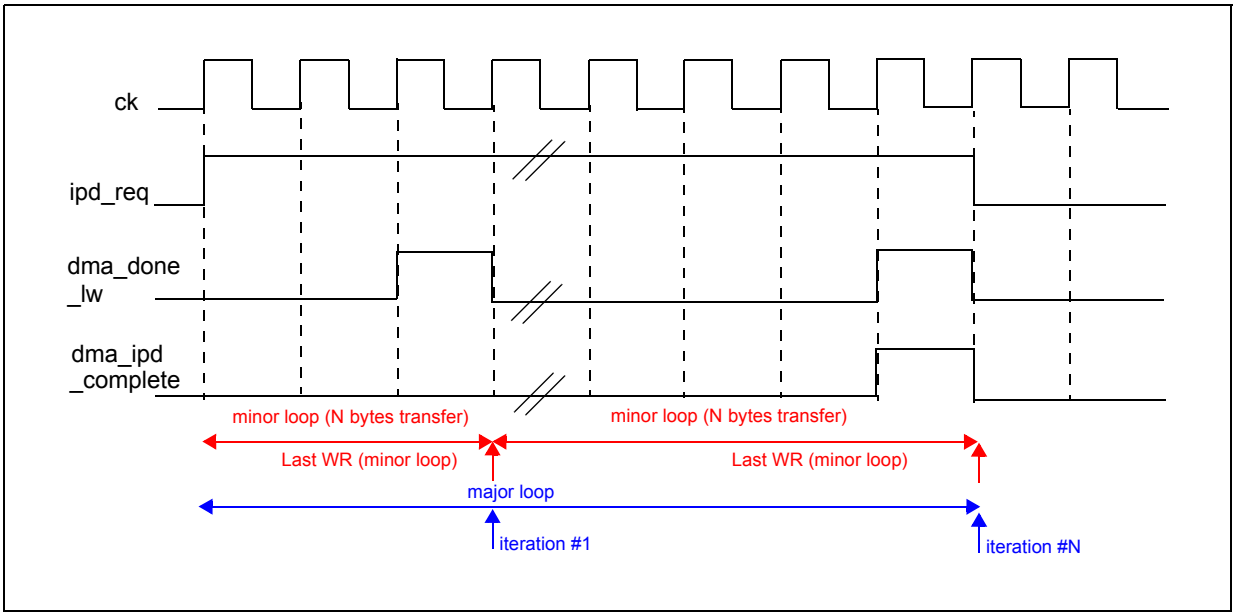


Figure 487. DMA interface (major loop — N iteration)



42.3.5.10 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.

42.4 Memory map and register description

42.4.1 Memory map

Table 535 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 535. LINFlexD memory map

Address offset (hex)	Register description	Register classification(1)	Location
0x00	LIN Control register 1 (LINCR1)	Saf-Relv	*Section 42.4.2.1*
0x04	LIN Interrupt Enable register (LINIER)	Saf-Relv	*Section 42.4.2.2*
0x08	LIN Status register (LINSR)	No-Saf	*Section 42.4.2.3*
0x0C	LIN Error Status register (LINESR)	No-Saf	*Section 42.4.2.4*
0x10	UART Mode Control register (UARTCR)	Saf-Relv	*Section 42.4.2.5*
0x14	UART Mode Status register (UARTSR)	No-Saf	*Section 42.4.2.6*
0x18	LIN Timeout Control Status register (LINTCSR)	Saf-Relv	*Section 42.4.2.7*
0x1C	LIN Output Compare register (LINOCR)	Saf-Relv	*Section 42.4.2.8*
0x20	LIN Timeout Control register (LINTOCR)	Saf-Relv	*Section 42.4.2.9*
0x24	LIN Fractional Baud Rate register (LINFBRR)	Saf-Relv	*Section 42.4.2.10*

Table 535. LINFlexD memory map(Continued)

Address offset (hex)	Register description	Register classification[1]	Location
0x28	LIN Integer Baud Rate register (LINIBRR)	Saf-Relv	*Section 42.4.2.11*
0x2C	LIN Checksum Field register (LINCFR)	Saf-Relv	*Section 42.4.2.12*
0x30	LIN Control register 2 (LINCR2)	Saf-Relv	*Section 42.4.2.13*
0x34	Buffer Identifier register (BIDR)	Saf-Relv	*Section 42.4.2.14*
0x38	Buffer Data Register Least Significant (BDRL)	No-Saf	*Section 42.4.2.15*
0x3C	Buffer Data Register Most Significant (BDRM)	No-Saf	*Section 42.4.2.16*
0x40	Identifier Filter Enable register (IFER)	Saf-Relv	*Section 42.4.2.17*
0x44	Identifier Filter Match index (IFMI)	Saf-Relv	*Section 42.4.2.18*
0x48	Identifier Filter Mode register (IFMR)	Saf-Relv	*Section 42.4.2.19*
0x4C[2]	Identifier Filter Control registers (IFCR0)	Saf-Relv	*Section 42.4.2.20*
0x50[2]	Identifier Filter Control registers (IFCR1)	Saf-Relv	*Section 42.4.2.21*
0x54[2]	Identifier Filter Control registers (IFCR2)	Saf-Relv	*Section 42.4.2.20*
0x58[2]	Identifier Filter Control registers (IFCR3)	Saf-Relv	*Section 42.4.2.21*
0x5C[2]	Identifier Filter Control registers (IFCR4)	Saf-Relv	*Section 42.4.2.20*
0x60[2]	Identifier Filter Control registers (IFCR5)	Saf-Relv	*Section 42.4.2.21*
0x64[2]	Identifier Filter Control registers (IFCR6)	Saf-Relv	*Section 42.4.2.20*
0x68[2]	Identifier Filter Control registers (IFCR7)	Saf-Relv	*Section 42.4.2.21*
0x6C[2]	Identifier Filter Control registers (IFCR8)	Saf-Relv	*Section 42.4.2.20*
0x70[2]	Identifier Filter Control registers (IFCR9)	Saf-Relv	*Section 42.4.2.21*
0x74[2]	Identifier Filter Control registers (IFCR10)	Saf-Relv	*Section 42.4.2.20*
0x78[2]	Identifier Filter Control registers (IFCR11)	Saf-Relv	*Section 42.4.2.21*
0x7C[2]	Identifier Filter Control registers (IFCR12)	Saf-Relv	*Section 42.4.2.20*
0x80[2]	Identifier Filter Control registers (IFCR13)	Saf-Relv	*Section 42.4.2.21*
0x84[2]	Identifier Filter Control registers (IFCR14)	Saf-Relv	*Section 42.4.2.20*
0x88[2]	Identifier Filter Control registers (IFCR15)	Saf-Relv	*Section 42.4.2.21*
0x8C[3]	Global Control register (GCR)	Saf-Relv	*Section 42.4.2.22*
0x90[3]	UART Preset Timeout register (UARTPTO)	Saf-Relv	*Section 42.4.2.23*
0x94[3]	UART Current Timeout register (UARTCTO)	No-Saf	*Section 42.4.2.24*
0x98[3]	DMA Tx Enable register (DMATXE)	Saf-Relv	*Section 42.4.2.25*
9C[3]	DMA Rx Enable register (DMARXE)	Saf-Relv	*Section 42.4.2.26*

1. No-Saf = Not safety relevant, Saf-Lat = Safety Latent, Saf-Relv = Safety Relevant, Saf-Crit = Safety Critical.
2. Refer to the device configuration chapter for the number of filters used in each instance of LINFlexD in this device.

3. The address offset depends on the no_of_filters and value of slave. Refer to the device configuration chapter for the number of filters and value of slave used in this device.

42.4.2 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.

42.4.2.1 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(2)	0/1	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 488. LIN Control Register 1 (LINCR1)

Table 536. 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.

Table 536. LINCR1 field descriptions(Continued)

Field	Description
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).
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 42.3.2.2*. Note that this bit can be programmed only in Initialization mode. This bit is utilized in UART mode also.

Table 536. LINCR1 field descriptions(Continued)

Field	Description
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.
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.

42.4.2.2 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 489. LIN Interrupt enable register (LINIER)

Table 537. 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 537. 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.

42.4.2.3 LIN Status Register (LINSR)

This register consists of status bits indicating the state of the LINFlexD hardware.

Address: 0x0008 Access: User read/write

	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	AUTOSYNC_COMP	RDC		
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	LINS				0	0	RMB	DRBNE	RXBUSY	RDI	WUF	DBFF	DBEF	DRF	DTF	HRF
W							W1C	W1C			W1C	W1C	W1C	W1C	W1C	W1C
Reset	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0

Figure 490. LIN Status Register (LINSR)

Table 538. 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 538. LINSR field descriptions(Continued)

Field	Description
16:19 LINS[3:0]	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 538. 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 flagThis 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.

42.4.2.4 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 491. LIN error status register (LINESR)

Refer to *Section 42.3.2.5* for detailed description of all errors.

Table 539. 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 539. 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.

42.4.2.5 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 492. UART Mode Control Register (UARTCR)

Table 540. 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 540. 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 540. 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 540. 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.

42.4.2.6 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	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 493. UART Mode Status Register (UARTSR)

Table 541. 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[0:3]	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 541. 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.

42.4.2.7 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 494. LIN Time-Out Control Status Register (LINTCSR)

Table 542. 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.

42.4.2.8 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 495. LIN Output Compare Register (LINOCR)

Table 543. LINOCR field descriptions

Field	Description
16:23 OC2[7:0]	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[7:0]	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.

42.4.2.9 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 496. LIN Time-Out Control Register (LINTOCR)

Table 544. LINTOCR field descriptions

Field	Description
20:23 RTO[3:0]	Response timeout value This is the response timeout duration (in bit time) for 1 byte. The reset value is 0Eh = 14, corresponding to $T_{Response_Maximum}$ = 1.4 * $T_{Response_Nominal}$
25:31 HTO[6:0]	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 the 2Dh = 45, corresponding to $T_{Header_Maximum}$ in the LIN specification. If Master mode is enabled then these bits will always reflect 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.

42.4.2.10 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 497. LIN Fractional Baud Rate Register (LINFBRR)

Table 545. LINFBRR field descriptions

Field	Description
28:31 FBR[3:0]	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

42.4.2.11 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[19:16](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[15:0](1)															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. This field can be read in any modes and written only in *Initialization* mode.

Figure 498. LIN Integer Baud Rate Register (LINIBRR)

Table 546. LINIBRR field descriptions

Field	Description
12:31 IBR[19:0]	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

42.4.2.12 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 499. LIN Checksum Field Register (LINCFR)

Table 547. LINCFR field descriptions

Field	Description
24:31 CF[7:0]	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

42.4.2.13 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, this field reads ‘1’ and cannot be programmed.

Figure 500. LIN Control Register 2 (LINCR2)

Table 548. 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.

42.4.2.14 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 501. Buffer Identifier Register (BIDR)

Table 549. BIDR field descriptions

Field	Description
16:21 DFL[5:0]	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[5:0]	Identifier Identifier part of the identifier field without the identifier parity. This field can be written only in Master mode (MME = ‘1’).

42.4.2.15 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 502. Buffer Data Register Least Significant (BDRL)

Table 550. BDRL field descriptions

Field	Description
0:7 DATA3[7:0]	Data Byte 3 Data byte 3 of the data field
8:15 DATA2[7:0]	Data Byte 2 Data byte 2 of the data field
16:23 DATA1[7:0]	Data Byte 1 Data byte 1of the data field
24:31 DATA0[7:0]	Data Byte 0 Data byte 0 of the data field

42.4.2.16 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 503. Buffer Data Register Most Significant (BDRM)

Table 551. BDRM field descriptions

Field	Description
0:7 DATA7[7:0]	Data Byte 7 Data byte 7 of the data field
8:15 DATA6[7:0]	Data Byte 6 Data byte 6 of the data field
16:23 DATA5[7:0]	Data Byte 5 Data byte 5 of the data field
24:31 DATA4[7:0]	Data Byte 4 Data byte 4 of the data field

42.4.2.17 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 504. Identifier Filter Enable Register (IFER)

Table 552. IFER field descriptions

Field	Description
16:31 FACT[15:0]	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 device configuration chapter to see the number of filters used in the device.

42.4.2.18 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 505. Identifier Filter Match Index (IFMI)

Table 553. IFMI field descriptions

Field	Description
27:31 IFMI[4:0]	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 42.3.2.4* for more details).

42.4.2.19 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 506. Identifier Filter Mode Register (IFMR)

Table 554. 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*

42.4.2.20 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+8×*n*) where *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 507. Identifier Filter Control Register (IFCR2n)

Table 555. IFCR2*n* field descriptions

Field	Description
16:21 DFL[5:0]	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[5:0]	Identifier Identifier part of the identifier field without the identifier parity.

42.4.2.21 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+8×*n*) where *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 508. Identifier Filter Control Register (IFCR2n+1)

Table 556. IFCR2*n*+1 field descriptions

Field	Description
16:21 DFL[5:0]	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[5:0]	Identifier Identifier part of the identifier field without the identifier parity.

42.4.2.22 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: 0x4C+4×no_of_filters×slave(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	0	0	0	0	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.
3. Refer to device configuration chapter to see the number of filters and value of slave used in the device.

Figure 509. Global Control Register (GCR)

Table 557. GCR field descriptions

Field	Description
26 TDFBM	Transmit data first bit MSBThis 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 557. 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

42.4.2.23 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: (GCR address)[1]+04h 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

1. +Depends on no_of_filters. Refer to device configuration chapter to see the number of filters used in the device.

Figure 510. UART Preset Timeout Register (UARTPTO)

Table 558. 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.

42.4.2.24 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 42.4.2.23* 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: (GCR address)(1)+08h 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

1. Depends on no_of_filters. Refer to device configuration chapter to see the number of filters used in the device.

Figure 511. UART Current Timeout Register (UARTCTO)

Table 559. UARTCTO field descriptions

Field	Description
20:31 CTO[11:0]	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.

42.4.2.25 DMA Tx Enable Register (DMATXE)

This register enables the DMA TX interface. This register can be written and read by software any time.

Address: (GCR address)(1)+0Ch 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

1. Depends on no_of_filters. Refer to device configuration chapter to see the number of filters used in the device.

Figure 512. DMA Tx Enable Register (DMATXE)

Table 560. 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.

42.4.2.26 DMA Rx Enable Register (DMARXE)

This register enables the DMA RX interface. This register can be written and read by software any time.

Address: (GCR address)(1)+10h 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	DRE															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. Depends on no_of_filters. Refer to device configuration chapter to see the number of filters used in the device.

Figure 513. DMA Rx Enable Register (DMARXE)

Table 561. DMARXE field descriptions

Field	Description
(32–2**TX_CH_NUM):31 DRE[2**TX_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.

42.5 Programming considerations

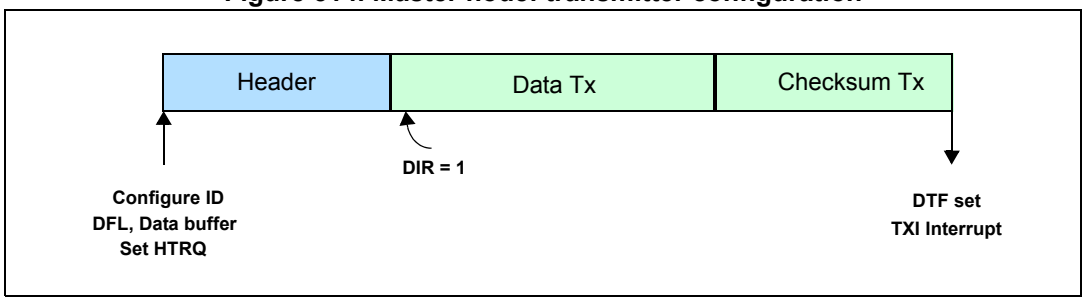
The next subsections describe the various configurations in which the LINFlexD module can be used.

42.5.1 Master node

LINFlexD acts as Master when the MME bit is set.

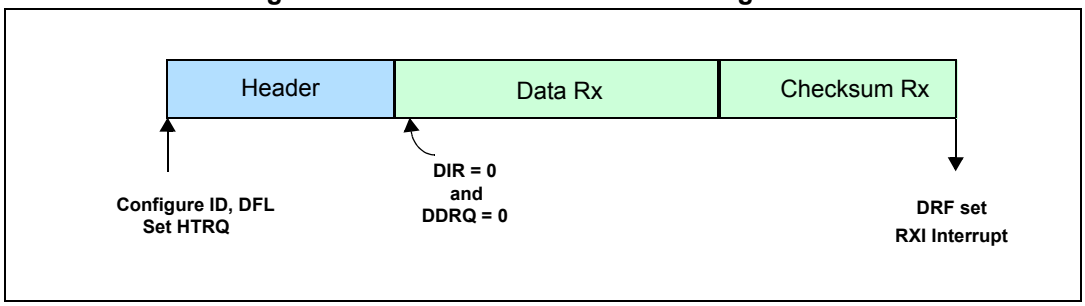
42.5.1.1 Transmitter

Figure 514. Master node: transmitter configuration



42.5.1.2 Receiver

Figure 515. Master node: receiver configuration



42.5.1.3 Transmitter, Bit Error

Figure 516. Master node: transmitter, bit error configuration

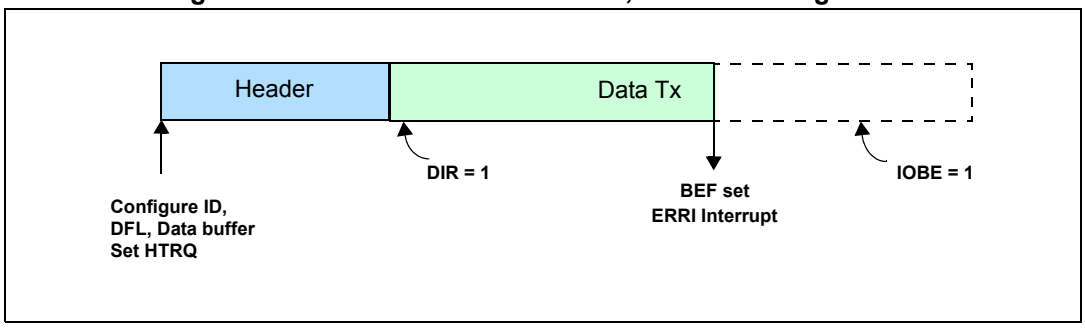
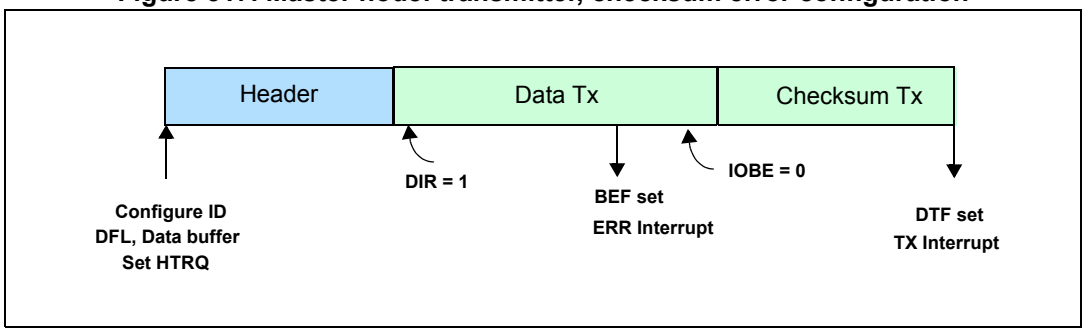
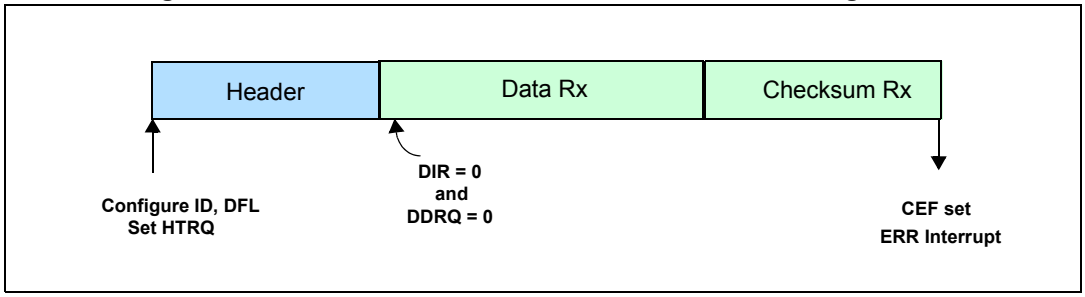


Figure 517. Master node: transmitter, checksum error configuration



42.5.1.4 Receiver, Checksum Error

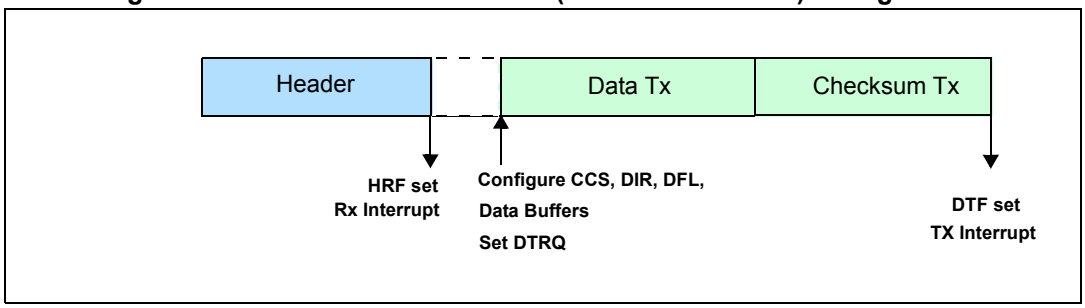
Figure 518. Master node: receiver, checksum error configuration



42.5.2 Slave node

42.5.2.1 Transmitter (no identifier filters)

Figure 519. Slave node: transmitter (no identifier filters) configuration



42.5.2.2 Receiver (no identifier filters)

Figure 520. Slave node: receiver (no identifier filters) configuration

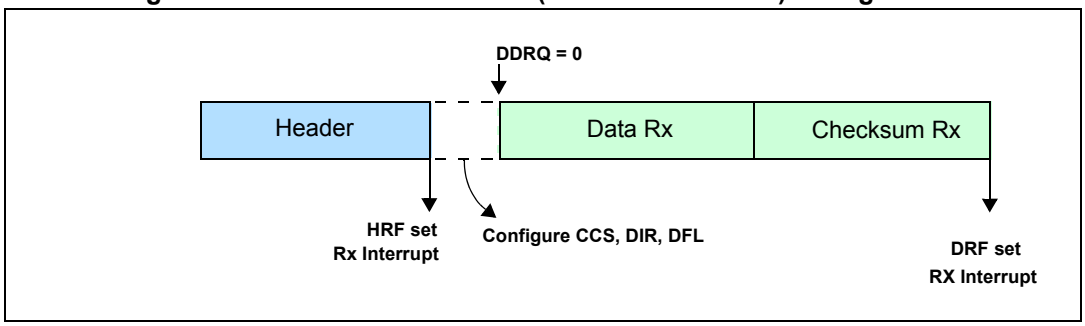
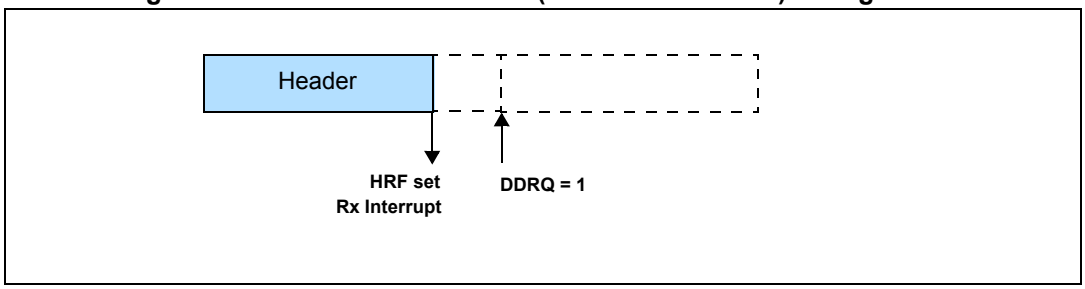
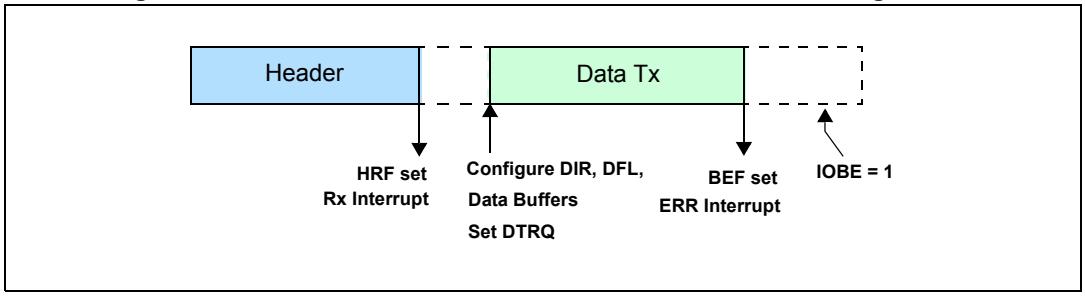


Figure 521. Slave node: receiver (no identifier filters) configuration



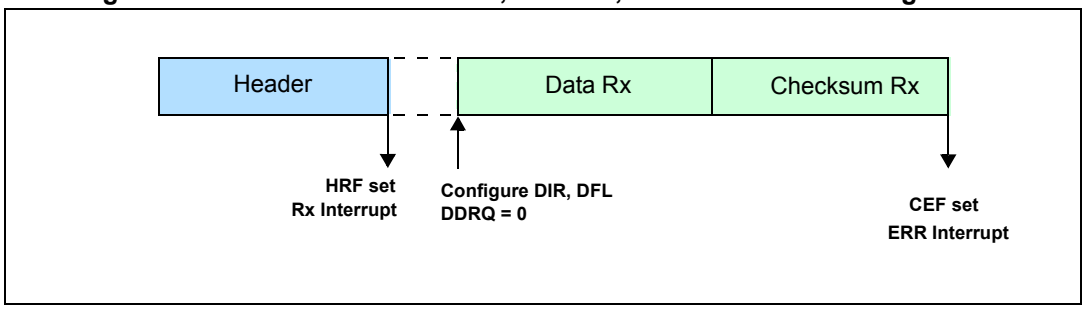
42.5.2.3 No filters, transmitter, bit error

Figure 522. Slave node: No filters, transmitter, bit error configuration



42.5.2.4 No filters, receiver, checksum error

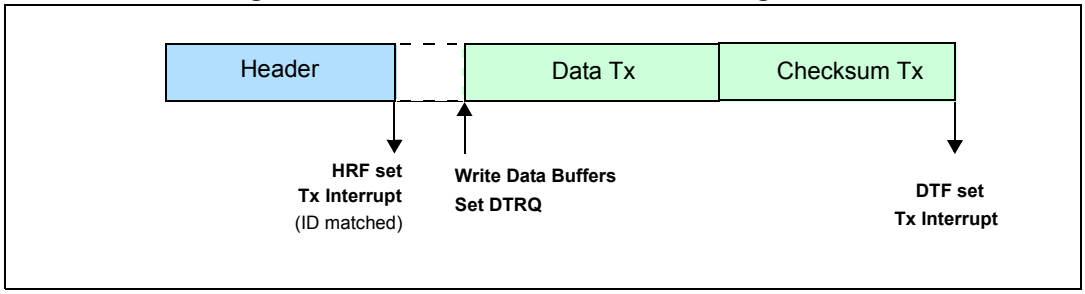
Figure 523. Slave node: No filters, receiver, checksum error configuration



42.5.2.5 At least one TX filter, BF is reset, ID matches filter

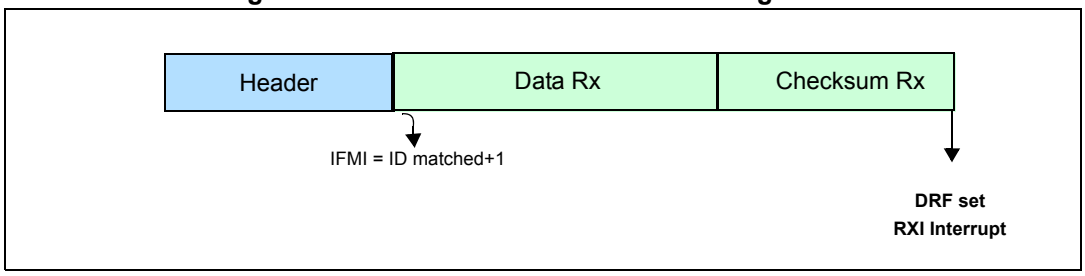
This configuration can be used in case slave never receives data, for example, sensor.

Figure 524. Slave node: one TX filter configuration



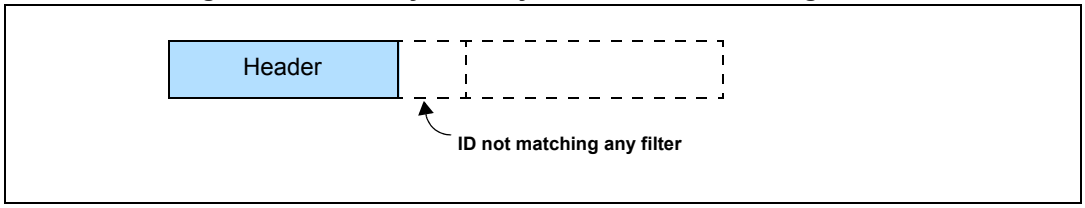
42.5.2.6 At least one RX filter, BF reset, ID matches filter

Figure 525. Slave node: one RX filter configuration



42.5.2.7 RX only, TX only, RX & TX filters, ID not matching filter, BF reset

Figure 526. RX only, TX only, RX & TX filters configuration



42.5.2.8 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 527. TX filter, BF is set configuration: ID has matched

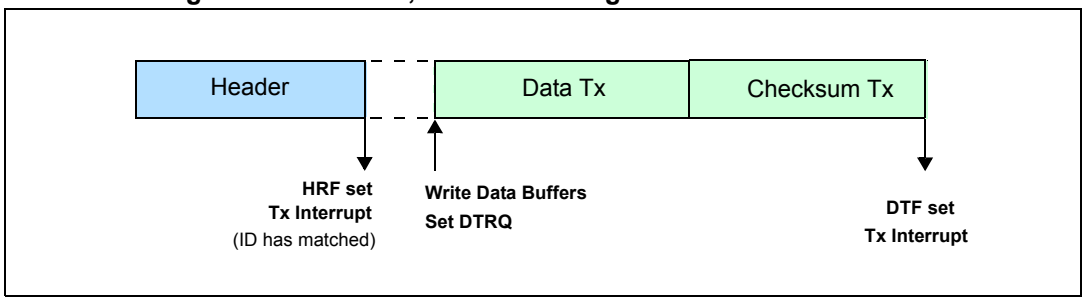
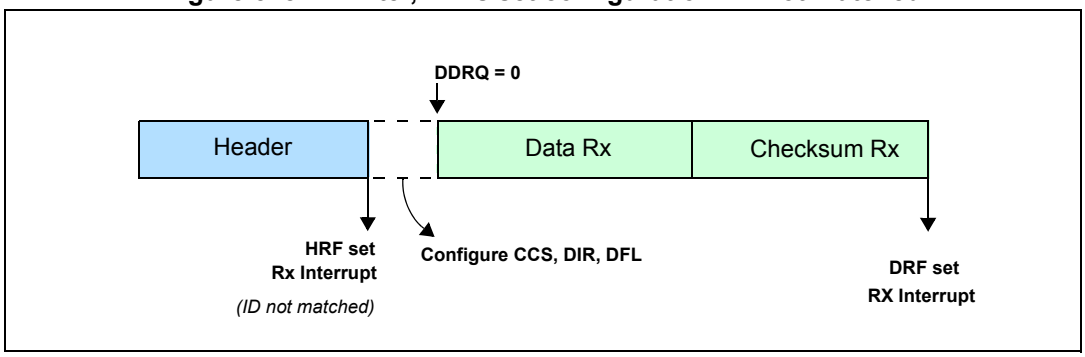


Figure 528. TX filter, BF is set configuration: ID not matched



42.5.2.9 RX filter, BF is set

Figure 529. RX filter, BF is set configuration: ID matched

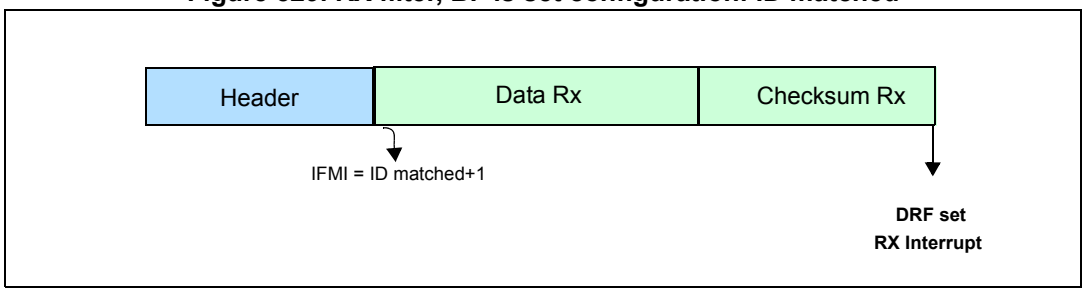


Figure 530. RX filter, BF is set configuration: ID not matched (ID is Rx)

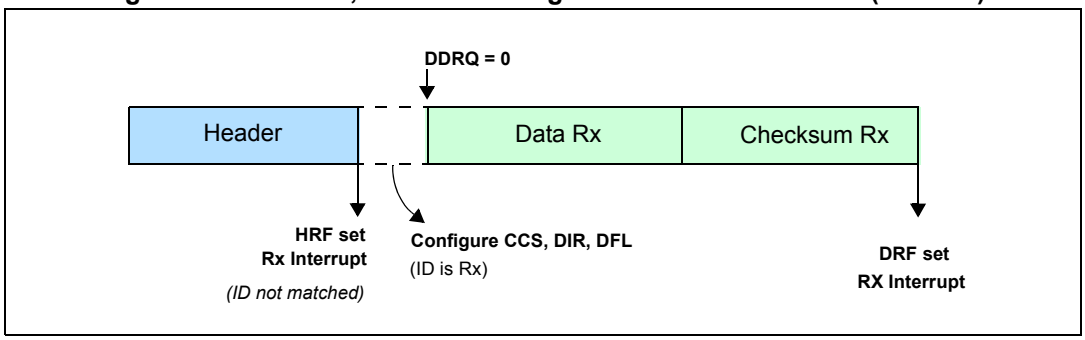
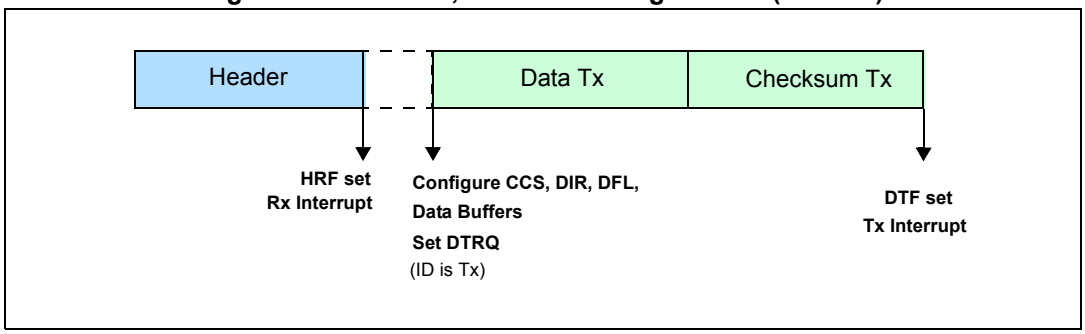


Figure 531. RX filter, BF is set configuration: (ID is Tx)



42.5.2.10 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 532. TX filter, RX filter, BF set configuration: (ID matched)

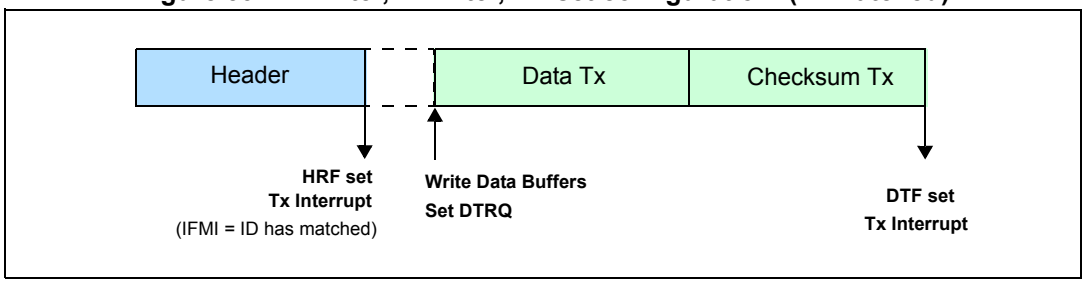


Figure 533. TX filter, RX filter, BF set configuration: (ID matched + 1)

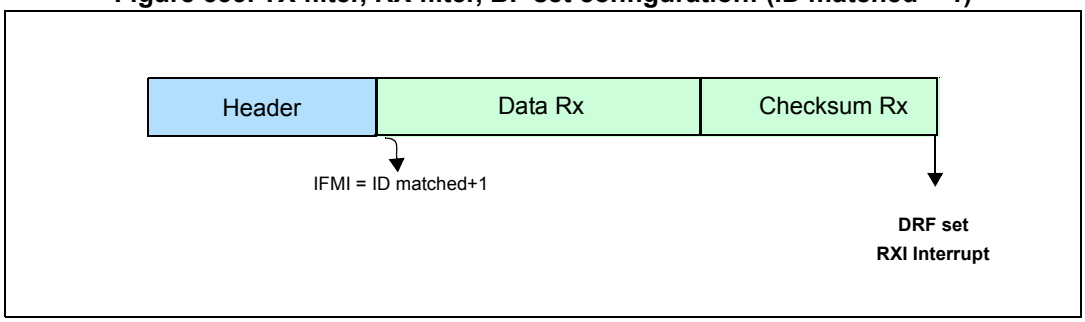
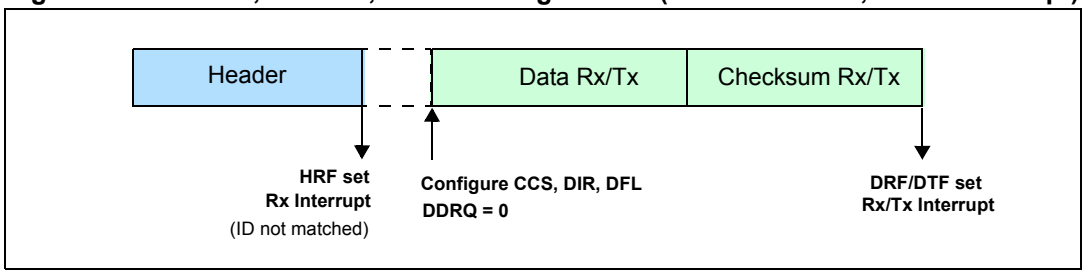


Figure 534. TX filter, RX filter, BF set configuration: (ID not matched, RX/TX Interrupt)



42.5.3 Extended frames

Figure 535. Extended frames (TX Interrupt)

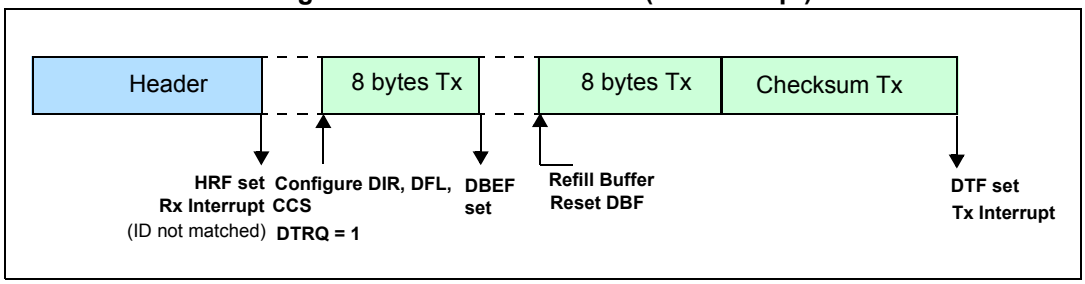
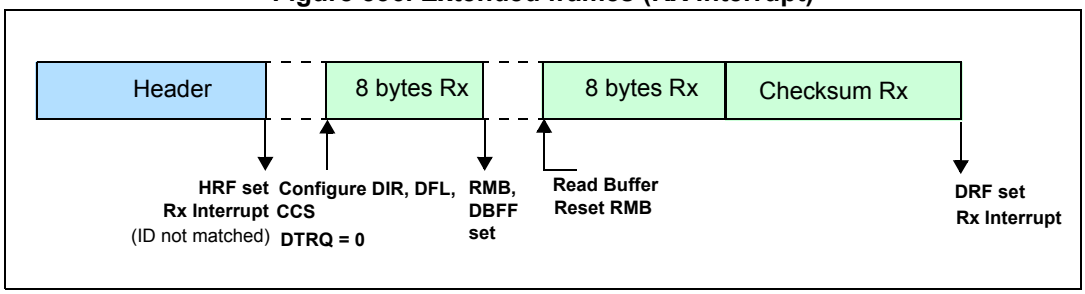


Figure 536. Extended frames (RX Interrupt)



42.5.4 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 537. Response timeout

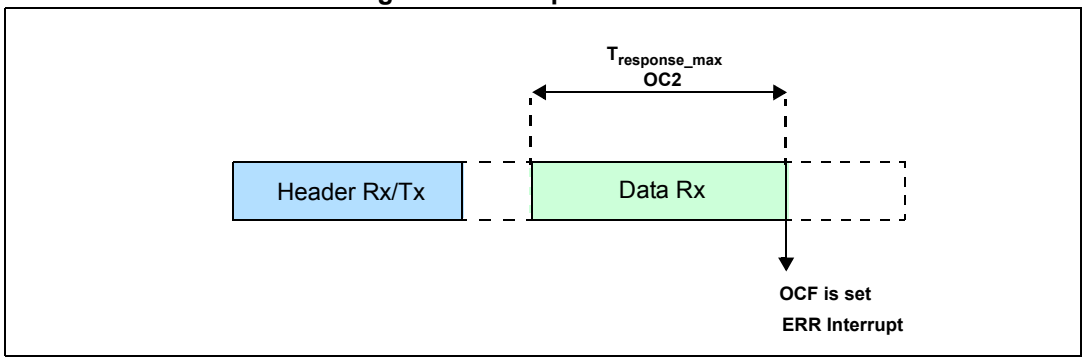
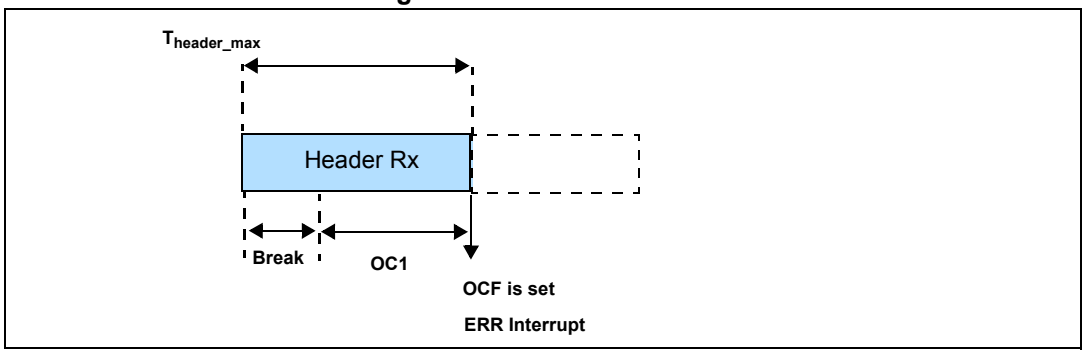
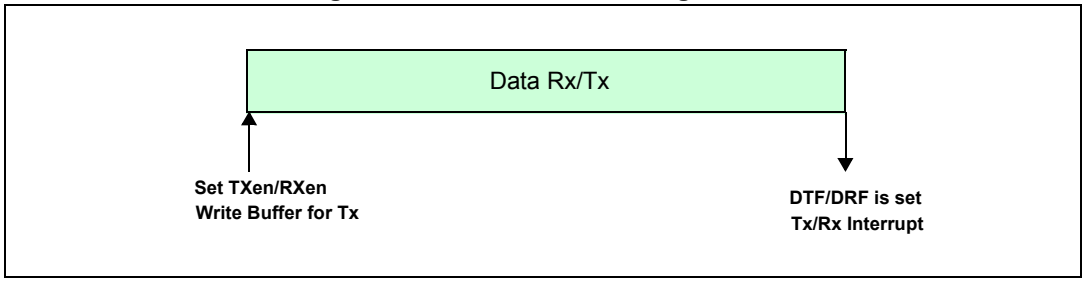


Figure 538. Header timeout



42.5.5 UART mode

Figure 539. UART mode configuration



42.5.6 Interrupts

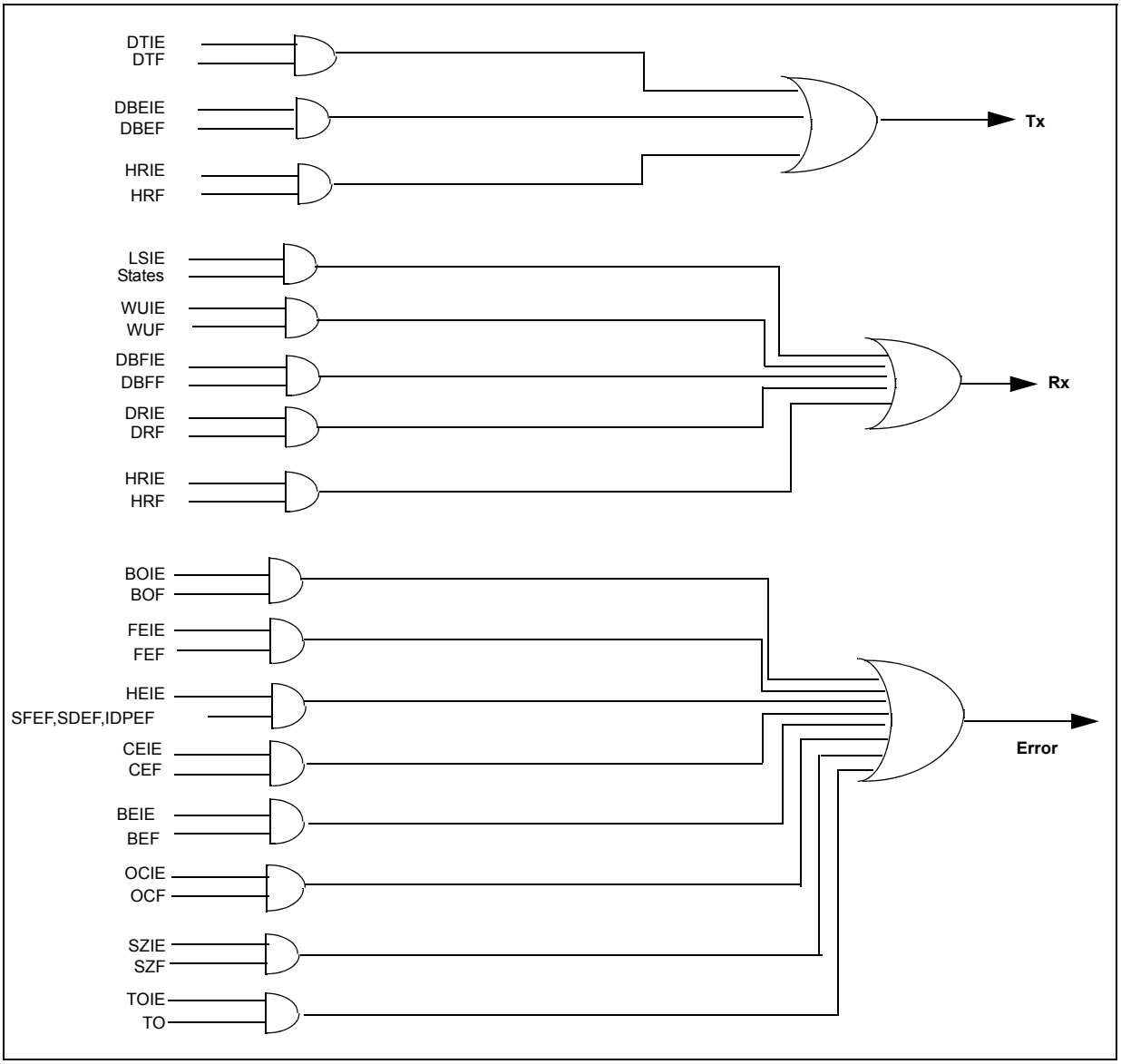
Table 562. Interrupts

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 562. 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 540. Interrupt diagram



43 Reset Generation Module (MC_RGM)

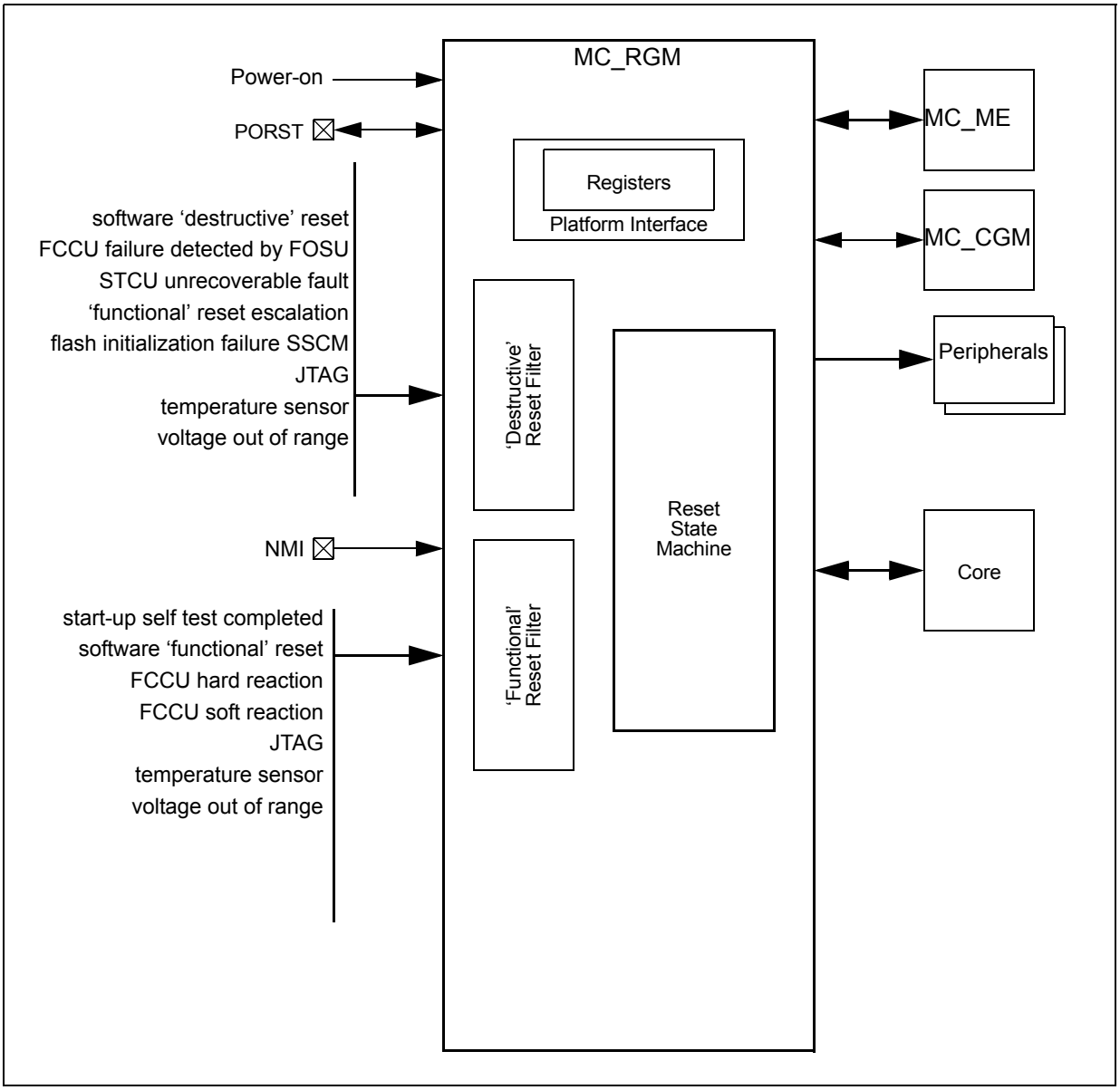
43.1 Introduction

43.1.1 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 541 shows the MC_RGM block diagram.

Figure 541. MC_RGM block diagram



43.1.2 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

43.1.3 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:
  - NMI
  - 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 43.3.2.2: Destructive Event Reset Disable Register (RGM_DERD)* and *Section 43.3.2.3: Destructive Event Alternate Request Register (RGM_DEAR)* for 'destructive' resets. See *Section 43.3.2.6: Functional Event Reset Disable Register (RGM_FERD)* and *Section 43.3.2.7: Functional Event Alternate Request Register (RGM_FEAR)* for 'functional' resets).

43.2 External signal description

The MC_RGM has no external interface signal.

43.3 Memory map and register definition

43.3.1 Memory map

Table 563. 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 43.3.2.1*
0x010	RGM_DERD	'Destructive' Event Reset Disable	word	read	read/write(2)	read/write(2)	*Section 43.3.2.2*
0x020	RGM_DEAR	'Destructive' Event Alternate Request	word	read	read/write	read/write	*Section 43.3.2.3*
0x030	RGM_DBRE	'Destructive' Bidirectional Reset Enable	word	read	read	read	*Section 43.3.2.4*
0x300	RGM_FES	'Functional' Event Status	word	read	read/write(1)	read/write(1)	*Section 43.3.2.5*
0x310	RGM_FERD	'Functional' Event Reset Disable	word	read(2)	read/write(2)	read/write(2)	*Section 43.3.2.6*
0x320	RGM_FEAR	'Functional' Event Alternate Request	word	read	read/write	read/write	*Section 43.3.2.7*
0x330	RGM_FBRE	'Functional' Bidirectional Reset Enable	word	read	read/write	read/write	*Section 43.3.2.8*
0x340	RGM_FESS	'Functional' Event Short Sequence	word	read	read/write	read/write	*Section 43.3.2.9*

Table 563. 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 43.3.2.10*
0x608	RGM_DRET	'Destructive' Reset Threshold	byte	read	read/write	read/write	*Section 43.3.2.11*
0x610	RGM_PRST0	Peripheral Reset 0	word	read	read	read	*Section 43.3.2.11*
0x614	RGM_PRST1	Peripheral Reset 1	word	read	read	read	*Section 43.3.2.13*
0x618	RGM_PRST2	Peripheral Reset 2	word	read	read	read	*Section 43.3.2.14*
0x61C	RGM_PRST3	Peripheral Reset 3	word	read	read	read	*Section 43.3.2.15*
0x620	RGM_PRST4	Peripheral Reset 4	word	read	read	read	*Section 43.3.2.16*
0x624	RGM_PRST5	Peripheral Reset 5	word	read	read	read	*Section 43.3.2.17*
0x628	RGM_PRST6	Peripheral Reset 6	word	read	read	read	*Section 43.3.2.18*
0x62C	RGM_PRST7	Peripheral Reset 7	word	read	read	read	*Section 43.3.2.19*
0x630	RGM_PSTAT0	Peripheral Reset Status 0	word	read	read	read	*Section 43.3.2.20*
0x634	RGM_PSTAT1	Peripheral Reset Status 1	word	read	read	read	*Section 43.3.2.21*
0x638	RGM_PSTAT2	Peripheral Reset Status 2	word	read	read	read	*Section 43.3.2.22*
0x63C	RGM_PSTAT3	Peripheral Reset Status 3	word	read	read	read	*Section 43.3.2.23*
0x640	RGM_PSTAT4	Peripheral Reset Status 4	word	read	read	read	*Section 43.3.2.24*
0x644	RGM_PSTAT5	Peripheral Reset Status 5	word	read	read	read	*Section 43.3.2.25*
0x648	RGM_PSTAT6	Peripheral Reset Status 6	word	read	read	read	*Section 43.3.2.26*
0x64C	RGM_PSTAT7	Peripheral Reset Status 7	word	read	read	read	*Section 43.3.2.27*

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

43.3.2 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.

43.3.2.1 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	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 542. Destructive Event Status Register (RGM_DES)

Table 564. 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 detected by FOSU 0 No FCCU failure detected by FOSU event has occurred since either the last clear or the last power-on reset assertion 1 An FCCU failure detected by FOSU 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 48: Mode Entry Module (MC_ME)*)

43.3.2.2 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 43.3.2.3: 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 543. Destructive Event Reset Disable Register (RGM_DERD)

Table 565. 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 565. 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 detected by FOSU 0 A FCCU failure detected by FOSU 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 48: Mode Entry Module (MC_ME)*)

43.3.2.3 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 544. Destructive Event Alternate Request Register (RGM_DEAR)

Table 566. 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

43.3.2.4 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 545. Destructive Bidirectional Reset Enable Register (RGM_DBRE)

Table 567. 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 567. 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 detected by FOSU 0 PAD_47 is asserted on FCCU failure detected by FOSU 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

43.3.2.5 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	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 546. Functional Event Status Register (RGM_FES)

Table 568. 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.
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.

Table 568. Functional Event Status Register (RGM_FES) field descriptions(Continued)

Field	Description
28 F_SOFT_FUNC	Flag for Software 'functional' reset(1) 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.
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 48: Mode Entry Module (MC_ME)*)
2. The software reset is generated by the NMI

43.3.2.6 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 43.3.2.7: 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 547. Functional Event Reset Disable Register (RGM_FERD)

Table 569. 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 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 569. 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

43.3.2.7 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 548. Functional Event Alternate Request Register (RGM_FEAR)

Table 570. 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

43.3.2.8 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 549. Functional Bidirectional Reset Enable Register (RGM_FBRE)

Table 571. 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 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.

43.3.2.9 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 550. Functional Event Short Sequence Register (RGM_FESS)

Table 572. 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 572. RGM_FESS register field descriptions(Continued)

Field	Description
28 SS_SOFT_FUNC	Short Sequence for Software 'functional' reset 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

43.3.2.10 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 43.4.5* 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 551. Functional Reset Escalation Threshold Register (RGM_FRET)

Table 573. 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

43.3.2.11 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 43.4.6* 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 552. Destructive Reset Escalation Threshold Register (RGM_DRET)

Table 574. 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

43.3.2.12 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 575* 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 553. Peripheral Reset Register 0 (RGM_PRST0)

Table 575. 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

43.3.2.13 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 576* 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	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	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 554. Peripheral Reset Register 1 (RGM_PRST1)

Table 576. RGM_PRST1 field descriptions

Field	Description
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

43.3.2.14 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 577* 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	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	Flexcan_0_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 555. Peripheral Reset Register 2 (RGM_PRST2)

Table 577. 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
16 Flexcan_0_RST	Flexcan_0 reset Writing a ‘1’ to this bit resets the Flexcan_0 peripheral. Writing a ‘0’ to this bit releases the reset to the Flexcan_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

43.3.2.15 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 578* 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_seq_0_RST	0	0	0	0	0	0	0	0	0	0	0	0	ADCSAR_seq_b_RST	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	0	0	0	0	0	0	0	0	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 556. Peripheral Reset Register 3 (RGM_PRST3)

Table 578. 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. 0 No forced reset of peripheral 1 Forced reset of peripheral
1 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
14 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
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. 0 No forced reset of peripheral 1 Forced reset of peripheral

Table 578. RGM_PRST3 register field descriptions(Continued)

Field	Description
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

43.3.2.16 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 579* 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	0	0	0	0	0	0	0	0	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 557. Peripheral Reset Register 4 (RGM_PRST4)

Table 579. RGM_PRST4 register field descriptions

Field	Description
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

43.3.2.17 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 580* 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	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	FCCU_RST	0	0	0	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 558. Peripheral Reset Register 5 (RGM_PRST5)

Table 580. RGM_PRST5 register field descriptions

Field	Description
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

43.3.2.18 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 581* 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	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	Flexcan_1_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 559. Peripheral Reset Register 6 (RGM_PRST6)

Table 581. 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
16 Flexcan_1_RST	Flexcan_1 reset Writing a '1' to this bit resets the Flexcan_1 peripheral. Writing a '0' to this bit releases the reset to the Flexcan_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

43.3.2.19 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 582* 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	0	0	0	0	0	0	0	0	0	0	0	CTU_RST	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	DSPI_2_RST	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 560. Peripheral Reset Register 7 (RGM_PRST7)

Table 582. RGM_PRST7 register field descriptions

Field	Description
11 CTU_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
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

Caution: Use the ***RGM_PRSTn*** registers with care. See *Section 43.4.7* for the proper sequence to prevent unexpected chip behavior.

43.3.2.20 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 583* 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 561. Peripheral Reset Status Register 0 (RGM_PSTAT0)

Table 583. 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

43.3.2.21 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 584* 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	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	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 562. Peripheral Reset Status Register 1 (RGM_PSTAT1)

Table 584. RGM_PSTAT1 register field descriptions

Field	Description
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

43.3.2.22 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 585* 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	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	Flexcan_0_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 563. Peripheral Reset Status Register 2 (RGM_PSTAT2)

Table 585. 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
16 Flexcan_0_STAT	Flexcan_0 reset status 0 peripheral is not in reset 1 peripheral is in reset

43.3.2.23 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 586* 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_seq_0_STAT	0	0	0	0	0	0	0	0	0	0	0	0	ADCSAR_seq_b_STAT	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	0	0	0	0	0	0	0	0	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 564. Peripheral Reset Status Register 3 (RGM_PSTAT3)

Table 586. 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_seq_0_STAT	ADCSAR_seq_0 reset status 0 peripheral is not in reset 1 peripheral is in reset
14 ADCSAR_seq_b_STAT	ADCSAR_seq_b 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
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

43.3.2.24 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 587* 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	0	0	0	0	0	0	0	0	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 565. Peripheral Reset Status Register 4 (RGM_PSTAT4)

Table 587. RGM_PSTAT4 register field descriptions

Field	Description
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

43.3.2.25 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 588* 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	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	FCCU_STAT	0	0	0	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 566. Peripheral Reset Status Register 5 (RGM_PSTAT5)

Table 588. RGM_PSTAT5 register field descriptions

Field	Description
22 FCCU_STAT	FCCU reset status 0 peripheral is not in reset 1 peripheral is in reset

43.3.2.26 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 589* 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	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	Flexcan_1_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 567. Peripheral Reset Status Register 6 (RGM_PSTAT6)

Table 589. 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
16 Flexcan_1_STAT	Flexcan_1 reset status 0 peripheral is not in reset 1 peripheral is in reset

43.3.2.27 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 590* 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	0	0	0	0	0	0	0	0	0	0	0	CTU_STAT	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	DSPI_2_STAT	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 568. Peripheral Reset Status Register 7 (RGM_PSTAT7)

Table 590. RGM_PSTAT7 register field descriptions

Field	Description
11 CTU_STAT	CTU 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

43.4 Functional description

43.4.1 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 591*.

Table 591. 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

Table 591. MC_RGM reset implications(Continued)

Source	What is reset	External reset assertion	Boot mode capture
'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

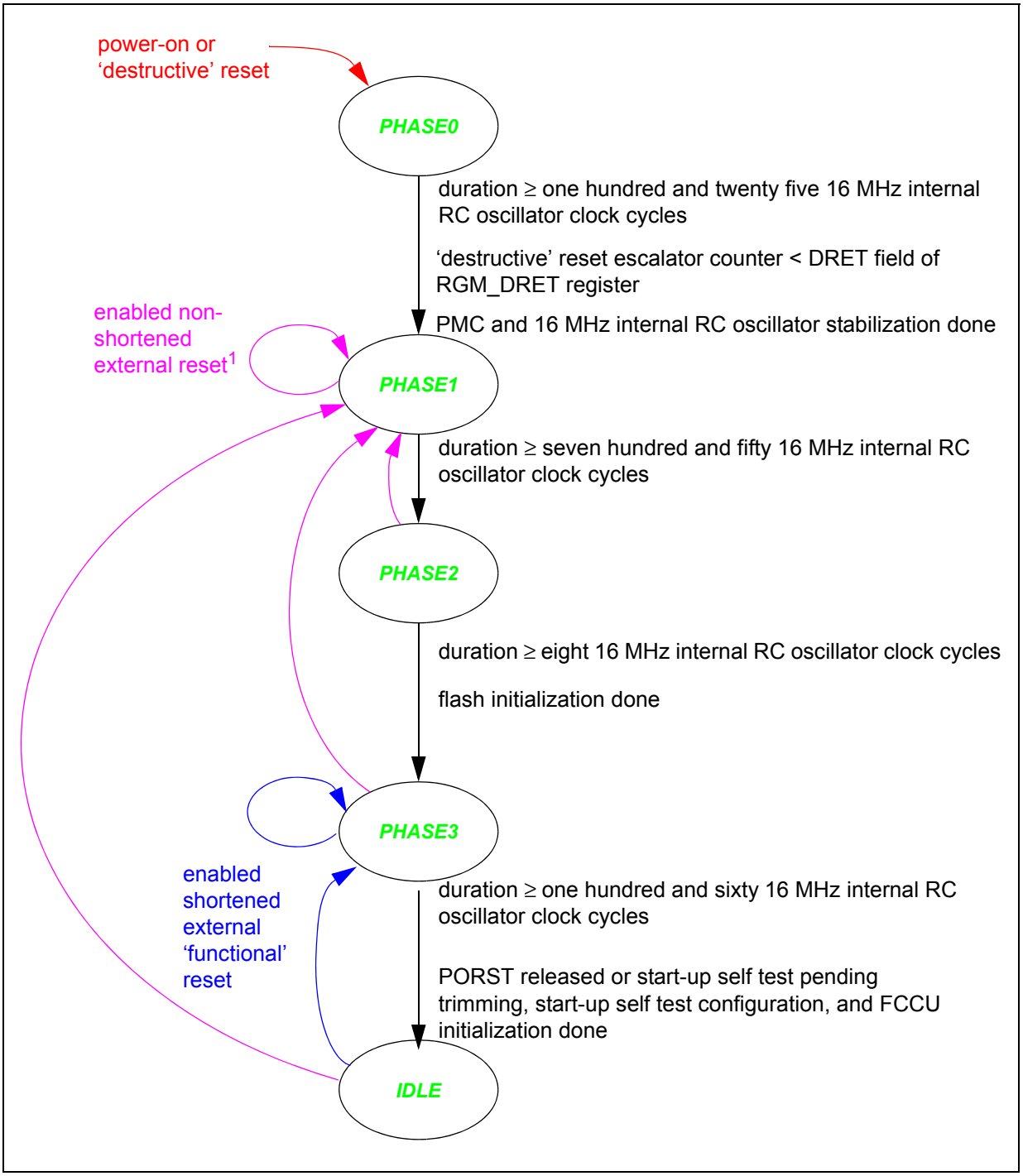
1. the boot mode is captured if the external reset is asserted.
2. the short sequence is enabled via the RGM_FESS register.

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 569*.

Figure 569. MC_RGM State Machine



43.4.1.1 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

43.4.1.2 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

43.4.1.3 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

43.4.1.4 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

43.4.1.5 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.

43.4.2 ‘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 and/or HVD) 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.

43.4.2.1 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.

43.4.3 ‘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.

43.4.4 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 592*.

Table 592. 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.

43.4.5 ‘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.

43.4.6 '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.

43.4.7 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.

44 Boot Assist Flash (BAF)

44.1 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 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 FlexCAN module. The package pins used by LINFlexD are the same pins used by the FlexCAN module. Hence, the external PHY can be either LIN or CAN (see *Section 44.3.6* for configuration details).

44.2 Memory map

For the BAF location in flash, refer to the Embedded memories chapter. The BAF block base address is 0040_4000h. 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 593*.

Table 593. BAF memory organization

BAF Component	Address
BAF image header	0040_4000h
BAF code entry point	0040_4100h
BAF data area and stack	4000_0000h-4000_01FFh

44.2.1 BAF image header

BAF image contains a 256-byte header starting from address 0040_4000h. The BAF header is explained in *Table 594*.

Table 594. BAF image header

Address offset	Size (bits)	Field description
0000h	32	SSCM boot record tag
0004h	32	BAF start address
0010h	32	BAF version number
TBD–00FFh	Reserved	

44.2.1.1 BAF image version

The BAF version is a 32-bit field in the image header starting at address 0040_4000h. The 32-bit field is explained in the following table.

Table 595. 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

44.3 Functional description

A functional description of the BAF modes is contained in the following sections.

44.3.1 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 FlexCAN 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.

44.3.2 Device initialization

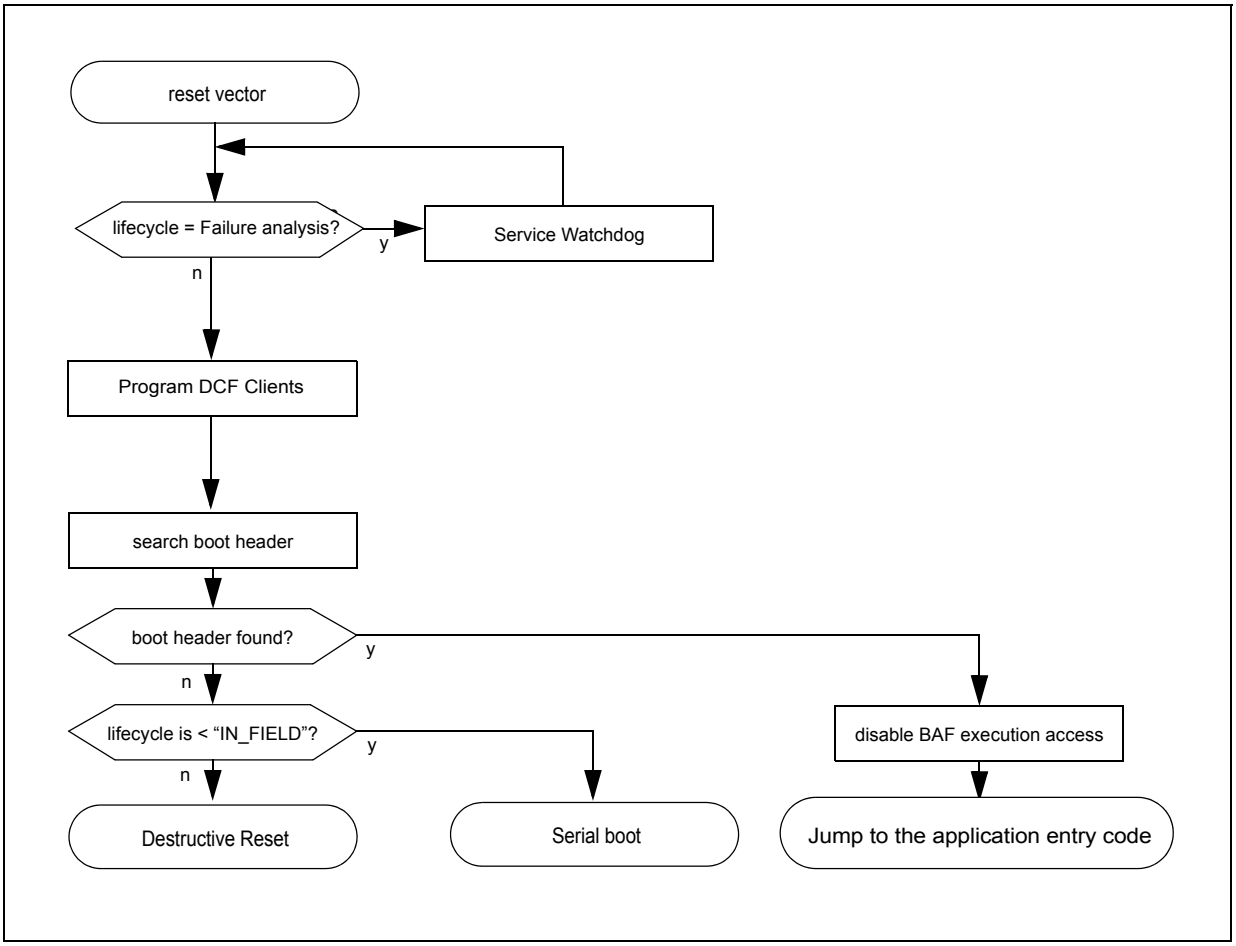
The BAF performs a minimal initialization of core registers and clocks required to enter the above described boot modes, then attempts to interact in the following way:

- 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 593: 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.

44.3.3 Flow of control

Figure 570 shows the overall flow of control in the BAF. The individual steps are explained in the following sections.

Figure 570. Flow of control



44.3.4 Initialization of BAF DCF clients

44.3.4.1 ‘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-bit chip select field (selects the module which implements the DCF clients)
- 15-bit 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 FFFF_FFFFh). 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 596*.

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 596* 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 44.3.4.5: SER_BOOT_CBACK – BAF serial boot DCF client*.

Table 596. BAF DCF client list

Address (hex)	Name	Label	Description
0000h–01FFh	Reserved		
0200h	Address	ADDR	The address to which DATAX is written
0201h	32-bit Data	DATA32	The 32-bit data that is written to ADDR
0202h	16-bit Data	DATA16	The 16-bit data that is written to ADDR
0203h	8-bit Data	DATA8	The 8-bit data that is written to ADDR
0204h–020Fh	Reserved		
0210h	Callback Address	CALLBACK	The address of a callback function
0211h	Serial boot Callback Address	SER_BOOT_CBACK	The address of serial boot callback function
0212h – 0213h	Reserved	–	Reserved for future use.

Table 596. BAF DCF client list(Continued)

Address (hex)	Name	Label	Description
0214h	Reserved	–	Reserved for future use.
0215h – 7FFFh	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.

44.3.4.2 BAF DCF client - ADDRESS

BAF parses the DCF records. If BAF finds a DCF record with CS14 set and Address field as 0200h 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.

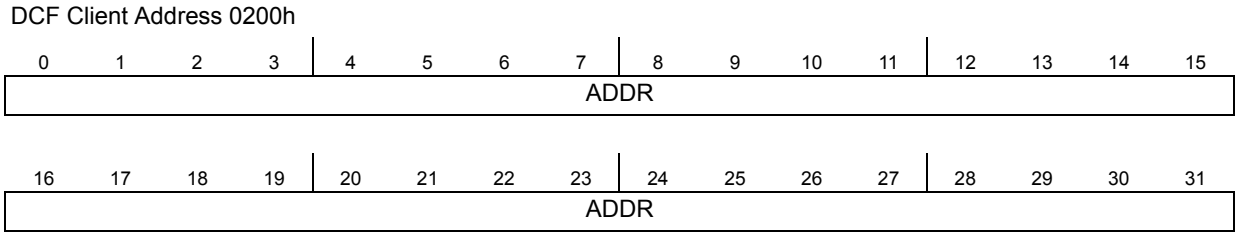


Figure 571. BAF DCF ADDR client configuration

Table 597. 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.

44.3.4.3 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.

44.3.4.3.1 DATA32 – 32-bit data DCF client

BAF parses the DCF records. If BAF finds a record with CS14 set and address field set as 0201h 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.

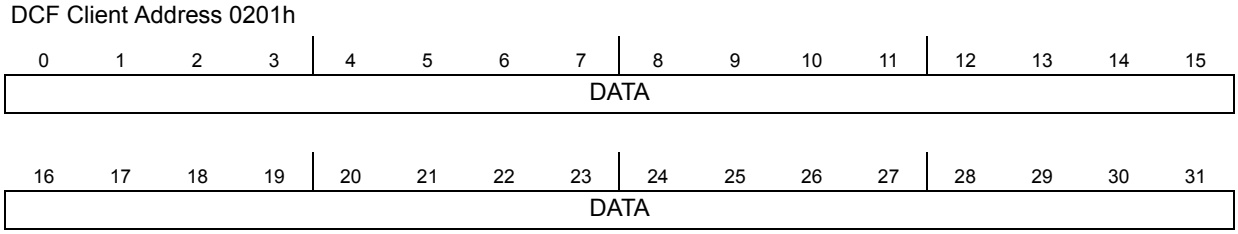


Figure 572. DATA32 DCF client configuration

Table 598. DATA32 DCF client configuration

Field	Description
0:31 DATA	32-Bit DCF data

44.3.4.3.2 DATA16 – 16-bit data DCF client

BAF parses the DCF records. If BAF finds a record with CS14 set and address field set as 0202h 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.

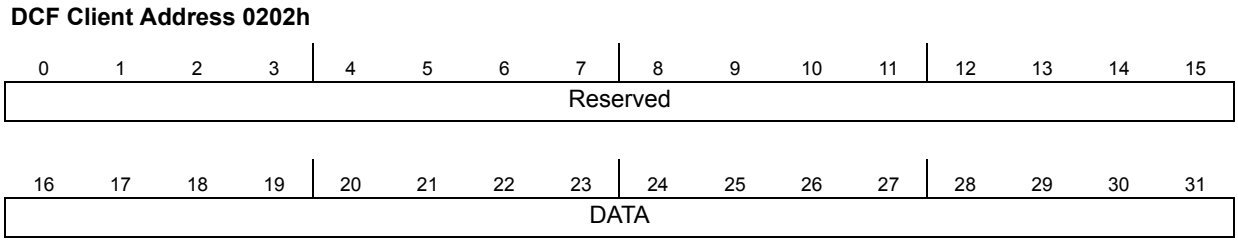


Figure 573. DATA16 DCF Client Configuration

Table 599. DATA16 DCF client configuration

Field	Description
16:31 DATA	16-Bit DCF data

44.3.4.3.3 DATA8 – 8-bit DCF client

BAF parses the DCF records. If BAF finds a record with CS14 set and address field set as 0203h 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.

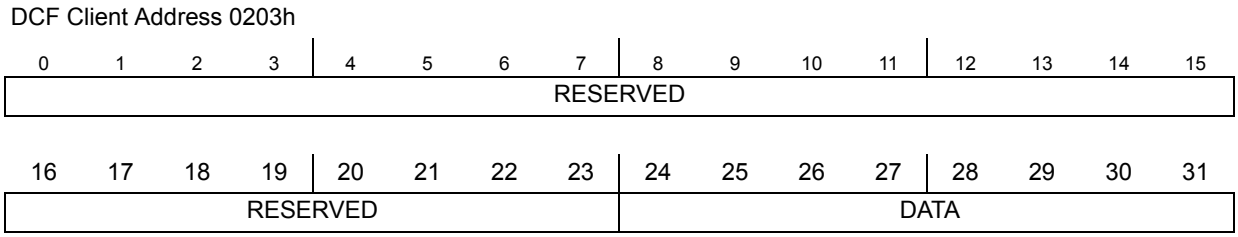


Figure 574. DATA8 DCF Client Configuration

Table 600. DATA8 DCF client configuration

Field	Description
24:31 DATA	8-Bit DCF data

44.3.4.4 CALLBACK – BAF callback DCF client

BAF parses the DCF records. If BAF finds a record with CS14 set and address field as 0210h 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.

Warning: 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.

44.3.4.5 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 0211h 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 601* for details regarding return value).

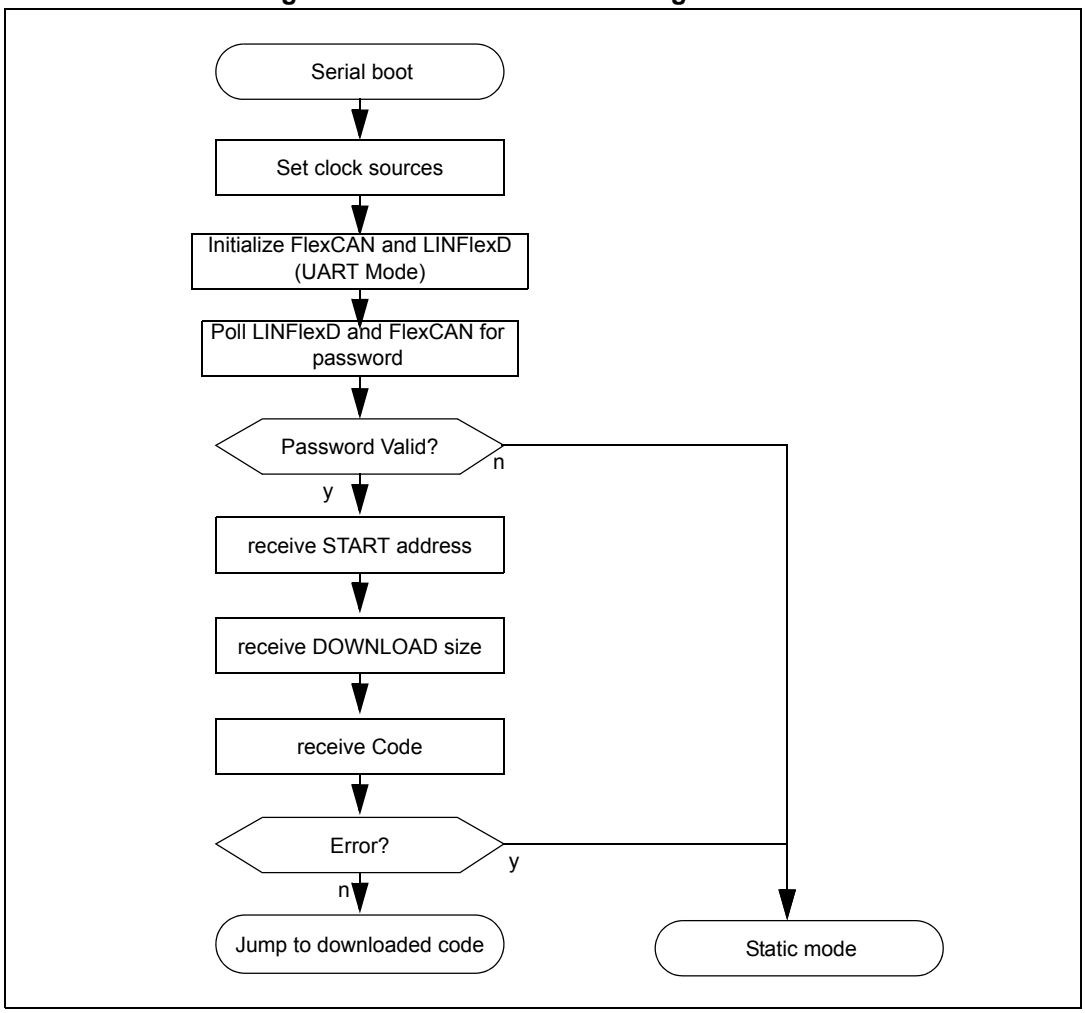
Table 601. Serial boot callback function return value definition

Return values	Description
00h	Boot from internal flash using standard search for boot record (see 'Search boot header and boot options' in *Chapter 7: Device configuration*).
01h	Serial boot using both FlexCAN 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 570*

44.3.5 Optionally perform a serial boot

If no boot header is found in one of the locations mentioned above or customer selects serial boot using serial boot DCF client, then BAF determines the lifecycle status of the device. If the Life Cycle is greater than Infield, it attempts a serial boot. Otherwise, the boot has failed and the BAF issues a destructive reset. The BAF is executed via the XOSC as clock source.

Figure 575. Flow of control during serial boot



The clock source in the MC_CGM is configured and the PADs are set up for LINFlexD and FlexCAN signals, then the code is downloaded using LINFlexD or FlexCAN 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 FlexCAN 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 FlexCAN 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 11h is received on the FlexCAN 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 11h is received by FlexCAN controller or a FlexCAN 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 FEh of the password. If not continue polling. The data may be a valid CAN message.
- If it is first byte of the password (FEh) then disable the FlexCAN controller and restore to its reset state.
- Configures the FlexCAN pads to their reset state.
- Transition to LINFlexD serial boot (UART mode).

44.3.5.1 Serial boot mode protocol

From a high-level perspective, the download protocol over FlexCAN and LIN interfaces follows steps below:

6. Host transmits the 64-bit password to MCU.
7. 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

8. Host transmits a sequence of data bytes that are to be executed.
9. 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 FlexCAN are slightly different.

44.3.5.2 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.).

44.3.5.3 FlexCAN download protocol

The FlexCAN download protocol uses very similar structure to the LINFlexD protocol, but it makes use of the message ID's provided by FlexCAN.

The tool sends data packed into standard CAN message in the following manner:

1. A message with 11h as the ID and an 8-byte length to send the password. The MCU transmits the same message, but with an ID of 1h.
2. A message with 12h 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 2h.
3. A message with 13h 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 3h.

44.3.5.4 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 FEED_FACE_CAFE_BEEFh.

- 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.

44.3.5.5 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 576* and *Figure 577*.

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.

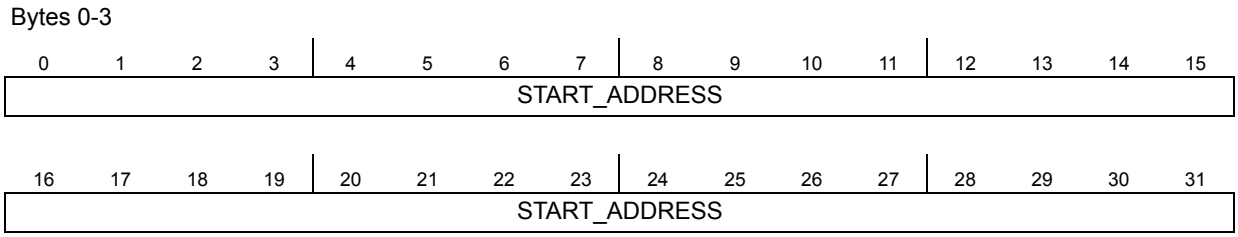


Figure 576. Start address

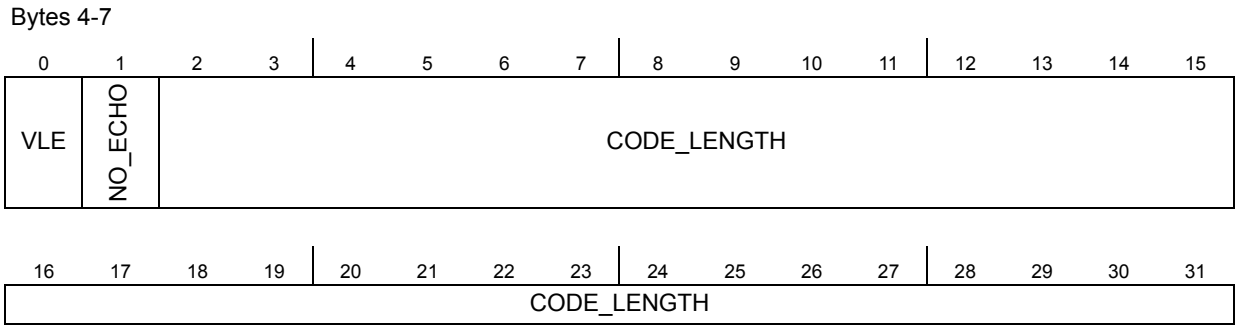


Figure 577. VLE bit, NO_ECHO bit and code size

44.3.5.6 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 (0000_0000_0000_0000h) to the next address location. This is written to avoid possible ECC error during core prefetch.

44.3.5.7 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.

44.3.6 Serial boot configuration

44.3.6.1 LINFlexD configuration

Boot using UART protocol is implemented by LINFlexD module.

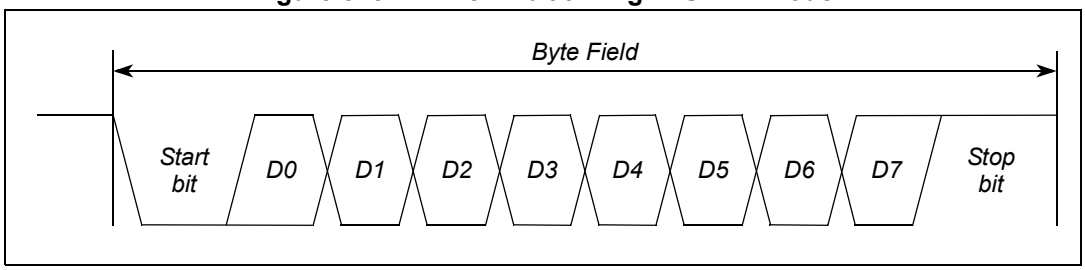
Note: See the pin configuration table in Chapter 7: Device configuration for pins used by the LINFlexD module.

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 578. LINFlexD bit timing in UART mode

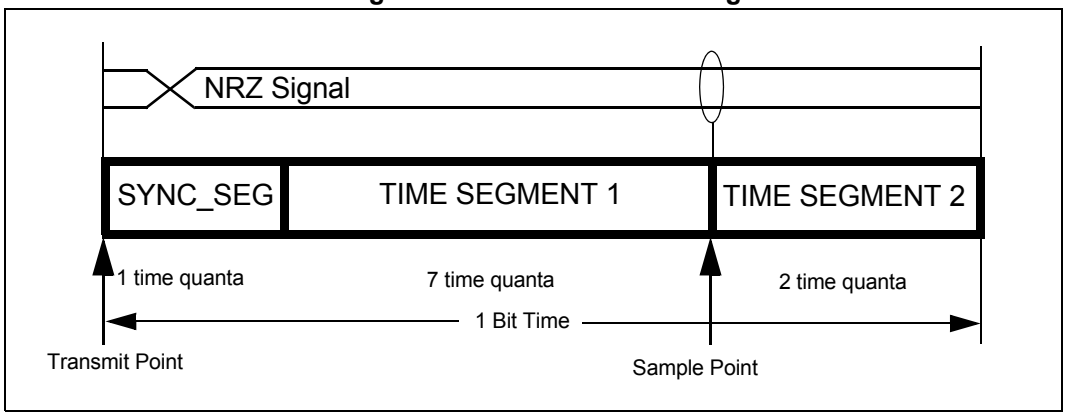


44.3.6.2 FlexCAN configuration

The FlexCAN 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 602* for baud rates).

The Bit timing is configured as shown in *Figure 579*

Figure 579. FlexCAN Bit Timing



44.3.6.3 Baud rates

The LINFlexD baud rate is set by frequency of external crystal oscillator/833.333 . *Table 602* below shows some possible baud rates.

The FlexCAN baud rate is set by the frequency of the external crystal/40.

Table 602. LinFlexD_0 and FlexCAN_1 baud rates

Frequency	LINFlexD baud rate	FlexCAN 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

44.3.6.4 LINFlexD and FlexCAN pin configuration

Please see *Chapter 7: Device configuration* for pin details.

44.3.6.5 Protocol summary

Table 603 summarizes the LINFlexD protocol and BAF action during this boot mode.

Table 603. 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 603. 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 604 shows the FlexCAN serial boot protocol.

Table 604. FlexCAN serial boot mode download protocol

Protocol steps	Host message sent	BAF response message	Action
1	CAN ID 011h + 64-bit password	CAN ID 001h + 64-bit password	Password checked against fixed password (FEED_FACE_CAFE_BEEFh). Watchdog timer is refreshed if the password check is successful.
2	CAN ID 012h + 32-bit store address + 32-bit (1 VLE bit, 1 NO_ECHO bit and 30 of download data size in bytes)	CAN ID 002h + 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 013h + 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

44.4 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 FlexCAN

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 (see *Section 44.3.5*).

- 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)

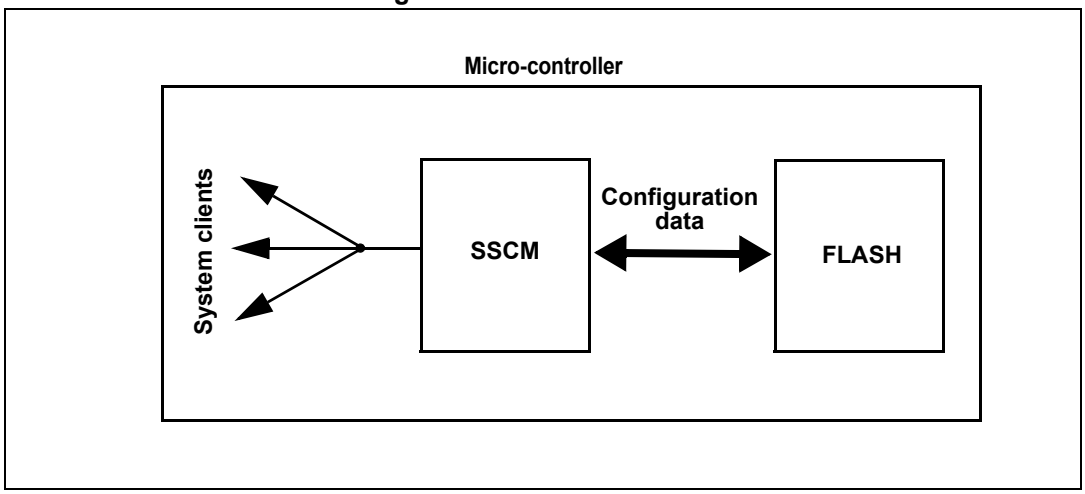
45 System Status and Configuration Module (SSCM)

45.1 Introduction

45.1.1 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 580*.

Figure 580. SSCM overview



45.1.2 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

45.1.3 Modes of operation

The SSCM operates identically in all system modes.

45.2 Memory map and register definition

This section provides a detailed description of all memory-mapped registers in the SSCM.

Table 605 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 605. SSCM memory map

Offset from SSCM_BASE	Register	Access	Reset Value	Section/ page
0x0000	SSCM System Status (STATUS)	R/W	0xyyy0(1)	*Section 45.2.1.1*
0x0002	SSCM System Memory and ID Register (MEMCONFIG)	R	0xyyyy(1)	*Section 45.2.1.2*
0x0004–0x0005	Reserved			
0x0006	SSCM Error Configuration Register (ERROR)	R/W	0x0000	*Section 45.2.1.3*
0x0008–0x0009	Reserved(2)			
0x000A–0x001C	Reserved			
0x0020–0x0023	Reserved			
0x0024–0x0027	Reserved			
0x0028	Processor Start Address Register (PSA)	R	0xyyyy_yyyy(1)	*Section 45.2.1.4*
0x002C–0x002F	Reserved			
0x0030–0x0033	Reserved			
0x0034	Life Cycle Status Register (LCSTAT)	R	0x0000_0y0y(1)	*Section 45.2.1.5*
0x0038–0x0048	Reserved			

1. See register description for reset value details.
2. No register abort on this location.

45.2.1 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.

45.2.1.1 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 581. System Status Register (STATUS)

Table 606. 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 Reserved 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

45.2.1.2 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 (see the Device Configuration chapter)

Figure 582. System memory and ID register (MEMCONFIG)

Table 607. MEMCONFIG field descriptions

Field	Description
0:9 JPIN	JTAG Part ID Number
11:14 MREV	Minor Mask Revision

45.2.1.3 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	PAE	RAE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 583. Error Configuration Register (ERROR)

Table 608. 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.

45.2.1.4 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 584. Processor Start Address Register (PSA)

Table 609. 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 the case of a serial download, this register will point to the BAM start location.

45.2.1.5 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 585. Life Cycle Status Register (LCSTAT)

Table 610. 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

45.3 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.

45.3.1 DCF mechanism

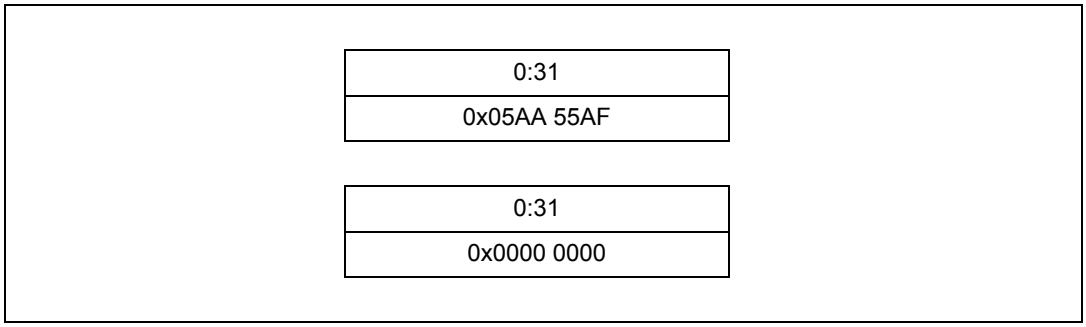
The DCF mechanism has been developed to handle settings of device parameters via OTP flash memory.

Starting at location UTEST_OFFSET[u] 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.

u. Refer to the Device Configuration chapter for UTEST_OFFSET value

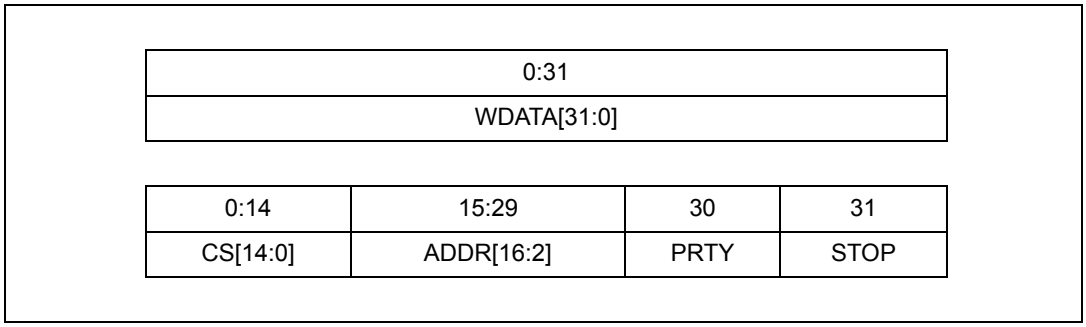
Each record is 64 bits in length. *Figure 586* shows the DCF start record. The start record must be placed at UTEST_OFFSET to indicate to the device that the following data records should be processed.

Figure 586. DCF start record



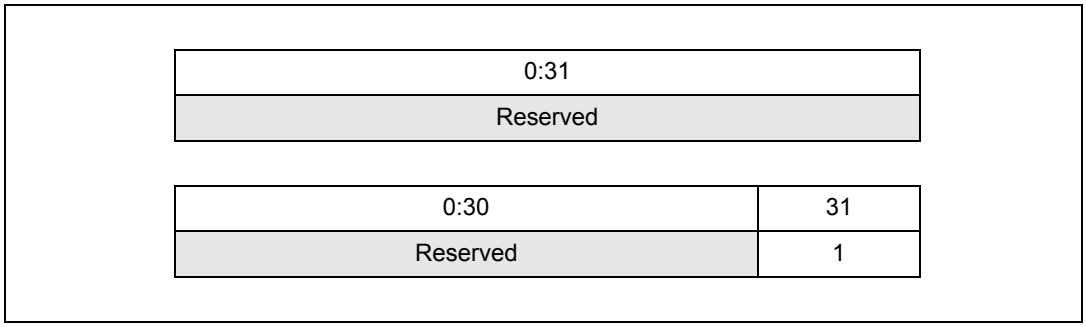
The data records are structured similar to a CPU write instruction - 15 bits of client select, 15 bits of address, 32 bits of payload data plus a STOP bit - See *Figure 587*. In a data record the STOP bit must always be programmed to 0. Some clients support a parity bit.

Figure 587. 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 588*. 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 588. DCF stop record



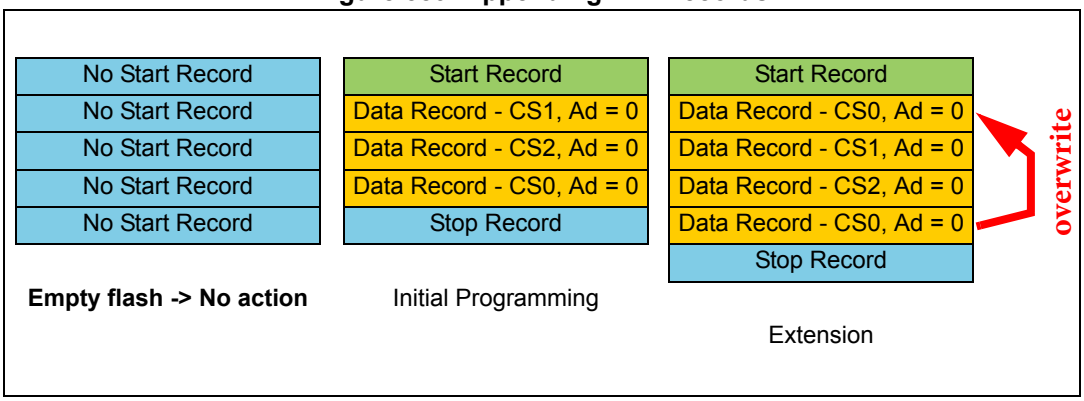
If *n* data records are to be stored in UTEST, the data structure must be as shown in *Table 611*.

Table 611. 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 589*.

Figure 589. 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).

45.3.2 ECC error monitoring

During flash memory scanning (such as Life Cycle or DCF records), the SSCM will monitor the flash memory status signals for ECC and other unexpected access errors. If such an error occurs the SSCM will request a destructive reset.

If ECC error is present during RCHW scanning, customer can check the status of ERR bit in FLASH_MCR register.

45.3.3 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).

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.

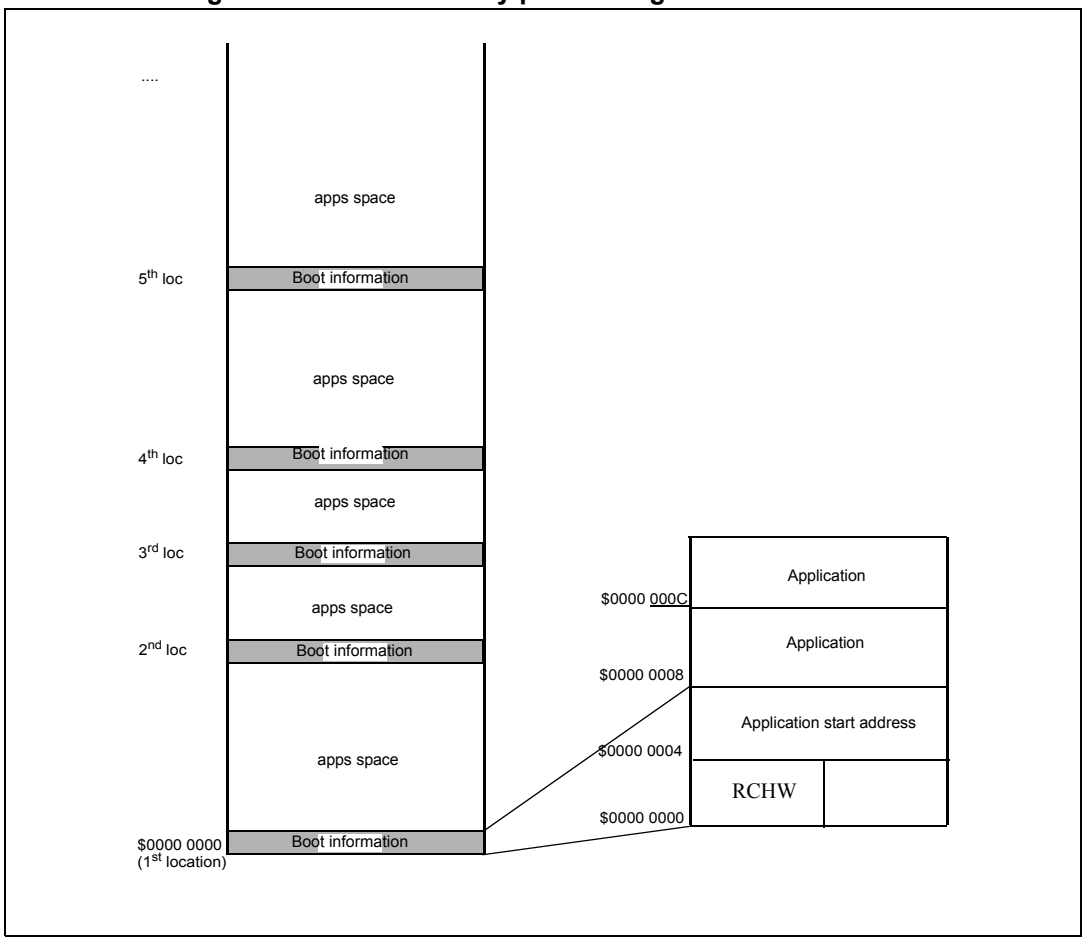
45.3.4 User code boot sector search

45.3.4.1 Potential boot sectors

As shown in *Figure 590* 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 *Section 8.8: HW boot record search (SSCM search)* of the “Reset and Boot” chapter.

Figure 590. Flash memory partitioning and RCHW search



45.3.4.2 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.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	VLE	BOOT_ID							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

= Reserved

Figure 591. 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.

45.3.4.3 Boot and alternate boot sectors

Some applications require an alternate boot sector so that the main boot can be erased and reprogrammed in the field.

When an alternate boot is needed, the user can create two bootable sectors; the lowest sector shall be the main boot sector and the highest shall be the alternate boot sector. The alternate boot sector does not need to be consecutive to the main boot sector.

This scheme ensures that there is always one active boot sector by erasing one of the boot sectors only:

- Sector is activated (that is, program a valid BOOT_ID instead of 0xFF as initially programmed).
- Sector is deactivated writing to 0 some of the bits BOOT-ID bit field (bit 1 and/or bit 3, and/or bit 4, and/or bit 6).

45.3.5 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 612. Life Cycle slots

LC Slot		LC slot value (status)
Valid Field (64 Bits)	**Invalid Field (64 Bits)**	
Erased	Erased	Erased
Marked	Erased	Active
Marked	Marked	Inactive
Any other values		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 613 shows how the slots are arranged in memory.

Table 613. Life Cycle Slots in Memory

Offset	LC Slot word
0x00	Valid[31:0]
0x04	Valid[63:32]
0x08	Invalid[31:0]
0x0C	Invalid[63:32]

The Life Cycle is determined as shown in *Table 614*. The priority of the entries is from top to bottom, so the first row which applies determines the resulting Life Cycle.

Table 614. 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
Inactive	Inactive	Inactive	Inactive	Erased	In Field
Inactive	Inactive	Inactive	Inactive	Not Erased	System Reset

1. Erased = all slot bits at ‘1’

45.4 Initialization and application information

45.4.1 Reset

The reset state of each individual bit is shown within the Register Description section (see *Section 45.2.1*).

45.5 Additional safety measures

45.5.1 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.

46 Power management controller digital interface (PMC_Dig)

46.1 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.

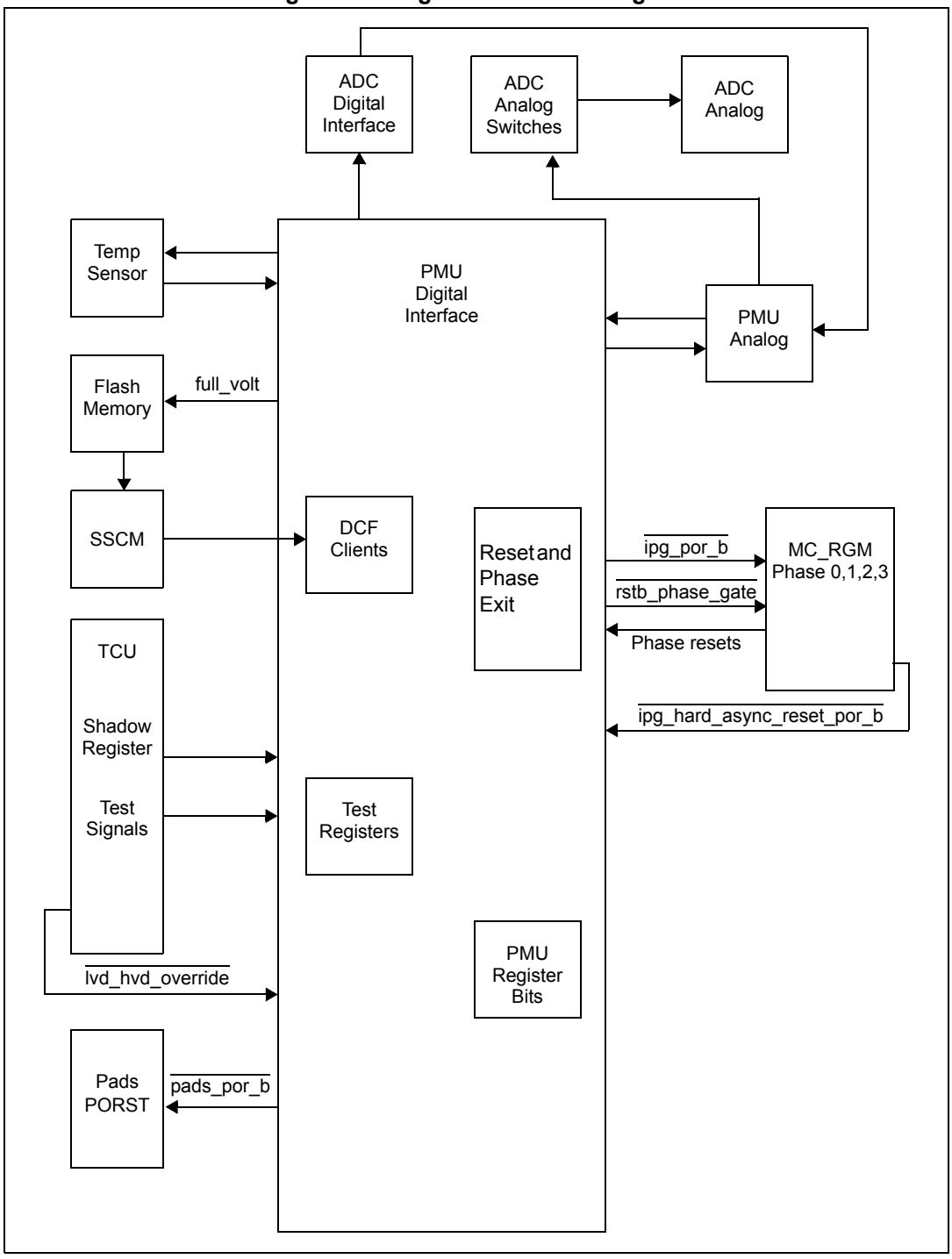
In addition, the PMU digital logic contains the enable signals, trim bits and other registers for the temperature sensor.

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 during initial POR.

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 592* the PMUDig top level block diagram interfaced to the device modules.

Figure 592. Digital PMU block diagram



46.2 Main features

- PMU management at a main reset
- Resets signal generation

46.2.1 Standard features

- IPS bus interface

46.3 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 PMU_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 PMU_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, MREG, CTL_TD.

46.4 Memory map/register definition

Table 615 shows the PMU memory map, which is part of the MC_PCU memory map, see Power control unit (MC_PCU) chapter . The PMU includes many registers for configuring and monitoring LVD monitors, some other control registers.

Table 615. Power Management Controller Memory Map

Address	Register	Location
0x0400	Event Pending Register LV0 (EPR_LV0)	*Section 46.5.1*
0x0404	Reset Event Enable LV0 (REE_LV0)	*Section 46.5.2*
0x0408	Reset Event Select LV0 (RES_LV0)	*Section 46.5.3*
0x040C	Interrupt Enable LV0 (IE_LV0)	*Section 46.5.4*
0x0410	FCCU Event Enable LV0 (FEE_LV0)	*Section 46.5.5*
0x0414–0x041C	***Reserved***	
0x0420	Event Pending Register LV1 (EPR_LV1)	*Section 46.5.6*
0x0424	Reset Event Enable LV1 (REE_LV1)	*Section 46.5.7*

Table 615. Power Management Controller Memory Map(Continued)

Address	Register	Location
0x0428	Reset Event Select LV1 (RES_LV1)	*Section 46.5.8*
0x042C	Interrupt Enable LV1 (IE_LV1)	*Section 46.5.9*
0x0430	FCCU Event Enable LV1 (FEE_LV1)	*Section 46.5.10*
0x0434–0x043C	***Reserved***	
0x0440	Event Pending Register HV0 (EPR_HV0)	*Section 46.5.11*
0x0444	Reset Event Enable HV0 (REE_HV0)	*Section 46.5.12*
0x0448	Reset Event Selection HV0 (RES_HV0)	*Section 46.5.13*
0x044C	Interrupt Enable HV0 (IE_HV0)	*Section 46.5.14*
0x0450	FCCU Event Enable HV0 (FEE_HV0)	*Section 46.5.15*
0x0454–0x045C	***Reserved***	
0x0460	Event Pending Register HV1 (EPR_HV1)	*Section 46.5.16*
0x0464	Reset Event Enable HV1 (REE_HV1)	*Section 46.5.17*
0x0468	Reset Event Selection HV1 (RES_HV1)	*Section 46.5.18*
0x046C	Interrupt Enable HV1 (IE_HV1)	*Section 46.5.19*
0x0470	FCCU Event Enable HV1 (FEE_HV1)	*Section 46.5.20*
0x0474–0x047C	***Reserved***	
0x0480	Supply Gauge Status Register (GR_S)	*Section 46.5.21*
0x0484	Pending Gauge Status Register (GR_P)	*Section 46.5.22*
0x0488	Global Interrupt Enable Register (IE_G)	*Section 46.5.23*
0x048C–0x063C	***Reserved***	
0x0640	Main Regulator Control Register (MREG_CTRL)	*Section 46.5.24*
0x0644–0x06FC	***Reserved***	
0x0700	Temperature Event Pending register (EPR_TD)	*Section 46.5.25*
0x0704	Temperature Reset Event Enable register (REE_TD)	*Section 46.5.26*
0x0708	Temperature Reset Event Selection register (RES_TD)	*Section 46.5.27*
0x070C	Temperature sensor configuration register (CTL_TD)	*Section 46.5.28*
0x0710–0x0714	***Reserved***	
0x0718	Temperature FCCU Event Enable register (FEE_TD)	*Section 46.5.29*
0x071C–0x07A4	***Reserved***	
0x07A8	ESR0 configuration register (ESR0_CFG)	*Section 46.5.30*
0x07AC–0x07CF	***Reserved***	
0x07D0	User BIST Flags register (BIST_FLAGS)	*Section 46.5.31*
0x07D4	User BIST Control register (BIST_CTRL)	*Section 46.5.32*
0x07D8	User BIST Time1 and Time0 register (BIST_TIME10)	*Section 46.5.33*
0x07DC	User BIST Time3 and Time2 register (BIST_TIME32)	*Section 46.5.34*

Table 615. Power Management Controller Memory Map(Continued)

Address	Register	Location
0x07E0	User BIST Time6 and Time5 register (BIST_TIME65)	*Section 46.5.35*
0x07E4	User BIST VD Under Test Monitor Register (BIST_DEBUG)	*Section 46.5.36*

Table 616 shows how the 32-bit registers for LV0, LV1, HV0, and HV1 are organized.

Table 616. 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_c_fl	VD6=hvd140_c	VD5=Reserved	VD4=Reserved
HV0	VD11=lvd290_ad_fl_c	VD10=mvd270_fl_c	VD9=mvd240t_c	VD8=por200_c
HV1	VD15 = uvd600_c	VD14=lvd400_ad	VD13=hvd400_c	VD12=Reserved

where each byte VD# is so structured:

Bit7, Bit6, ADC, FLASH, Bit3, Bit2, Bit1, CORE.

Table 617 shows the correspondence between VD number and VD name, where HV means a 3.3 V/5.0 V supply and LV means a 1.2 V supply.

Table 617. VD number versus VD name table

VD Number	VD Name	VD Meaning
VD0	POR041_C	LV detector asserts when PMU supply < 0.41 V
VD1	MVD085t_C	LV detector asserts when PMU supply < 0.85 V
VD2	MVD098_C	LV detector asserts when PMU supply < 0.98 V
VD2	MVD098_FL	LV detector asserts when FLASH supply < 0.98 V
VD3	LVD108_C	LV detector asserts when PMU supply < 1.08 V
VD6	HVD140_C	LV detector asserts when PMU supply > 1.40 V
VD7	UVD145_C	LV detector asserts when PMU supply > 1.45 V
VD7	UVD145_FL	LV detector asserts when FLASH supply > 1.45 V
VD8	POR200_C	HV detector asserts when PMU supply < 2.00 V
VD9	MVD240t_C	HV detector asserts when PMU supply < 2.40 V
VD10	MVD270_C	HV detector asserts when PMU supply < 2.70 V
VD10	MVD270_FL	HV detector asserts when FLASH supply < 2.70 V
VD11	LVD290_AD	HV detector asserts when ADC supply < 2.90 V
VD11	LVD290_C	HV detector asserts when PMU supply < 2.90 V
VD11	LVD290_FL	HV detector asserts when FLASH supply < 2.90 V

Table 617. VD number versus VD name table(Continued)

VD Number	VD Name	VD Meaning
VD13	HVD400_C	HV detector asserts when PMU supply > 4.00 V
VD14	LVD400_AD	HV detector asserts when ADC supply < 4.00 V
VD15	UVD600_C	HV detector asserts when PMU supply > 6.00 V

46.5 Registers description

46.5.1 Event Pending Register (EPR_LV0)

This Event Pending Register indicates the present and past state of the voltage detect signals in 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: 0x0400 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 593. Event Pending Register (EPR_LV0)

Table 618. 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 1.08 V 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 voltage1.08 V point supply.

46.5.2 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: 0x0404 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 594. Reset Event Enable Register (REE_LV0)

Table 619. REE_LV0 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 1.08 V 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.

46.5.3 Reset Event Select Register (RES_LV0)

This Reset Event Select Register controls, in LOW VOLTAGE 0 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: 0x0408 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 595. Reset Event Select Register (RES_LV0)

Table 620. 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.

46.5.4 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: 0x040C 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 596. Interrupt Enable Register (IE_LV0)

Table 621. 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 1.08 V supply event occurs. 0 No interrupt occurs. 1 An interrupt occurs.

46.5.5 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 Collection and 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: 0x0410 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 597. FCCU Event Enable Register (FEE_LV0)

Table 622. FEE_LV0 field descriptions

Field	Description
7 FEE3_C	FEE3_C FCCU event enable This bit defines whether a VD assertion of LVD108_C VD, the voltage level detect for the low voltage CORE 1.08 V supply, generates an FCCU event. 0 Disabled. A VD assertion of LV0 on the supply of the CORE does not cause an FCCU event. 1 Enabled. A VD assertion of LV0 on the supply of the CORE causes an FCCU event.

46.5.6 Event Pending Register (EPR_LV1)

This Event Pending Register indicates the present and past state of the voltage detect signals in LOW VOLTAGE 1 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: 0x0420 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 598. Event Pending Register (*EPR_LV1*)

Table 623. EPR_LV1 field descriptions

Field	Description
15 HVD6_C	HVD6_C flag This bit is the Low Voltage status flag associated with HVD140_C VD, the voltage level detect for the low voltage CORE 1.40 V supply. It is asserted when the supply falls below its corresponding Voltage Detector threshold, and clears when the supply rises above its corresponding HVD6 threshold and a '1' is written to this bit location. 0 Currently no occurrence. 1 VD occurrence detected on the low voltage 1.40 V point supply.

46.5.7 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: 0x0424 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 599. Reset Event Enable Register (REE_LV1)

Table 624. 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 1.40 V 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.

46.5.8 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: 0x0428 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 600. Reset Event Select Register (RES_LV1)

Table 625. RES_LV1 field descriptions

Field	Description
15 RES6_C	RES6_C event select. his bit defines whether a HVD140_C VD, the voltage level detect for the low voltage CORE 1.40 V 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.

46.5.9 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: 0x042C 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 601. Interrupt Enable Register (IE_LV1)

Table 626. 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 1.40 V supply the voltage event occurs. 0 No interrupt occurs. 1 An interrupt occurs.

46.5.10 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 Collection and 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: 0x0430 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 602. FCCU Event Enable Register (FEE_LV1)

Table 627. FEE_LV1 field descriptions

Field	Description
15 FEE6_C	FEE6_C FCCU event enable This bit defines whether a HVD140_C VD assertion of Core 1.40 supply generates an FCCU event. 0 Disabled. A VD assertion of LV1on the supply of the CORE does not cause an FCCU event. 1 Enabled. A VD assertion of LV0 on the supply of the CORE causes an FCCU event.

46.5.11 Event Pending Register (EPR_HV0)

This Event Pending Register indicates the present and past state of the voltage detect signals in 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: 0x0440 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	LVD11_AD	LVD11_FL	0	0	0	LVD11_C	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	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 603. Event Pending Register (EPR_HV0)

Table 628. EPR_HV0 field descriptions

Field	Description
2 LVD11_AD	LVD11_AD flag This bit is the high-voltage status flag associated with LVD290_AD, the voltage level detect for the low voltage ADC 2.90 V 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 2.90 V 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 2.90 V 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 2.90 V 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 2.90 V 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 2.90 V point supply.

46.5.12 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: 0x0444 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	REE11_AD	REE11_FL	0	0	0	REE11_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 604. Reset Event Enable Register (REE_HV0)

Table 629. REE_HV0 field descriptions

Field	Description
2 REE11_AD	LVD11_AD reset enable This bit defines whether LVD290_AD VD, the voltage level detect for the high voltage assertion on the ADC 2.9 V supply generates a system reset. The REE11_AD 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.
3 REE11_FL	LVD11_FL reset enable This bit defines whether LVD290_FL VD, the voltage level detect for the high voltage assertion on the FLASH 2.90 V supply of the cold point generates a system reset. The REE11_FL 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.
7 REE11_C	LVD11_C reset enable This bit defines whether an LVD assertion LVD290_C VD, the voltage level detect for the high voltage CORE 2.90 V supply, of the cold point generates a system reset. The REE11_C 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.

46.5.13 Reset Event Select Register (RES_HV0)

This Reset Event Select Register controls, in HIGH VOLTAGE 0 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: 0x0448 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	RES11_AD	RES11_FL	0	0	0	RES11_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 605. Reset Event Select Register (RES_HV0)

Table 630. RES_HV0 field descriptions

Field	Description
2 RES11_AD	RES11_AD reset event select This bit defines whether an LVD assertion on the ADC 2.9 Vsupply of the cold point generates a destructive reset or a functional reset. The RES11_AD 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.
3 RES11_FL	RES11_FL reset event select This bit defines whether an LVD assertion on the FLASH 2.9 V supply of the cold point generates a destructive reset or a functional reset. The RES11_FL 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.
7 RES11_C	RES11_C reset event select This bit defines whether an LVD assertion on the CORE 2.9 V supply of the cold point generates a destructive reset or a functional reset. The RES11_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.

46.5.14 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: 0x044C Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	LVD11IE_AD	LVD11IE_FL	0	0	0	LVD11IE_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 606. Interrupt Enable Register (IE_HV0)

Table 631. IE_HV0 field descriptions

Field	Description
2 LVD11IE_AD	LVD11IE_AD Interrupt Enable This bit determines whether an interrupt is seen by the system when LVD290_AD VD, the voltage level detect for the high voltage assertion on the ADC 2.9 Vsupply event occurs. 0 No interrupt occurs. 1 An interrupt occurs.
3 LVD11IE_FL	LVD11IE_FL Interrupt Enable This bit determines whether an interrupt is seen by the system when the LVD290_FL VD, the voltage level detect for the high voltage assertion on the FLASH 2.90 V supply event occurs. 0 No interrupt occurs. 1 An interrupt occurs.
7 LVD11IE_C	LVD11IE_C Interrupt Enable This bit determines whether an interrupt is seen by the system when LVD290_C VD, the voltage level detect for the high voltage CORE 2.90 V supply event occurs. 0 No interrupt occurs. 1 An interrupt occurs.

46.5.15 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 Collection and 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: 0x0450 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	FEE11_AD	FEE11_FL	0	0	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 607. FCCU Event Enable Register (FEE_HV0)

Table 632. FEE_HV0 field descriptions

Field	Description
2 FEE11_AD	FEE11_AD FCCU event enable This bit defines whether a LVD290_AD VD, the voltage level detect for the high voltage assertion on the ADC 2.90 V supply assertion generates an FCCU event. 0 Disabled. A VD assertion of HV0 on the supply of the ADC does not cause an FCCU event. 1 Enabled. A VD assertion of HV0 on the supply of the ADC 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 2.90 V supply assertion generates an FCCU event. 0 Disabled. A VD assertion of HV0 on the supply of the FLASH does not cause an FCCU event. 1 Enabled. A VD assertion of HV0 on the supply of the FLASH 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 2.90 V supply assertion generates an FCCU event. 0 Disabled. A VD assertion of HV0 on the supply of the CORE does not cause an FCCU event. 1 Enabled. A VD assertion of HV0 on the supply of the CORE causes an FCCU event.

46.5.16 Event Pending Register (EPR_HV1)

This Event Pending Register indicates the present and past state of the voltage detect signals in HIGH VOLTAGE 1 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.

Note: There is a possibility that we will have a incorrect value latched inside the EPR_HV1 register when PMU BIST is being run. Please clear the EPR_HV1 register after completion of BIST.

Offset: 0x0460 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	LVD14_AD	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	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 608. Event Pending Register (EPR_HV1)

Table 633. EPR_HV1 field descriptions

Field	Description
10 LVD14_AD	LVD14_AD flag This bit is the High voltage status flag associated with LVD400_AD VD, the VD for the high voltage ADC 4.00 V 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_AD threshold and a one is written to this bit location. 0 Currently no occurrence. 1 VD occurrence detected on the high voltage 4.00 V 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 4.00 V 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 4.00 V point supply.

46.5.17 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: 0x0464 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	REE14_AD	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	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 609. Reset Event Enable Register (REE_HV1)

Table 634. REE_HV1 field descriptions

Field	Description
10 REE14_AD	REE14_AD reset enable This bit defines whether a LVD400_AD VD, the VD for the high voltage ADC 4.00 V supply generates a system reset. The RES14_AD 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 4.00 V supply generates a system reset. The RES13_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.

46.5.18 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: 0x0468 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	RES14_AD	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	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 610. Reset Event Select Register (RES_HV1)

Table 635. RES_HV1 field descriptions

Field	Description
10 RES14_AD	RES14_AD reset event select This bit defines whether a LVD400_AD VD, the VD for the high voltage ADC 4.00 V supply generates a destructive reset or a functional reset by LVD14_AD 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 4.00 V 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.

46.5.19 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: 0x046C 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	LVDIE14_AD	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	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 611. Interrupt Enable Register (IE_HV1)

Table 636. IE_HV1 field descriptions

Field	Description
10 LVDIE14_AD	LVDIE14_AD Interrupt Enable This bit determines whether an interrupt is seen by the system when LVD400_AD VD, the VD for the high voltage ADC 4.00 V 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 4.00 V supply event occurs. 0 No interrupt occurs. 1 An interrupt occurs.

46.5.20 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 Collection and 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: 0x0470 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	FEE14_AD	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	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 612. FCCU Event Enable Register (FEE_HV1)

Table 637. FEE_HV1 field descriptions

Field	Description
10 FEE14_AD	FEE14_AD FCCU event enable This bit defines whether a VD assertion of LV14 on lvd400_ad for the supply of the ADC generates an FCCU event. 0 Disabled. A VD assertion of LV14on the supply of the CORE does not cause an FCCU event. 1 Enabled. A 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 a VD assertion of HV14 on hvd400_c for the supply of the CORE generates an FCCU event. 0 Disabled. A VD assertion of HV14on the supply of the CORE does not cause an FCCU event. 1 Enabled. A VD assertion of HV14 on the supply of the CORE causes an FCCU event.

46.5.21 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 638*.

Offset: 0x0480 Access: User read-only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	VD15_C	VD14_AD	0	0	VD13_C	0	VD11_AD	VD11_FL	VD11_C	VD10_FL	VD10_C	0	VD9_C	0	VD8_C
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	VD7_FL	VD7_C	0	VD6_C	0	0	0	0	0	VD3_C	VD2_FL	VD2_C	0	VD1_C	0	VD0_C
W																
Reset(1)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 613. Supply Gauge Status Register (*GR_S*)

1. When the device is operating at 3.3 V, lvd400_ad will not lift so the reset value of GR_S register is 0x20000000. When the device is operating at 5 V, hvd400_c will not lift so the reset value of GR_S register is 0x04000000.

Table 638. *GR_S* field descriptions

Field	Description
1 VD15_C	VD15_C high-voltage detect flag This read-only bit is the high-voltage status flag associated with the voltage level detect uvd600_c for the high voltage 6 V 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. All the 6.00 V HVDs are combined to form this bit. 0 Currently no occurrence. 1 HVD occurrence detected on the high voltage 6.00 V supply.
2 VD14_AD	VD14_AD low-voltage detect flag This read-only bit is the low-voltage status flag associated with the voltage level detect lvd400_ad for the high voltage 4 V ADC supply. It is asserted when any of the ADC supplies rises above the corresponding LVD threshold, and clears when all the ADC supplies fall below the corresponding LVD threshold. All the 4.00 V LVDs are combined to form this bit. 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 supply rises above its corresponding HVD threshold, and clears when the supply rises above its corresponding HVD threshold. 0 Currently no occurrence. 1 HVD occurrence detected on the high voltage 4.00 V supply.
7 VD11_AD	VD11_AD low-voltage detect flag This read-only bit is the low-voltage status flag associated with the voltage level detect lvd290_ad for the high voltage 2.9 V ADC supplies. It is asserted when any of the ADC supplies falls below the corresponding LVD threshold, and clears when all the ADC supplies rise above the corresponding LVD threshold. All of the 2.9 V LVDs are combined to form this bit. 0 Currently no occurrence. 1 LVD occurrence detected on the high voltage 2.9 V supplies.

Table 638. *GR_S* field descriptions(Continued)

Field	Description
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. All of the 2.9 V LVDs are combined to form this bit. 0 Currently no occurrence. 1 LVD occurrence detected on the high voltage 2.9 V supplies.
9 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. All of the 2.9 V LVDs are combined to form this bit. 0 Currently no occurrence. 1 LVD occurrence detected on the high voltage 2.9 V supplies.
10 VD10_FL	VD10_FL low-voltage detect flag This read-only bit is the low-voltage status flag associated with the voltage level detect mvd270_fl for the high voltage 2.70 V FLASH supply. It is asserted when any of the FLASH supplies rises above the corresponding LVD threshold, and clears when all the FLASH supplies fall below the corresponding LVD threshold. All the 2.70 V LVDs are combined to form this bit. 0 Currently no occurrence. 1 LVD occurrence detected on the high voltage 2.70 V supplies.
11 VD10_C	VD10_C low-voltage detect flag This read-only bit is the low-voltage status flag associated with the voltage level detect mvd270_c for the high voltage 2.70 V CORE supply. It is asserted when any of the CORE supplies rises above the corresponding LVD threshold, and clears when all the CORE supplies fall below the corresponding LVD threshold. All the 2.70 V LVDs are combined to form this bit. 0 Currently no occurrence. 1 LVD occurrence detected on the high voltage 2.70 V supplies.
13 VD9_C	VD9_C low-voltage detect flag This read-only bit is the low-voltage status flag associated with the voltage level detect mvd240_c for the high voltage 2.40 V CORE supply. It is asserted when any of the CORE supplies rises above the corresponding LVD threshold, and clears when all the CORE supplies fall below the corresponding LVD threshold. All the 2.40 V LVDs are combined to form this bit. 0 Currently no occurrence. 1 LVD occurrence detected on the high voltage 2.40 V supplies.
15 VD8_C	VD8_C low-voltage detect flag This read-only bit is the low-voltage status flag associated with the voltage level detect por200_c for the high voltage 2.00 V supply. It is asserted when any of the supplies rises above the corresponding LVD threshold, and clears when all the supplies fall below the corresponding LVD threshold. All the 2.00 V LVDs are combined to form this bit. 0 Currently no occurrence. 1 LVD occurrence detected on the high voltage 2.00 V supplies.

Table 638. *GR_S* field descriptions(Continued)

Field	Description
16 VD7_FL	VD7_FL high-voltage detect flag This read-only bit is the high-voltage status flag associated with the voltage level detect uvd145_c for the high voltage 1.45 V FLASH supplies. It is asserted when any of the FLASH supplies rises above the corresponding HVD threshold, and clears when all the FLASH supplies fall below the corresponding HVD threshold. All the 1.45 V HVDs are combined to form this bit. 0 Currently no occurrence. 1 HVD occurrence detected on the high voltage 1.45 V supply.
17 VD7_C	VD7_C high-voltage detect flag This read-only bit is the high-voltage status flag associated with the voltage level detect uvd145_c for the high voltage 1.45 V 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. All the 1.45 V HVDs are combined to form this bit. 0 Currently no occurrence. 1 HVD occurrence detected on the high voltage 1.45 V supply.
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 hvd145_c for the high voltage 1.40 V 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. All the 1.40 V HVDs are combined to form this bit. 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. All of the 1.08 V LVDs are combined to form this bit. 0 Currently no occurrence. 1 LVD occurrence detected on the high voltage 1.08 V supplies.
26 VD2_FL	VD2_FL low-voltage detect flag This read-only bit is the low-voltage status flag associated with the voltage level detect mvd098_fl for the high voltage 0.98 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. All of the 0.98 V LVDs are combined to form this bit. 0 Currently no occurrence. 1 LVD occurrence detected on the high voltage 0.98 V supplies.
27 VD2_C	VD2_C low-voltage detect flag This read-only bit is the low-voltage status flag associated with the voltage level detect mvd098_c for the high voltage 0.98 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. All of the 0.98 V LVDs are combined to form this bit. 0 Currently no occurrence. 1 LVD occurrence detected on the high voltage 0.98 V supplies.

Table 638. *GR_S* field descriptions(Continued)

Field	Description
29 VD1_C	VD1_C low-voltage detect flag This read-only bit is the low-voltage status flag associated with the voltage level detect mvd085t_c for the high voltage 0.85 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. All of the 0.85 V LVDs are combined to form this bit. 0 Currently no occurrence. 1 LVD occurrence detected on the high voltage 0.85 V supplies.
31 VD0_C	VD0_C low-voltage detect flag This read-only bit is the low-voltage status flag associated with the voltage level detect por041_c for the high voltage 0.41 V supply. It is asserted when any of the supplies rises above the corresponding LVD threshold, and clears when all the supplies fall below the corresponding LVD threshold. All the 0.41 V LVDs are combined to form this bit. 0 Currently no occurrence. 1 LVD occurrence detected on the high voltage 0.41 V supplies.

46.5.22 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: 0x0484 Access: User read-only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	VD15_C	VD14_AD	0	0	VD13_C	0	VD11_AD	VD11_FL	VD11_C	VD10_FL	VD10_C	0	VD9_C	0	VD8_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	VD7_FL	VD7_C	0	VD6_C	0	0	0	0	0	VD3_C	VD2_FL	VD2_C	0	VD1_C	IRQ_ST	VD0_C
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 614. Pending Gauge Status Register (GR_P)

Table 639. GR_P field descriptions

Field	Description
1 VD15_C	VD15_C Pending Monitor. Read-only register.
2 VD14_AD	VD14_AD Pending Monitor. Read-only register.
5 VD13_C	VD13_C Pending Monitor. Read-only register.
7 VD11_AD	VD11_AD Pending Monitor. Read-only register.
8 VD11_FL	VD11_FL Pending Monitor. Read-only register.
9 VD11_C	VD11_C Pending Monitor. Read-only register.
10 VD10_FL	VD10_FL Pending Monitor. Read-only register.
11 VD10_C	VD10_C Pending Monitor. Read-only register.
13 VD9_C	VD9_C Pending Monitor. Read-only register.
15 VD8_C	VD8_C Pending Monitor. Read-only register
16 VD7_FL	VD7_FL Pending Monitor. Read-only register.
17 VD7_C	VD7_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.
26 VD2_FL	VD2_FL Pending Monitor. Read-only register.
27 VD2_C	VD2_C Pending Monitor. Read-only register.
29 VD1_C	VD1_C Pending Monitor. Read-only register.
30 IRQ_ST	IRQ_ST user BIST Pending Monitor. Read-only register.
31 VD0_C	VD0_C Pending Monitor. Read-only register

46.5.23 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: 0x0488 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_AD	0	0	VDIE13_C	0	VDIE11_AD	VDIE11_FL	VDIE11_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

	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 615. Interrupt Enable Global Register (IE_G)

Table 640. 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_AD	VDIE14_AD enable interrupt monitor. Read-only register.
5 VDIE13_C	VDIE13_C enable interrupt monitor. Read-only register.
7 VDIE11_AD	VDIE11_AD enable interrupt monitor. Read-only register.
8 VDIE11_FL	VDIE11_FL enable interrupt monitor. Read-only register.
9 VDIE11_C	VDIE11_C enable interrupt monitor. Read-only register.
19 VDIE6_C	VDIE6_C enable interrupt monitor. Read-only register.

Table 640. IE_G field descriptions(Continued)

Field	Description
25 VDIE3_C	VDIE3_C enable interrupt monitor. Read-only register.
30 IRQ_EN	IRQ_EN user BIST enable interrupt monitor. Read-only register.

46.5.24 Main Regulator Control Register (MREG_CTRL)

This control register controls and indicates the state of the main regulator. If the MREG_ENB signal is low, the main regulator is enabled. If the MREG_ENB signal is high, the main regulator is disabled.

Offset: 0x0640 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	MREG_ENB
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 616. Main Regulator Control Register (MREG_CTRL)

Table 641. MREG_CTRL field descriptions

Field	Description
31 MREG_ENB	MREG_ENB flag This bit enables and disables the main regulator. 0 Main regulator is enabled. 1 Main regulator is disabled.

46.5.25 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: 0x0700 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	TEMP_2	TEMP_1	TEMP_0
W														w1c	w1c	w1c
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	TEMP_1	TEMP_0
W															w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 617. Event Pending Register (EPR_TD)

Table 642. EPR_TD field descriptions

Field	Description
29 TEMP_2	TEMP_2 flag This bit is the temperature status flag associated with the temperature for the temperature sensor point (165°C). It is asserted when the temperature exceeds its corresponding threshold, and cleared by the user by writing one in this bit location when the temperature falls below its corresponding threshold. If the IETD_2 bit is also asserted, an interrupt is sent to the CPU. If REE_TD[TEMP_2] is also asserted, a system reset is generated, which clears IETD_2 and negate the interrupt request. If REE_TD[TEMP_2] is asserted and a destructive reset is selected (via RES_TD[TEMP_2] being 0), then a destructive reset is generated. If RES_TD[TEMP_2] is set, then a functional reset is generated. 0 Currently no occurrence 1 Temperature occurrence detected

Table 642. EPR_TD field descriptions(Continued)

Field	Description
30 TEMP_1	TEMP_1 flag This bit is the temperature status flag associated with the temperature for the temperature sensor point (150°C). It is asserted when the temperature exceeds its corresponding threshold, and cleared by the user by writing one 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
31 TEMP_0	TEMP_0 flag This bit is the temperature status flag associated with the temperature for the cold temperature sensor point (-40°C). It is asserted when the temperature exceeds its corresponding threshold, and cleared by the user by writing one in this bit location when the temperature falls below its corresponding threshold. If the IETD_0 bit is also asserted, an interrupt is sent to the CPU. If REE_TD[TEMP_0] is also asserted, a system reset is generated, which clears IETD_0 and negate the interrupt request. If REE_TD[TEMP_0] is asserted and a destructive reset is selected (via RES_TD[TEMP_0] being 0), then a destructive reset is generated. If RES_TD[TEMP_0] is set, then a functional reset is generated. 0 Currently no occurrence 1 Temperature occurrence detected

46.5.26 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: 0x0704 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	TEMP_2	TEMP_1	TEMP_0
W																
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	TEMP_1	TEMP_0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 618. Reset Event Enable Register (REE_TD)

Table 643. REE_TD field descriptions

Field	Description
29 TEMP_2	TEMP_2 reset enable This bit defines whether a temperature detect assertion generates a system reset. The RES_TD[TEMP_2] 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 TEMP_1	TEMP_1 reset enable This bit defines whether a temperature detect assertion generates a system reset. The RES_TD[TEMP_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 TEMP_0	TEMP_0 reset enable This bit defines whether a temperature detect assertion generates a system reset. The RES_TD[TEMP_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

46.5.27 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: 0x0708 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	TEMP_2	TEMP_1	TEMP_0
W																
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	TEMP_1	TEMP_0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 619. Reset Event Select Register (RES_TD)

Table 644. RES_TD field descriptions

Field	Description
29 TEMP_2	TEMP_2 reset event select This bit defines whether a temperature detect assertion generates a destructive reset or a functional reset. The REE_TD[TEMP_2] bit determines whether the reset event is enabled. 0 Destructive Reset Generated. Temperature Sensor assertion causes a destructive system reset 1 Functional Reset Generated. Temperature Sensor assertion causes a functional system reset
30 TEMP_1	TEMP_1 reset event select This bit defines whether a temperature detect assertion generates a destructive reset or a functional reset. The REE_TD[TEMP_1] bit determines whether the reset event is enabled. 0 Destructive Reset Generated. Temperature Sensor assertion causes a destructive system reset 1 Functional Reset Generated. Temperature Sensor assertion causes a functional system reset
31 TEMP_0	TEMP_0 reset event select This bit defines whether an temperature detect assertion generates a destructive reset or a functional reset. The REE_TD[TEMP_0] bit determines whether the reset event is enabled. 0 Destructive Reset Generated. Temperature Sensor assertion causes a destructive system reset 1 Functional Reset Generated. Temperature Sensor assertion causes a functional system reset

46.5.28 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 5-

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: 0x070C 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	TS2IE	TS1IE	TS0IE	0(1)	0(1)	0(1)	0(1)	0(1)	0(1)	0(1)	0(1)	0(1)	0(1)	0(1)	MUX_CTRL	AOUT_EN
W																
R	0	TS1IE	TS0IE	0(1)	0(1)	0(1)	0(1)	0(1)	0(1)	0(1)	0(1)	0(1)	0(1)	0(1)	MUX_CTRL	AOUT_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 620. Temperature Sensor Configuration Register (CTL_TD)

Table 645. CTL_TD field descriptions

Field	Description
16 TS2IE	TS2IE Temperature Sensor input 2 Interrupt Enable This bit determines whether an interrupt is seen by the system when the temperature detect event occurs. The MSB in IE_G must be set to access this bit. 0 No interrupt occurs 1 An interrupt occurs
17 TS1IE	TS1IE Temperature Sensor input 1 Interrupt Enable This bit determines whether an interrupt is seen by the system when the temperature detect event occurs. The MSB in IE_G must be set to access this bit. 0 No interrupt occurs 1 An interrupt occurs
18 TS0IE	TS0IE Temperature Sensor input 0 Interrupt Enable This bit determines whether an interrupt is seen by the system when the temperature detect event occurs. The MSB in IE_G must be set to access this bit. 0 No interrupt occurs 1 An interrupt occurs
30 MUX_CTRL	MUX_CTRL Multiplex Control for CTAT analog mux of temperature sensor. See the Temperature Sensor chapter for details.
31 AOUT_EN	AOUT_EN Analog Output Enable See the Temperature Sensor chapter for details.

46.5.29 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 Collection and 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: 0x0718 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	FEE_TS2	FEE_TS1	FEE_TS0
W																
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	FEE_TS1	FEE_TS0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1

Figure 621. Temperature Sensor FCCU Event Enable Register (FEE_TD)

Table 646. Temperature Sensor FEE_TD field descriptions

Field	Description
29 FEE_TS2	FEE_TS2 Temp Sensor 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 FEE_TS1	FEE_TS1 Temp Sensor 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 FEE_TS0	FEE_TS0 Temp Sensor 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

46.5.30 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: 0x07A8 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 622. ESR0 ConFiGuration Register (ESR0_CFG)

Table 647. 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 0x1F8 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

46.5.31 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: 0x07D0 Access: User read-only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	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	0	0	FLAGS_P1													
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 623. User BIST Flags Register (BIST_FLAGS)

Table 648. BIST_FLAGS field description

Field	Description
2:15 FLAGS_P2	The 14 bits of field FLAGS_P2 are the Phase 2 flags of the User BIST that indicates the status of monitor when it is lifted: – Bit 2 - FLAGS_P2[13]: UVD600_C – Bit 3 - FLAGS_P2[12]: UVD145_C – Bit 4 - FLAGS_P2[11]: UVD145_FL – Bit 5 - FLAGS_P2[10]: HVD400_C – Bit 6 - FLAGS_P2[9]: HVD140_C – Bit 7 - FLAGS_P2[8]: LVD400_AD – Bit 8 - FLAGS_P2[7]: LVD290_C – Bit 9 - FLAGS_P2[6]: LVD290_AD – Bit 10 - FLAGS_P2[5]: LVD290_FL – Bit 11 - FLAGS_P2[4]: LVD108_C – Bit 12 - FLAGS_P2[3]: MVD270_C – Bit 13 - FLAGS_P2[2]: MVD270_FL – Bit 14 - FLAGS_P2[1]: MVD098_C – Bit 15 - FLAGS_P2[0]: MVD098_FL They are read-only bits.
18:31 FLAGS_P1	The 14 bits of field FLAGS_P1 are the Phase 1 flags of the User BIST that indicates the status of monitor when it is tripped: – Bit 18 - FLAGS_P2[13]: UVD600_C – Bit 19 - FLAGS_P2[12]: UVD145_C – Bit 20 - FLAGS_P2[11]: UVD145_FL – Bit 21 - FLAGS_P2[10]: HVD400_C – Bit 22 - FLAGS_P2[9]: HVD140_C – Bit 23 - FLAGS_P2[8]: LVD400_AD – Bit 24 - FLAGS_P2[7]: LVD290_C – Bit 25 - FLAGS_P2[6]: LVD290_AD – Bit 26 - FLAGS_P2[5]: LVD290_FL – Bit 27 - FLAGS_P2[4]: LVD108_C – Bit 28 - FLAGS_P2[3]: MVD270_C – Bit 29 - FLAGS_P2[2]: MVD270_FL – Bit 30 - FLAGS_P2[1]: MVD098_C – Bit 31 - FLAGS_P2[0]: MVD098_FL They are read-only bits.

46.5.32 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: 0x07D4 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 624. User BIST Control Register (BIST_CTRL)

Table 649. BIST_CTRL field descriptions

Field	Description
11 IRQST	IRQST is the interrupt request pending status bit to generate a User BIST Interrupt. It is set by HW and cleared by SW. The destructive reset resets this bit. 0 No interrupt occurred 1 An interrupt occurred
15 IRQEN	IRQEN is the interrupt request enable bit to permit a User BIST Interrupt. It is a R/W register. The destructive reset resets this bit. 0 No interrupt occurred 1 An interrupt occurred
19 NCFST	NCFST is the Not Critical Fault bit. The assertion of this bit will generate FCCU event once user BIST fails. It is set by HW and cleared by SW. The destructive reset resets this bit. 0 No User BIST NCF occurred 1 A User BIST NCF occurred
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 reset 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 PBRIDGE0_CLK clock cycle only, then the START bit register is clear by HW. The destructive reset resets this bit. 0 No start pulse 1 A start pulse occurs of a 1 PBRIDGE0_CLK clock cycle duration

46.5.33 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 PBRIDGE0_CLK cycles number required by the User BIST timer1 and timer0.

Note: For timing values to configure, see Section 10.4.7: BIST details.

Offset: 0x07D8 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	0	1	1	0	0	0	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	TIME_0												
W																
Reset	0	0	0	0	0	0	0	1	1	0	0	0	1	1	1	1

Figure 625. User BIST Time1 and Time0 Register (BIST_TIME10)

Table 650. BIST_TIME10 field descriptions

Field	Description
3:15 TIME_1	TIME_1 is the PBRIDGE0_CLK cycles number required by the TIMER 1 in the User BIST management. The destructive ipg_hard_async_reset_dest_b resets these bits.
19:31 TIME_0	TIME_0 is the PBRIDGE0_CLK cycles number required by the TIMER 0 in the User BIST management. The destructive ipg_hard_async_reset_dest_b resets these bits.

46.5.34 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 PBRIDGE0_CLK cycles number required by the User BIST timer3 and timer2.

Note: For timing values to configure, see Section 10.4.7: BIST details.

Offset: 0x07DC 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	0	0	1	1	0	0	0	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	TIME_2												
W																
Reset	0	0	0	0	0	0	0	0	1	1	0	0	0	1	1	1

Figure 626. User BIST Time3 and Time2 Register (BIST_TIME32)

Table 651. BIST_TIME32 field descriptions

Field	Description
3:15 TIME_3	TIME_3 is the PBRIDGE0_CLK cycles number required by the TIMER 3 in the User BIST management. The destructive ipg_hard_async_reset_dest_b resets these bits.
19:31 TIME_2	TIME_2 is the PBRIDGE0_CLK cycles number required by the TIMER 2 in the User BIST management. The destructive ipg_hard_async_reset_dest_b resets these bits.

46.5.35 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 PBRIDGE0_CLK cycles number required by the User BIST timer6 and timer5.

Note: For timing values to configure, see Section 10.4.7: BIST details.

Offset: 0x07E0 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	0	0	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	0	0	0

Figure 627. User BIST Time6 and Time5 Register (BIST_TIME65)

Table 652. BIST_TIME65 field descriptions

Field	Description
3:15 TIME_6	TIME_6 is the PBRIDGE0_CLK cycles number required by the TIMER 6 in the User BIST management. The destructive ipg_hard_async_reset_dest_b resets these bits.
19-31 TIME_5	TIME_5 is the PBRIDGE0_CLK cycles number required by the TIMER 5 in the User BIST management. The destructive ipg_hard_async_reset_dest_b resets these bits.

46.5.36 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: 0x07E4 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 628. User BIST VD Under Test Monitor Register (BIST_DEBUG)

Table 653. 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: 001100 MVD098_FL 001011 MVD098_C 000101 MVD270_FL 000100 MVD270_C 001010 LVD108_C 000011 LVD290_FL 000010 LVD290_AD 000001 LVD290_C 010101 LVD400_AD 001001 HVD140_C 011101 HVD400_C 001000 UVD145_FL 000111 UVD145_C 011100 UVD600_C 000000 No VD under Test

46.6 Temperature Sensor Interface Logic

The Temperature Sensor interface consists of an analog enable, a set of mode registers, and an overtemp threshold select control.

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 *Chapter 22: Temperature sensor (TSENS)*.

46.7 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.

46.8 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, Temperature Sensor, and ADC Bandgap).

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 46.5.26: Reset Event Enable Register (REE_TD)* 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 *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 *Chapter 31: Embedded Flash Memory*.

47 Power Control Unit (MC_PCU)

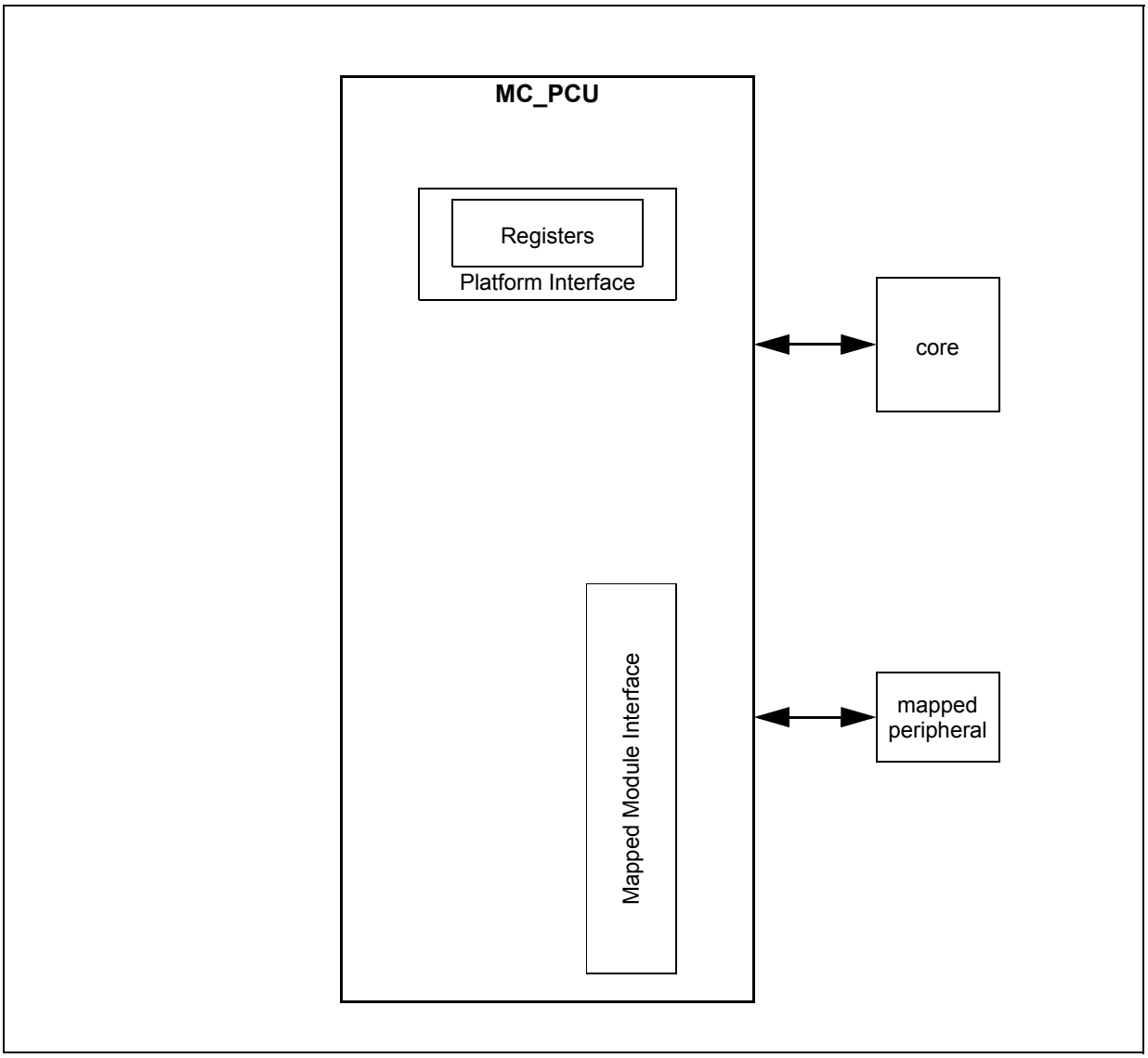
47.1 Introduction

47.1.1 Overview

The power control unit (MC_PCU) acts as a bridge for mapping the PMC peripheral to the MC_PCU address space.

Figure 629 shows the MC_PCU block diagram.

Figure 629. MC_PCU block diagram



47.1.2 Features

The MC_PCU includes the following feature:

- Maps the PMC registers to the MC_PCU address space

47.2 External signal description

The MC_PCU has no connections to any external pins.

47.3 Memory map and register definition

47.3.1 Memory map

Table 654. 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	*47.3.2.1 on page 1123*

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 655. MC_PCU memory map

Address	Name
0x0004 – 0x003C	Reserved
0x0040	PCU_PSTAT
0x0044 – 0x03FC	Reserved
0x0400 – 0x07FC	PMC registers
0x0800 – 0x3FFC	Reserved

47.3.2 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.

47.3.2.1 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 630. Power Domain Status Register (PCU_PSTAT)

This register reflects the power status of all available power domains.

Table 656. 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

48 Mode Entry Module (MC_ME)

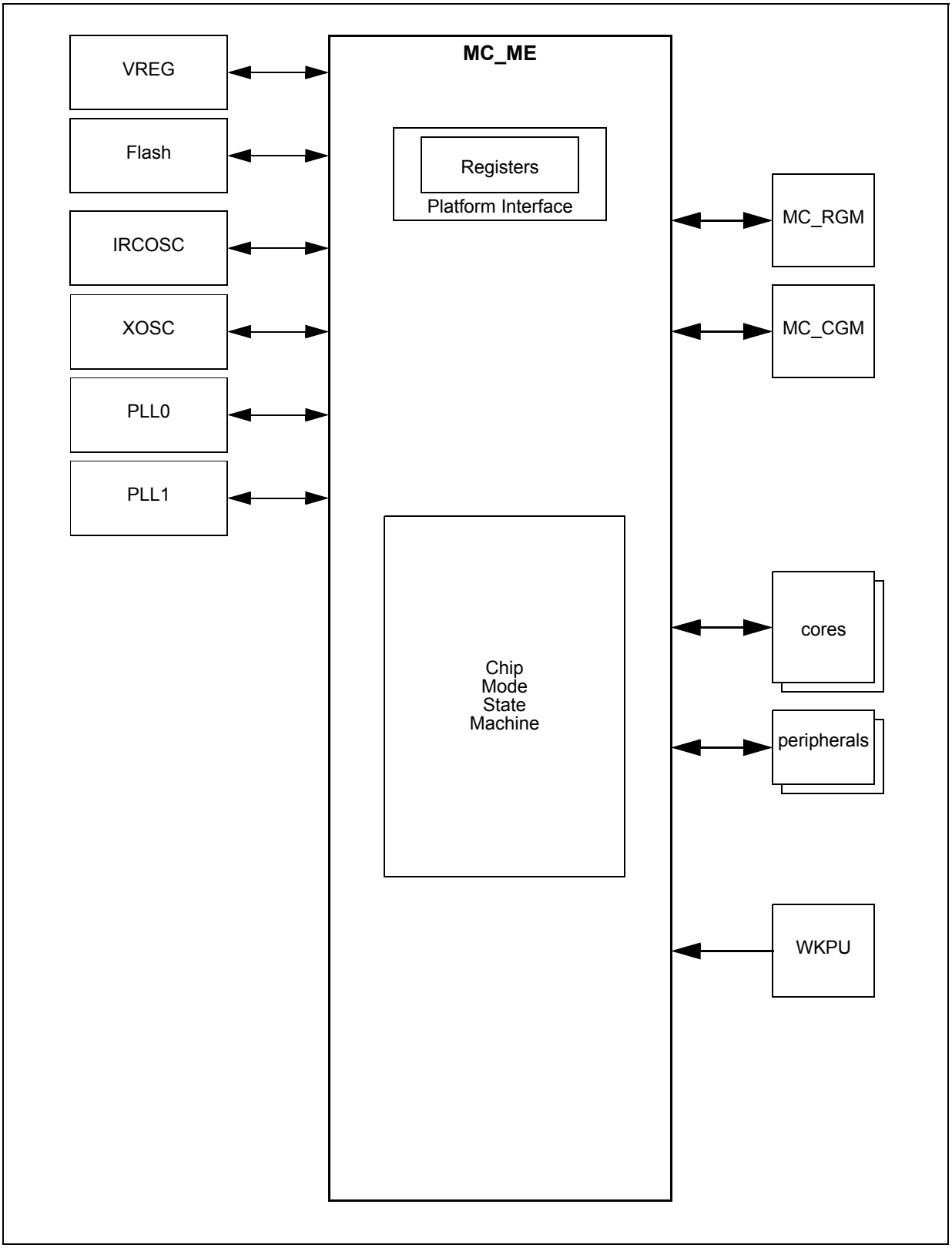
48.1 Introduction

48.1.1 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 631 shows the MC_ME block diagram.

Figure 631. MC_ME block diagram



48.1.2 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_PCTLn 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

48.1.3 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 657 describes the MC_ME modes.

Table 657. 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 657. 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 like clock sources, flash, main regulator, etc. 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

48.2 External signal description

The MC_ME has no connections to any external pins.

48.3 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

48.3.1 Memory map

Table 658. MC_ME register description

Address	Name	Description	Location
0x000	ME_GS	Global Status	*Section 48.3.2.1*
0x004	ME_MCTL	Mode Control	*Section 48.3.2.2*
0x008	ME_ME	Mode Enable	*Section 48.3.2.3*
0x00C	ME_IS	Interrupt Status	*Section 48.3.2.4*
0x010	ME_IM	Interrupt Mask	*Section 48.3.2.5*
0x014	ME_IMTS	Invalid Mode Transition Status	*Section 48.3.2.6*
0x018	ME_DMTS	Debug Mode Transition Status	*Section 48.3.2.7*
0x020	ME_RESET_MC	RESET Mode Configuration	*Section 48.3.2.8*
0x024	ME_TEST_MC	TEST Mode Configuration	*Section 48.3.2.9*
0x028	ME_SAFE_MC	SAFE Mode Configuration	*Section 48.3.2.10*
0x02C	ME_DRUN_MC	DRUN Mode Configuration	*Section 48.3.2.11*
0x030	ME_RUN0_MC	RUN0 Mode Configuration	*Section 48.3.2.12*
0x034	ME_RUN1_MC	RUN1 Mode Configuration	*Section 48.3.2.12*
0x038	ME_RUN2_MC	RUN2 Mode Configuration	*Section 48.3.2.12*
0x03C	ME_RUN3_MC	RUN3 Mode Configuration	*Section 48.3.2.12*
0x040	ME_HALT0_MC	HALT0 Mode Configuration	*Section 48.3.2.13*
0x048	ME_STOP0_MC	STOP0 Mode Configuration	*Section 48.3.2.14*
0x060	ME_PS0	Peripheral Status 0	*Section 48.3.2.15*
0x064	ME_PS1	Peripheral Status 1	*Section 48.3.2.16*
0x068	ME_PS2	Peripheral Status 2	*Section 48.3.2.17*
0x06C	ME_PS3	Peripheral Status 3	*Section 48.3.2.18*
0x070	ME_PS4	Peripheral Status 4	*Section 48.3.2.19*
0x074-0x077	Reserved		
0x078	ME_PS6	Peripheral Status 6	*Section 48.3.2.20*
0x07C	ME_PS7	Peripheral Status 7	*Section 48.3.2.21*
0x080	ME_RUN_PC0	Run Peripheral Configuration 0	*Section 48.3.2.22*
0x084	ME_RUN_PC1	Run Peripheral Configuration 1	*Section 48.3.2.22*
0x088	ME_RUN_PC2	Run Peripheral Configuration 2	*Section 48.3.2.22*
0x08C	ME_RUN_PC3	Run Peripheral Configuration 3	*Section 48.3.2.22*
0x090	ME_RUN_PC4	Run Peripheral Configuration 4	*Section 48.3.2.22*
0x094	ME_RUN_PC5	Run Peripheral Configuration 5	*Section 48.3.2.22*
0x098	ME_RUN_PC6	Run Peripheral Configuration 6	*Section 48.3.2.22*
0x09C	ME_RUN_PC7	Run Peripheral Configuration 7	*Section 48.3.2.22*

Table 658. MC_ME register description(Continued)

Address	Name	Description	Location
0x0A0	ME_LP_PC0	Low-Power Peripheral Configuration 0	*Section 48.3.2.23*
0x0A4	ME_LP_PC1	Low-Power Peripheral Configuration 1	*Section 48.3.2.23*
0x0A8	ME_LP_PC2	Low-Power Peripheral Configuration 2	*Section 48.3.2.23*
0x0AC	ME_LP_PC3	Low-Power Peripheral Configuration 3	*Section 48.3.2.23*
0x0B0	ME_LP_PC4	Low-Power Peripheral Configuration 4	*Section 48.3.2.23*
0x0B4	ME_LP_PC5	Low-Power Peripheral Configuration 5	*Section 48.3.2.23*
0x0B8	ME_LP_PC6	Low-Power Peripheral Configuration 6	*Section 48.3.2.23*
0x0BC	ME_LP_PC7	Low-Power Peripheral Configuration 7	*Section 48.3.2.23*
0x0C0–0x0DD	Reserved		
0x0DE	ME_PCTL30	PIT_0 Control	*Section 48.3.2.24*
0x0DF–0x0E3	Reserved		
0x0E4	ME_PCTL36	DMAMUX_0 Control	*Section 48.3.2.24*
0x0E5	Reserved		
0x0E6	ME_PCTL38	CRC_0 Control	*Section 48.3.2.24*
0x0E7–0x10E	Reserved		
0x10F	ME_PCTL79	FlexCAN_0 Control	*Section 48.3.2.24*
0x110–0x11B	Reserved		
0x11C	ME_PCTL92	LINFlexD_0 Control	*Section 48.3.2.24*
0x11D–0x121	Reserved		
0x122	ME_PCTL98	DSPI_1 Control	*Section 48.3.2.24*
0x123	ME_PCTL99	DSPI_0 Control	*Section 48.3.2.24*
0x124–0x12F	Reserved		
0x130	ME_PCTL112	ADCSAR_B Control	*Section 48.3.2.24*
0x131	ME_PCTL113	ADC sequencer SARADC_B Control	*Section 48.3.2.24*
0x132–0x13D	Reserved		
0x13E	ME_PCTL126	ADC sequencer SARADC_0 Control	*Section 48.3.2.24*
0x13F	ME_PCTL127	ADCSAR_0 Control	*Section 48.3.2.24*
0x140	ME_PCTL128	eTimer_1 Control	*Section 48.3.2.24*
0x141	ME_PCTL129	eTimer_0 Control	*Section 48.3.2.24*
0x142	ME_PCTL130	eTimer_3 Control	*Section 48.3.2.24*
0x143	ME_PCTL131	eTimer_2 Control	*Section 48.3.2.24*
0x144–0x18E	Reserved		
0x18F	ME_PCTL207	FlexCAN_1 Control	*Section 48.3.2.24*
0x190–0x19B	Reserved		
0x19C	ME_PCTL220	LINFlexD_1 Control	*Section 48.3.2.24*

Table 658. MC_ME register description(Continued)

Address	Name	Description	Location
0x19D–0x1A2	Reserved		
0x1A3	ME_PCTL227	DSPI_2 Control	*Section 48.3.2.24*
0x1A4–0x1B3	Reserved		
0x1B4	ME_PCTL244	CTU Control	*Section 48.3.2.24*
0x1B5–0x1BF	Reserved		
0x1C0	ME_CS	Core Status	*Section 48.3.2.25*
0x1C4	ME_CCTL0	Main Core_0 Control	*Section 48.3.2.26*
0x1C6–0x1DF	Reserved		
0x1E0	ME_CADDR0	Main Core_0 Address	*Section 48.3.2.27*
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*

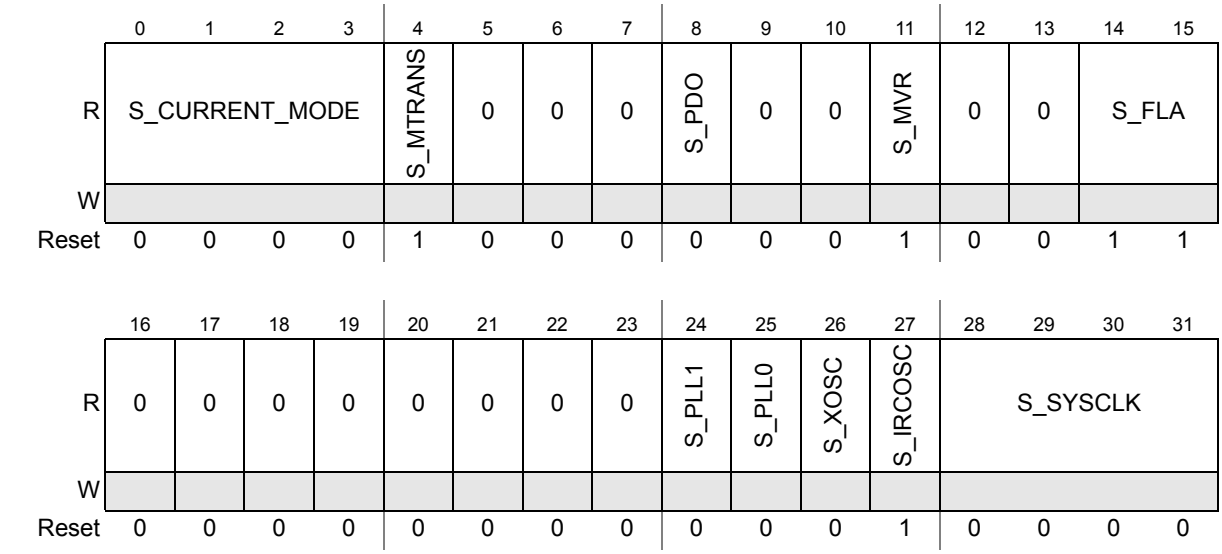
48.3.2 Register description

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.

48.3.2.1 Global Status Register (ME_GS)

Address: 0x000 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_CURRENT_MODE				S_MTRANS	0	0	0	S_PDO	0	0	S_MVR	0	0	S_FLA	
W																
Reset	0	0	0	0	1	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	S_PLL1	S_PLL0	S_XOSC	S_IRCOSC	S_SYSCLK			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0

Figure 632. Global Status Register (ME_GS)

This register contains global mode status.

Table 659. 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 659. 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

48.3.2.2 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 633. 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 660. 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

48.3.2.3 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 634. 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 661. 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

48.3.2.4 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 635. Interrupt Status Register (ME_IS)

This register provides the current interrupt status.

Table 662. 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_CCTL0 registers is attempted while a mode transition is in progress. 0 No write to an ME_CCTL0 register was attempted during an ongoing mode transition 1 A write to an ME_CCTL0 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 662. 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

48.3.2.5 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 636. Interrupt Mask Register (ME_IM)

This register controls whether an event generates an interrupt or not.

Table 663. 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 663. 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

48.3.2.6 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	S_MRIG	S_MTI	S_MRI	S_DMA	S_NMA	S_SEA
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 637. Invalid Mode Transition Status Register (ME_IMTS)

This register provides the status bits for the possible causes of an invalid mode interrupt.

Table 664. ME_IMTS field descriptions

Field	Description
26 S_MRIG	Mode Request Ignored Status This bit is set whenever a new mode is requested when a transition to the SAFE mode is in progress. It is cleared by writing 1 to this bit. 0 Mode transition requested is not illegal 1 Mode transition requested is illegal
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 48.4.5: Mode Transition Interrupts* 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

Table 664. ME_IMTS field descriptions(Continued)

Field	Description
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
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

48.3.2.7 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 638. 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 665. 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 665. 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 665. 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 665. 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_PCTLn register. Please refer to *Table 658* for the ME_PCTLn locations actually occupied, which in turn indicates which peripherals are reported in the ME_DMTS register.

48.3.2.8 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 639. RESET Mode Configuration Register (ME_RESET_MC)

This register configures system behavior during RESET mode. Please refer to *Table 666* for details.

Note: Half word access is only allowed to the lower half word of the register.

Table 666. 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 666. 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.

48.3.2.9 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 640. TEST Mode Configuration Register (ME_TEST_MC)

This register configures system behavior during TEST mode. Please refer to *Table 667* for details.

Note: Half word access is only allowed to the lower half word of the register.

Table 667. 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 667. 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

48.3.2.10 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 641. SAFE Mode Configuration Register (ME_SAFE_MC)

This register configures system behavior during SAFE mode. Please refer to *Table 668* for details.

Note: Half word access is only allowed to the lower half word of the register.

Table 668. 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 668. 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

48.3.2.11 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 642. DRUN Mode Configuration Register (ME_DRUN_MC)

This register configures system behavior during DRUN mode. Please refer to *Table 669* for details.

Note: Half word access is only allowed to the lower half word of the register.

Table 669. 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 669. 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.

48.3.2.12 RUNn Mode Configuration Register (ME_RUNn_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 643. RUNn Mode Configuration Register (ME_RUNn_MC)

This register configures system behavior during RUN0–3 modes. Please refer to *Table 670* for details.

Note: Half word access is only allowed to the lower half word of the register.

Table 670. ME_RUNn_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 670. ME_RUNn_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

48.3.2.13 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 644. HALT0 Mode Configuration Register (ME_HALT0_MC)

This register configures system behavior during HALT0 mode. Please refer to *Table 671* for details.

Note: Half word access is only allowed to the lower half word of the register.

Table 671. 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 671. 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

48.3.2.14 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 645. STOP0 Mode Configuration Register (ME_STOP0_MC)

This register configures system behavior during STOP0 mode. Please refer to *Table 672* for details.

Note: Half word access is only allowed to the lower half word of the register.

Table 672. 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 672. 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

48.3.2.15 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_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 646. Peripheral Status Register 0 (ME_PS0)

This register provides the status of the peripherals. Please refer to *Table 673* for details.

48.3.2.16 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	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_DMAMUX_0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 647. Peripheral Status Register 1 (ME_PS1)

This register provides the status of the peripherals. Please refer to *Table 673* for details.

48.3.2.17 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	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_FlexCAN_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. Peripheral Status Register 2 (ME_PS2)

This register provides the status of the peripherals. Please refer to *Table 673* for details.

48.3.2.18 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_ADCSEQ_0	0	0	0	0	0	0	0	0	0	0	0	0	S_ADCSEQ_B	S_ADCSAR_B
W																
Reset	0	0	0	0	0	0	0	0	0	0	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	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 649. Peripheral Status Register 3 (ME_PS3)

This register provides the status of the peripherals. Please refer to *Table 673* for details.

48.3.2.19 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	0	0	0	0	0	0	0	0	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 650. Peripheral Status Register 4 (ME_PS4)

This register provides the status of the peripherals. Please refer to *Table 673* for details.

48.3.2.20 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	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_FlexCAN_1	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. Peripheral Status Register 6 (ME_PS6)

This register provides the status of the peripherals. Please refer to *Table 673* for details.

48.3.2.21 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	0	0	0	0	0	0	0	0	0	0	0	S_CTU	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	S_DSPI_2	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 652. Peripheral Status Register 7 (ME_PS7)

This register provides the status of the peripherals. Please refer to *Table 673* for details.

Table 673. 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

48.3.2.22 Run Peripheral Configuration Register n (ME_RUN_PCn)

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 653. Run Peripheral Configuration Registers n (ME_RUN_PCn)

These registers configure eight different types of peripheral behavior during run modes.

Table 674. ME_RUN_PCn 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

Table 674. ME_RUN_PCn field descriptions(Continued)

Field	Description
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

48.3.2.23 Low-Power Peripheral Configuration Register n (ME_LP_PCn)

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 654. Low-Power Peripheral Configuration Register n (ME_LP_PCn)

These registers configure eight different types of peripheral behavior during non-run modes.

Table 675. ME_LP_PCn 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

48.3.2.24 Peripheral Control Register n (ME_PCTLn)

Address: 0x0DE (ME_PCTL30)
0x0E4 (ME_PCTL36)
0x0E6 (ME_PCTL38)
0x10F (ME_PCTL79)
0x11C (ME_PCTL92)
0x122 (ME_PCTL98)
0x123 (ME_PCTL99)
0x130 (ME_PCTL112)
0x131 (ME_PCTL113)
0x13E (ME_PCTL126)
0x13F (ME_PCTL127)
0x140 (ME_PCTL128)
0x141 (ME_PCTL129)
0x142 (ME_PCTL130)
0x143 (ME_PCTL131)
0x18F (ME_PCTL207)
0x19C (ME_PCTL220)
0x1A3 (ME_PCTL227)
0x1B4 (ME_PCTL244)

Access: User read/write, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7
R	0	DBG_F	LP_CFG			RUN_CFG		
W								
Reset	0	0	0	0	0	0	0	0

Figure 655. Peripheral Control Register n (ME_PCTLn)

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 676. ME_PCTLn field descriptions

Field	Description
1 DBG_F	Peripheral control in debug mode This bit controls the state of the peripheral in debug mode. 0 Peripheral state depends on RUN_CFG/LP_CFG bits and the chip mode 1 Peripheral is frozen if not already frozen in chip modes. **Note:** This feature is useful to freeze the peripheral state while entering debug. For example, this may be used to prevent a reference timer from running while making a debug accesses.

Table 676. ME_PCTLn field descriptions(Continued)

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

48.3.2.25 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 656. Core Status Register (ME_CS)

This register provides the status of each core.

Table 677. 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

48.3.2.26 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 657. 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 678. 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 678. 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.

48.3.2.27 Core Address Register 0 (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 658. Core Address Register 0 (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 679. 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

48.4 Functional description

48.4.1 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

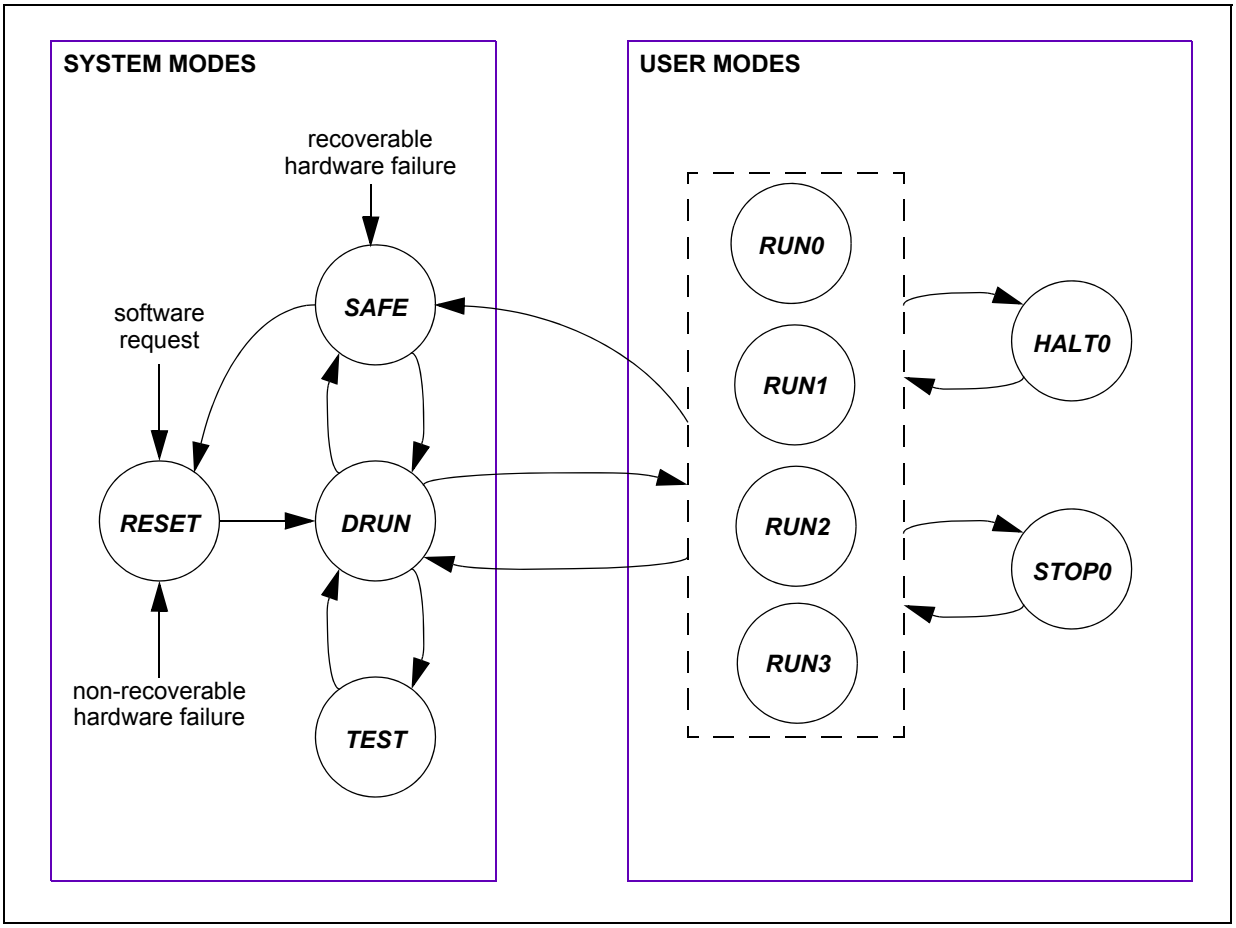
- the first time with the value of the key (0x5AF0) into the KEY bit field and the required target mode into the TARGET_MODE bit field,
- and the second time with the inverted value of the key (0xA50F) into the KEY bit field and the required target mode into the TARGET_MODE bit field.

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 48.4.5: Mode Transition Interrupts*.

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 659. MC_ME mode diagram



48.4.2 Modes details

48.4.2.1 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.

48.4.2.2 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.

48.4.2.3 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.

48.4.2.4 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.

48.4.2.5 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.

48.4.2.6 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.

48.4.2.7 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.

48.4.3 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.

48.4.3.1 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 48.4.5: Mode Transition Interrupts* 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.

48.4.3.2 Target Mode Configuration Loading

On completion of the *Section 48.4.3.1: Target Mode Request* step, the target mode configuration from the ME_<*target mode*>_MC register is loaded to start the resources (voltage sources, clock sources, flash, pads, etc.) control process.

An overview of resource control possibilities for each mode is shown in *Table 680*. A '√' indicates that a given resource is configurable for a given mode.

Table 680. 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 680. 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

48.4.3.3 Peripheral Clocks Disable

On completion of the *Section 48.4.3.1: Target Mode Request* step and the system clock frequency ramp-down of the *Section 48.4.3.13: System Clock Switching* 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_PCTLn

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 48.4.6: Peripheral Clock Gating* 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.

48.4.3.4 Processor Low-Power Mode Entry

If, on completion of the *Section 48.4.3.3: Peripheral Clocks Disable* step, the mode transition is to the HALT0, STOP0 mode, the MC_ME requests the processor to enter its stopped state. The processor acknowledges its stop state request bit after completing all outstanding bus transactions.

If, on completion of the *Section 48.4.3.3: Peripheral Clocks Disable* step, the mode transition is from a non-low-power mode to another non-low-power mode, the MC_ME requests the processor which is configured in the ME_CCTL0 registers to be disabled to

enter its stopped state. The processor acknowledges its stop state request bit after completing all outstanding bus transactions.

48.4.3.5 Processor Clocks Disable

If, on completion of the *Section 48.4.3.4: Processor Low-Power Mode Entry* step, and the processor which are configured in the ME_CCTL0 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.

48.4.3.6 System Clock Disable

If, on completion of the *Section 48.4.3.5: Processor Clocks Disable* step, all processor clocks have been disabled, the system clock used for non-processor system functions and memory is disabled.

48.4.3.7 Clock Sources Switch-On

On completion of the *Section 48.4.3.4: Processor Low-Power Mode Entry* 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.

48.4.3.8 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.**

48.4.3.9 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

48.4.3.10 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_PCTLn, 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.

48.4.3.11 System and Processor Clocks Enable

On completion of the *Section 48.4.3.8: Flash Module Switch-On* step, the MC_ME enables the non-processor system clock and the clocks of the processor which are configured in the ME_CCTL0 register to be disabled in the current mode and to be running in the target mode.

The MC_ME initiates the resetting of the processor which are configured in the ME_CCTL0 register to be running in the target mode and which have their RMC bits in the ME_CADDR0 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 48.4.3.13: System Clock Switching* process is completed.

48.4.3.12 Processor Low-Power Mode Exit

On completion of the *Section 48.4.3.11: System and Processor Clocks Enable* step, the MC_ME requests the processor which are configured in the ME_CCTL0 registers to be disabled in the current mode and to be running in the target mode.

48.4.3.13 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 48.4.3.7: Clock Sources Switch-On* 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 48.4.3.3: Peripheral Clocks Disable* 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 681. MC_ME system clock selection overview

System clock source	Mode						
	RESET	TEST	SAFE	DRUN	RUN 0–3	HALT 0	STOP 0
16 MHz internal RC oscillator	√ (default)	√ (default)	√ (default)	√ (default)	√ (default)	√ (default)	√ (default)
external crystal oscillator		√		√	√	√	√

Table 681. MC_ME system clock selection overview(Continued)

System clock source	Mode						
	RESET	TEST	SAFE	DRUN	RUN 0–3	HALT 0	STOP 0
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.

48.4.3.14 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 48.4.3.3: Peripheral Clocks Disable* process has completed.

48.4.3.15 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 48.4.3.13: System Clock Switching* process has completed.

48.4.3.16 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 48.4.3.5: Processor Clocks Disable* and *Section 48.4.3.6: System Clock Disable* processes have completed.

48.4.3.17 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:

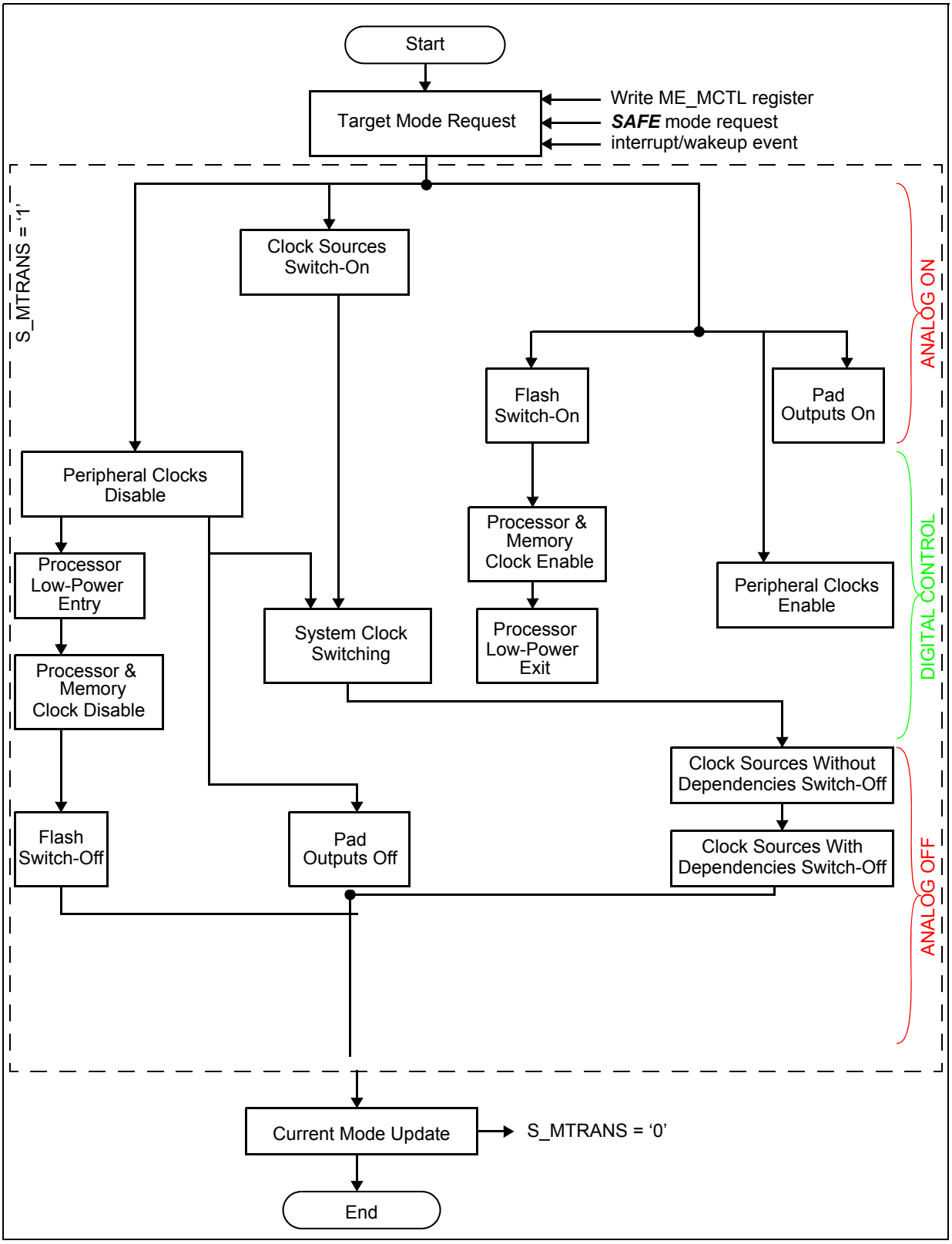
- all the updated status bits in the ME_GS register match the configuration specified in the ME_<*target mode*>_MC register
- power sequences are done
- clock disable/enable process is finished
- processor low-power mode (stop) entry and exit processes are finished

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 660. MC_ME transition diagram



48.4.4 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.

48.4.5 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.

48.4.5.1 Invalid Mode Configuration Interrupt

Whenever a write operation is attempted to the ME_<*mode*>_MC registers violating the protection rules mentioned in the *Section 48.4.4: Protection of mode configuration registers*, 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_PCTLn 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’.

48.4.5.2 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’.

48.4.5.3 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).

48.4.5.4 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.

48.4.6 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_PCTLn 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_PCTLn registers.

Any modifications to the ME_RUN_PC0–7, ME_LP_PC0–7, and ME_PCTLn registers do not affect the clock gating behavior until a new mode transition request is generated.

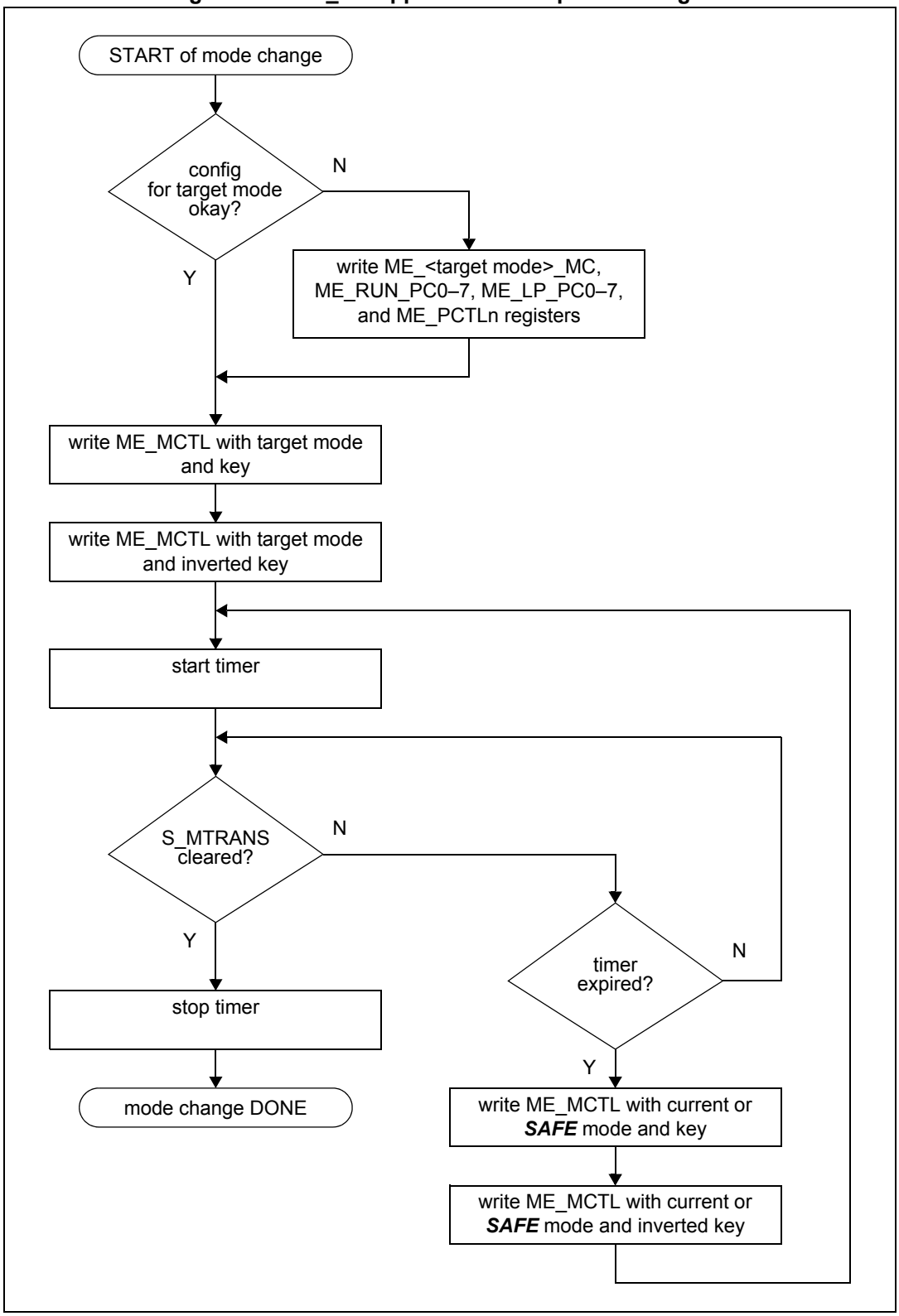
Whenever the chip enters a debug session during any mode, the following occurs for each peripheral:

- The clock is gated if the DBG_F bit of the associated ME_PCTLn register is set. Otherwise, the peripheral clock gating status depends on the RUN_CFG and LP_CFG bits.

48.4.7 Application example

Figure 661 shows an example application flow for requesting a mode change and then waiting until the mode transition has completed.

Figure 661. MC_ME application example flow diagram



49 e200z0h Core Debug Support

This chapter describes the debug features of the e200z0h core.

49.1 Overview

Internal debug support in the e200z0h 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 are protected from software modification.

Software debug facilities are defined as part of Power Architecture Book E. e200z0h supports a subset of these defined facilities. In addition to the facilities defined in Power Architecture Book E, e200z0h provides additional flexibility and functionality in the form of linked instruction and data breakpoints, and sequential debug event detection. These features are also available to a hardware-based debugger.

The e200z0h core provides support for an external Nexus real-time debug module. Real-time debugging in a e200z0h-based system is supported by a Nexus Class 1 module.

49.2 Software debug facilities

e200z0h 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), a set of address compare registers (IAC1, IAC2, IAC3, IAC4, DAC1, and DAC2), a Debug Status Register (DBSR) for enabling and recording various kinds of debug events, and a special Debug interrupt type built into the interrupt mechanism. The debug facilities also provide a mechanism for software-controlled processor reset in a debug environment.

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 DBCR0[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 DBCR0[EDM] being set) no Debug interrupts 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.

49.2.1 Power Architecture Book E compatibility

The e200z0h core implements a subset of the Power Architecture Book E 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
- Data value compares are not supported.

49.2.2 Additional debug facilities

In addition to the debug functionality defined in Power Architecture Book E, e200z0h provides capability to link instruction and data breakpoints, and also provides a sequential breakpoint control mechanism.

e200z0h also defines two new debug events (CIRPT, CRET) for debugging around critical interrupts.

In addition, e200z0h implements the Debug APU, which when enabled allows Debug Interrupts to utilize a dedicated set of save/restore registers (DSRR0, DSRR1) for saving state information when a Debug Interrupt occurs, and for restoring this state information at the end of a debug interrupt handler by means of the se_rfdi instruction.

49.2.3 Hardware debug facilities

The e200z0h 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 e200z0h core, read and write internal state and debug facilities, single-step instructions, and resume normal execution.

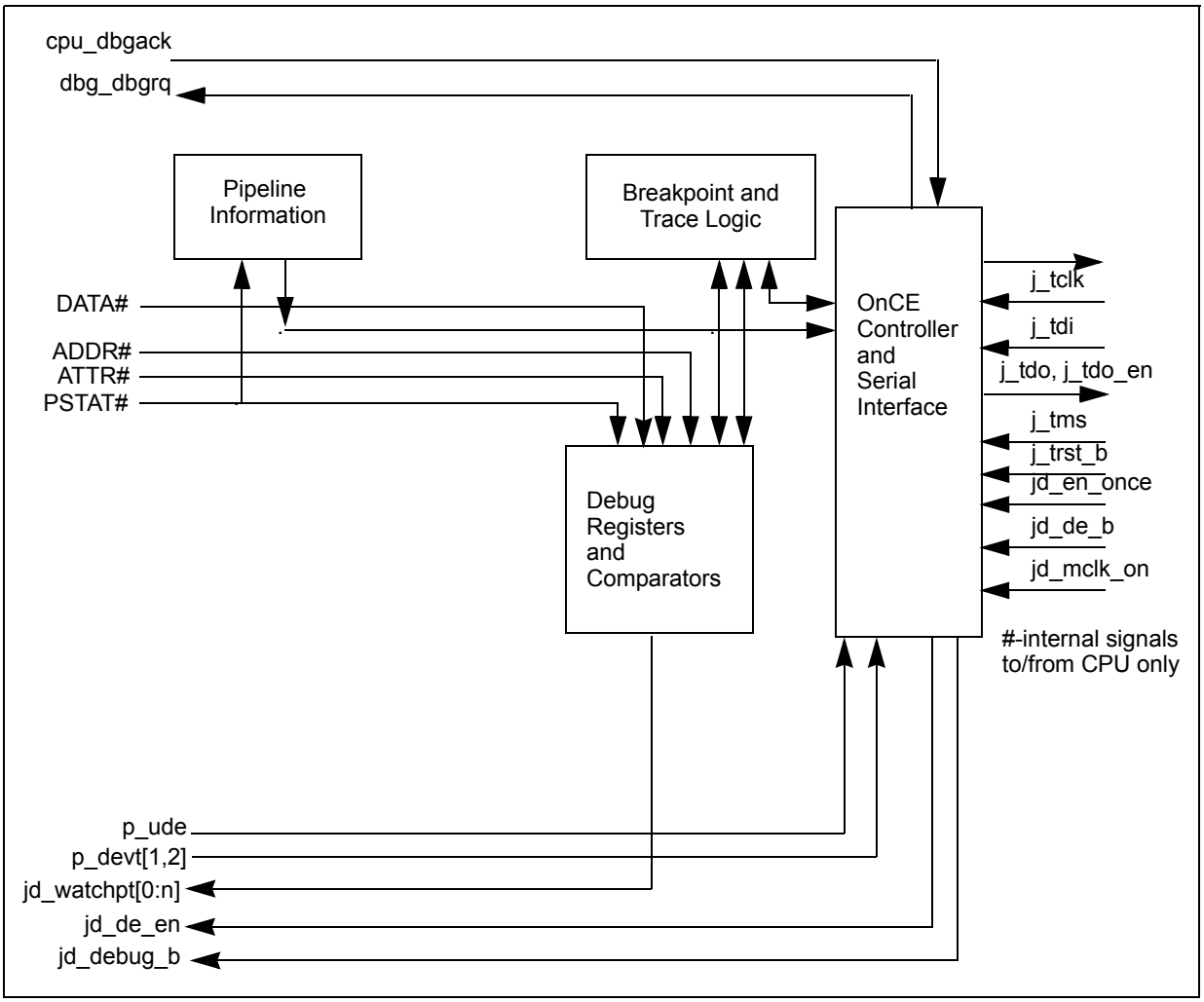
Hardware Debug is enabled by setting the External Debug mode enable bit in Debug Control register 0 (DBCR0[EDM]). Setting DBCR0[EDM] overrides the Internal Debug mode enable bit DBCR0[IDM]. When the Hardware Debug facility is enabled, software is blocked from modifying the debug facilities. In addition, because resources are “owned” by the Hardware debugger, inconsistent values may be present if software attempts to read debug-related resources.

When hardware debug is enabled by setting DBCR0[EDM] = 1, the registers and resources described in *Section 49.4: Debug registers* are reserved for use by the external debugger. The same events described in *Section 49.3: Software debug events and exceptions* are also used for external debugging, but exceptions are not generated to running software. Debug events enabled in the respective DBCR[0–2] registers are recorded in the DBSR regardless of MSR[DE], and no debug interrupts are generated. Instead, the CPU enters Debug mode when an enabled event causes a DBSR bit to become set. DBCR0[EDM] may only be written through the OnCE™ (On-Chip Emulation) port.

Access to most debug resources (registers) requires that the core clock (m_clk) be running in order to perform write accesses from the external hardware debugger.

Figure 662 shows the e200z0h debug resources.

Figure 662. e200z0h debug resources



49.3 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 (DBCR0[EDM] must be cleared). 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 (DBCR0–2). The Unconditional Debug Event (UDE) is an exception to this rule; it is always enabled. Once a Debug Status Register (DBSR) bit is set (other than MRR), if Debug interrupts are enabled by MSR[DE], a Debug interrupt is 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[EDM] = 0. 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 is delayed until MSR[DE] is later set to ‘1’.

When a Debug Status Register bit is set while MSR[DE] = 0 and DBCR0[EDM] = 0, an Imprecise Debug Event flag (DBSR[IDE]) also is 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 Power Architecture Book E, as follows:

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 Taken debug events
7. Return debug events
8. Unconditional debug events

In addition, e200z0h defines additional debug events:

- The External debug events DEVT1 and DEVT2 which are described in *Section 49.3.11: External debug event.*
- The Critical Interrupt Taken debug event CIRPT which is described in *Section 49.3.8: Critical interrupt taken debug event.*
- The Critical Return debug event CRET which is described in *Section 49.3.10: Critical return debug event.*

The e200z0h debug configuration supports most of these event types. Unsupported Power Architecture Book E defined functionality is as follows:

- Instruction Address Compare and Data Address Compare real address mode is not supported.
- Data Value Compare mode is not supported.

A brief description of each of the event types follows. In these descriptions, DSRR0 and DSRR1 are used, assuming that the Debug APU is enabled. If it is disabled, use CSRR0 and CSRR1, respectively.

49.3.1 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, IAC1, IAC2, IAC3, and IAC4 Registers. Instruction Address compares may specify user/supervisor mode and instruction space (MSR[IS]), 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 do 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 instructions. Each halfword fetched by the instruction fetch unit is marked with a set of bits indicating whether an Instruction Address

Compare occurred on that halfword. Debug exceptions occur if enabled and a 16-bit instruction, or the first halfword of a 32-bit instruction, is tagged with an IAC hit.

49.3.2 Data address compare event

Data Address Compare debug events occur when enabled and execution of a load or store class instruction results in a data access that meets the criteria specified in the DBCR0, DBCR2, DAC1, and DAC2 Registers. Data address compares may specify user/supervisor mode and data space (MSR[DS]), 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]. Two address compare values (DAC1, DAC2) are provided.

Note: In contrast to the Power Architecture Book E definition, Data Address Compare events on e200z0h do not prevent the load or store class instruction from completing. If a load or store class instruction completes successfully without a Data TLB or 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 (or CSRR0 if the Debug APU is disabled) is the address of the instruction following the load or store class instruction.

If a load or store class instruction does not complete successfully due to a Data Storage exception, and a Data Address Compare debug exception also occurs, the result is an imprecise Debug interrupt, the address value saved in DSRR0 (or CSRR0 if the Debug APU is disabled) is the address of the load or store class instruction, and the DBSR[IDE] bit is set. In addition to occurring when DBCR0[IDM] = 1, this circumstance can also occur when DBCR0[EDM] = 1.

Note: DAC events are not recorded or counted if a lmw or stmw instruction is interrupted prior to completion by a critical input or external input interrupt.

Note: DAC events are not signaled on the second portion of a misaligned load or store that is broken up into two separate accesses.

49.3.3 Linked instruction address and data address compare event

Data Address Compare debug events may be ‘linked’ with an Instruction Address Compare event by setting the DAC1LNK and/or DAC2LNK control bits in DBCR2 to further refine when a Data Address Compare debug event is generated. DAC1 may be linked with IAC1, and DAC2 (when not used as a mask or range bounds register) may be linked with IAC3. When linked, a DAC1 (or DAC2) debug event occurs when the same instruction which generates the DAC1 (or DAC2) ‘hit’ also generates an IAC1 (or IAC3) ‘hit’. When linked, the IAC1 (or IAC3) event is not recorded in the Debug Status register, regardless of whether a corresponding DAC1 (or DAC2) event occurs, or whether the IAC1 (or IAC3) 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 DBCR0, DBCR2, DAC1, and DAC2 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 and DAC2 status bits in the DBSR are used for recording these events. The IAC1 and IAC3 status bits are not set if the corresponding Instruction Address Compare register is linked.

Linking is enabled using control bits in DBCR2.

Note: Linked DAC events are not recorded if a load multiple word or store multiple word instruction is interrupted prior to completion by a critical input or external input interrupt.

49.3.4 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.

49.3.5 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 is taken (either an unconditional branch, or a conditional branch whose branch condition is true), and MSR[DE] = 1 or DBCR0[EDM] = 1. Branch Taken debug events are not recognized if MSR[DE] = 0 and DBCR0[EDM] = 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 is recorded in DSRR0.

49.3.6 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 or DBCR0[EDM] = 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 ***sc*** instruction does not fall into the category of an instruction whose execution is suppressed, because the instruction actually executes and then generates a System Call interrupt. In this case, the Instruction Complete debug exception is also 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 is recorded in DSRR0.

Instruction Complete debug events are not recognized if MSR[DE] = 0 and DBCR0[EDM] = 0 at the time of execution of the instruction, thus DBSR[IDE] is not generally set by an ICMP debug event.

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 or DBCR0[EDM] = 1.

49.3.7 Interrupt taken debug event

An Interrupt Taken debug event (IRPT) occurs if Interrupt Taken debug events are enabled (DBCR0[IRPT] = 1) and a non-critical interrupt occurs. Only non-critical class interrupts 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 is the address of the non-critical interrupt handler.

49.3.8 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 (other than a Debug interrupt when the Debug APU is disabled) occurs. Only critical class interrupts 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 is the address of the critical interrupt handler. Note that this debug event should not normally be enabled unless the Debug APU is also enabled to avoid corruption of CSRR0/1.

49.3.9 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 and DBCR0[EDM] = 0 at the time of the execution of the **se_rfi** (such as 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 occurs provided there exists no higher priority exception which is enabled to cause an interrupt. Debug Save/Restore Register 0 is set to the address of the **se_rfi** instruction.

49.3.10 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 and DBCR0[EDM] = 0 at the time of the execution of the **se_rfci** (such as 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 occurs provided there exists no higher priority exception which is enabled to cause an interrupt. Debug Save/Restore Register 0 is set to the address of the **se_rfci** instruction. Note that this debug event should not normally be enabled unless the Debug APU is also enabled to avoid corruption of CSRR0/1.

49.3.11 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. 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.

49.3.12 Unconditional debug event

An Unconditional debug event (UDE) occurs when the Unconditional Debug Event (*p_ude*) input transitions to the asserted state, and either DBCR0[IDM] = 1 or DBCR0[EDM] = 1. 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.

49.4 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 by software is conditioned by the External Debug Mode control bit (DBCR0[EDM]) which can be set by the hardware debug port. If DBCR0[EDM] is set, software is prevented from modifying debug register values. Execution of a **mtspr** instruction targeting a debug register does not cause modifications to occur. In addition, because the external debugger hardware may be manipulating debug register values, the state of these registers is not guaranteed to be consistent if accessed (read) by software with a **mfspr** instruction, other than the DBCR0[EDM] bit itself.

49.4.1 Debug address and value registers

Instruction Address Compare registers IAC1, IAC2, IAC3, and IAC4 are used to hold instruction addresses for address comparison purposes. In addition, IAC2 and IAC4 hold mask information for IAC1 and IAC3 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.

Data Address Compare registers DAC1 and DAC2 are used to hold data access addresses for address comparison purposes. In addition, DAC2 holds mask information for DAC1 when *Address Bit Match* compare mode is selected.

49.4.2 Debug control and status registers

Debug Control Registers 0–2 (DBCR0, DBCR1, DBCR2) 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 or External Debug Mode is enabled.

e200z0h requires that a context synchronizing instruction follow a **mtspr** DBCR0–2 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 **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 and DBCR2 are used, even though the corresponding event(s) may be disabled (via DBCR0) from setting DBSR flags.

49.4.2.1 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 flags. e200z0h adds some implementation specific bits to this register.

SPR: 308 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	EDM	IDM	RST		ICMP	BRT	IRPT	TRAP	IAC1	IAC2	IAC3	IAC4	DAC1		DAC2		RET	0			DEVT1	DEVT2	0		CIRPT	CRET	0					
Reset(1)	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

1. DBCR0[EDM] is affected by *j_trst_b* or *m_por* assertion, and while in the Test_Logic_Reset state, but not by *p_reset_b*. All other bits are reset by processor reset *p_reset_b* as well as by *m_por*.

Figure 663. DBCR0 register

Table 682. 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. Events do not cause the CPU to vector to interrupt code. Software is not permitted to write to debug registers {DBCRx, DBSR, DBCNT, IAC1–4, DAC1–2}.
	Programming Notes: It is recommended that debug status bits in the Debug Status Register 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 unless EDM is set. 1 Debug exceptions are enabled. Enabled debug events 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, causes a Debug interrupt.
2:3 RST	Reset Control 00 No function 01 Reserved 10 *p_resetout_b* pin asserted by Debug Reset Control. Allows external device to initiate processor 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

Table 682. DBCR0 field descriptions(Continued)

Field	Description
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
17:20	Reserved
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
23:24	Reserved
25 CIRPT	Critical Interrupt Taken Debug Event Enable 0 CIRPT debug events are disabled 1 CIRPT debug events are enabled

Table 682. DBCR0 field descriptions(Continued)

Field	Description
26 CRET	Critical Return Debug Event Enable 0 CRET debug events are disabled 1 CRET debug events are enabled
27:31	Reserved

49.4.2.2 Debug Control Register 1 (DBCR1)

Debug Control Register 1 is used to configure Instruction Address Compare operation.

SPR: 309 Access: Read/Write

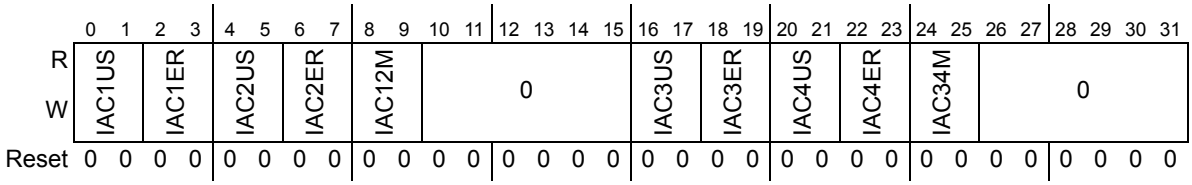


Figure 664. DBCR1 register

Table 683. 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 in e200z0h (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 in e200z0h (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

Table 683. DBCR1 field descriptions(Continued)

Field	Description
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.
10:15	Reserved
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 in e200z0h (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
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 in e200z0h (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.
26:31	Reserved

49.4.2.3 Debug Control Register 2 (DBCR2)

Debug Control Register 2 is used to configure Data Address Compare and Data Value Compare operation.

SPR: 310 Access: Read/Write

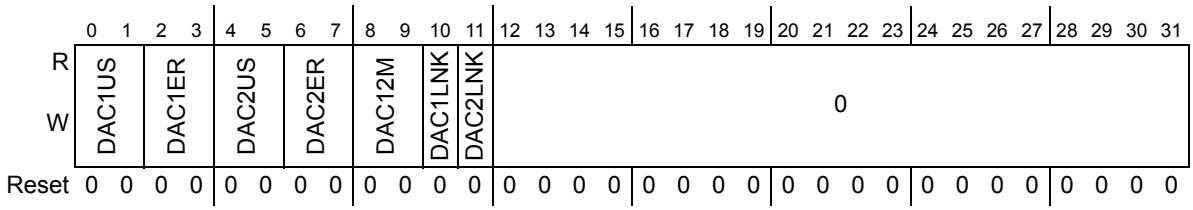


Figure 665. DBCR2 register

Table 684. 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 in e200z0h (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 in e200z0h (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.

Table 684. DBCR2 field descriptions(Continued)

Field	Description
10 DAC1LNK	Data Address Compare 1 Linked 0 no affect 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
11 DAC2LNK	Data Address Compare 2 Linked 0 no affect 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* because DAC2 debug events are not generated in the other compare modes.
12:31	Reserved for Data Value Compare control (not supported by e200z0h)

49.4.2.4 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 or External Debug Mode is enabled. When debug interrupts are enabled (MSR[DE] = 1, DBCR0[IDM] = 1, and DBCR0[EDM] = 0), a set bit in DBSR other than MRR or VLES causes a debug interrupt to be generated. The debug interrupt handler is responsible for clearing DBSR bits prior to returning to normal execution. The Power Architecture VLE APU adds the DBSR[VLES] status bit to indicate debug events occurring due to a Power Architecture VLE instruction.

SPR: 304 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	IDE	UDE	MRR		ICMP	BRT	IRPT	TRAP	IAC1	IAC2	IAC3	IAC4	DAC1R	DAC1W	DAC2R	DAC2W	RET	0			DEVT1	DEVT2	0		CIRPT	CRET	VLES		0			
Reset	0	0	0	1	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 666. DBSR register

Table 685. DBSR field descriptions

Field	Description
0 IDE	Imprecise Debug Event Set to 1 if MSR[DE] = 0 and DBCR0[EDM] = 0 and a debug event causes its respective Debug Status Register bit to be set to 1. It may also be set to '1' if DBCR0[EDM] = 1 and an imprecise debug event occurs due to a DAC event on a load or store which is terminated with error.
1 UDE	Unconditional Debug Event Set to 1 if an Unconditional debug event occurred.
2:3 MRR	Most Recent Reset 00 No reset occurred because these bits were last cleared by software 01 A hard reset occurred because these bits were last cleared by software 10 Reserved 11 Reserved
4 ICMP	Instruction Complete Debug Event Set to 1 if an Instruction Complete debug event occurred.
5 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 IAC1	Instruction Address Compare 1 Debug Event Set to 1 if an IAC1 debug event occurred.
9 IAC2	Instruction Address Compare 2 Debug Event Set to 1 if an IAC2 debug event occurred.
10 IAC3	Instruction Address Compare 3 Debug Event Set to 1 if an IAC3 debug event occurred.
11 IAC4	Instruction Address Compare 4 Debug Event Set to 1 if an IAC4 debug event occurred.
12 DAC1R	Data Address Compare 1 Read Debug Event Set to 1 if a read-type DAC1 debug event occurred while DBCR0[DAC1] = 0b10 or DBCR0[DAC1] = 0b11
13 DAC1W	Data Address Compare 1 Write Debug Event Set to 1 if a write-type DAC1 debug event occurred while DBCR0[DAC1] = 0b01 or DBCR0[DAC1] = 0b11
14 DAC2R	Data Address Compare 2 Read Debug Event Set to 1 if a read-type DAC2 debug event occurred while DBCR0[DAC2] = 0b10 or DBCR0[DAC2] = 0b11
15 DAC2W	Data Address Compare 2 Write Debug Event Set to 1 if a write-type DAC2 debug event occurred while DBCR0[DAC2] = 0b01 or DBCR0[DAC2] = 0b11
16 RET	Return Debug Event Set to 1 if a Return debug event occurred
17:20	Reserved

Table 685. DBSR field descriptions(Continued)

Field	Description
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
23:24	Reserved
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 Power Architecture VLE Instruction. Undefined for IRPT, CIRPT, DEVT[1,2], and UDE events
28:31	Reserved

49.5 External debug support

External debug support is supplied through the e200z0h OnCE controller serial interface which allows access to internal core registers and other system state while in External Debug Mode (EDM). All debug resources including DBCR0–2, DBSR, IAC1–4, and DAC1–2 are accessible through the serial OnCE interface in External Debug mode. Setting the DBCR0[EDM]bit to ‘1’ through the OnCE interface enables External Debug mode and disables software updates to the debug registers. When DBCR0[EDM] is set, debug events enabled to set respective DBSR status bits also cause the CPU to enter Debug Mode as opposed to generating Debug Interrupts. In Debug Mode, the CPU is halted at a recoverable boundary, and an external Debug Control Module may control CPU operation through the OnCE logic. No Debug interrupts can occur while DBCR0[EDM] remains set.

Note: On the initial setting of DBCR0[EDM] to ‘1’, other bits in DBCR0 remain unchanged. After DBCR0[EDM] has been set, all debug register resources may be subsequently controlled through the OnCE interface. The DBSR register should be cleared as part of the process of enabling external debug activity. The core 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.

Note: It is intended for the core to remain in External Debug mode (DBCR0[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.

Note: DBCR0[EDM] operation is blocked if OnCE operation is disabled (jd_en_once negated) regardless of whether it is set or cleared. This means that if DBCR0[EDM] was previously set, and then jd_en_once is negated (this should not occur), entry into debug mode is blocked, all events are blocked, and watchpoints are 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), or the DBCR0[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 and DBSR 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.

49.5.1 OnCE introduction

The e200z0h on-chip emulation circuitry (OnCE™/Nexus Class 1 interface) provides a means of interacting with the e200z0h 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.

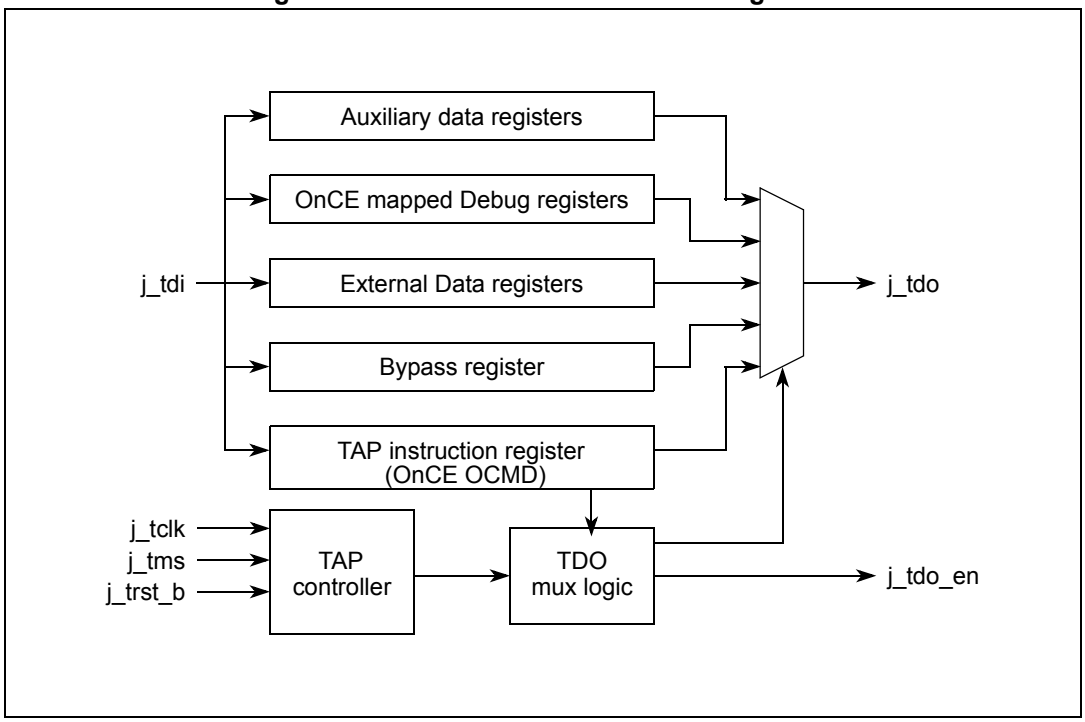
The OnCE logic provides for Nexus Class 1 static debug capability (utilizing the same set of resources available to software while in Internal Debug mode), and is present in all e200z0h-based designs. The OnCE module also provides support for directly integrating a Nexus Class 2 or Class 3 Real-Time Debug unit with the e200z0h core for development of real-time systems where traditional static debug is insufficient. The partitioning between a OnCE module and a connected Nexus module to provide real-time debug allows for capability and cost trade-offs to be made.

The e200z0h core is designed to be a module that can be fully integrated. The OnCE TAP controller and associated enabling logic are designed to allow concatenation with an existing JTAG controller if present in the system. Thus, the e200z0h module can be easily integrated with existing JTAG designs or as a stand-alone controller.

In order to enable full OnCE operation, the *jd_enable_once* input signal must be asserted. In some system integrations, this is automatic, because the input is tied asserted. Other integrations may require the execution of the Enable OnCE command via the TAP and appropriate entry of serial data. Exact requirements are 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.

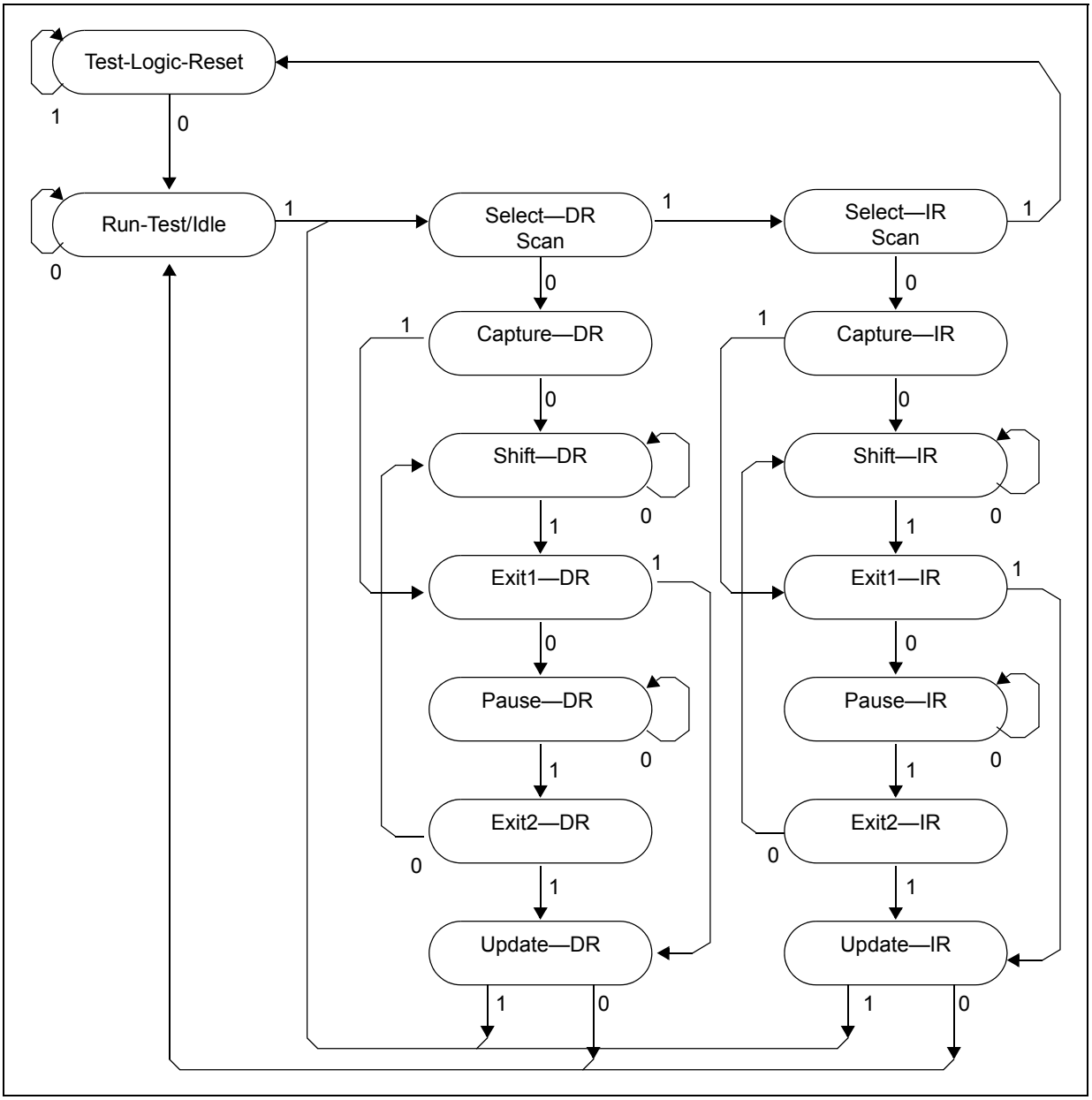
The following figures show the TAP controller state model and the TAP registers implemented by the OnCE logic.

Figure 667. 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 668. OnCE controller implementation



Access to e200z0h processor registers and the contents of memory locations are performed by enabling External Debug mode (setting DBCR0[EDM] to '1'), placing the processor into debug mode, followed by scanning instructions and data into and out of the e200z0h CPU Scan Chain (CPUSCR); execution of scanned instructions by the e200z0h is used as the method to access required data. Memory locations may be read by scanning a load instruction into the e200z0h core, which references 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, stopped, halted, or checkstop states (all indicated via the OnCE Status Register (OSR), *Section 49.5.5.1: e200z0h OnCE Status Register*) 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 indicates 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 49.5.5.2: e200z0h OnCE Command Register (OCMD)*). The CPU then temporarily exits the debug state (but not the debug session) to execute the instruction, and then returns 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 returns 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.

49.5.2 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* causes 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 686 details the primary JTAG/OnCE interface signals.

Table 686. 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.

49.5.3 OnCE internal interface signals

The following paragraphs describe the e200z0h OnCE interface signals to other internal blocks associated with the e200z0h OnCE controller.

49.5.3.1 CPU debug request (*dbg_dbgrq*)

The *dbg_dbgrq* signal is asserted by the e200z0h 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.

49.5.3.2 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 e200z0h OnCE control logic and the rest of the CPU. The CPU core may enter debug mode either through a software or hardware event.

49.5.3.3 CPU address, attributes

The CPU address and attribute information are used by a Nexus Class 2-4 debug unit with information for real-time address trace information.

49.5.3.4 CPU data

The CPU data bus(es) are used to supply a Nexus Class 2-4 debug unit with information for real-time data trace capability.

49.5.4 OnCE interface signals

The following paragraphs describe additional e200z0h OnCE interface signals to other external blocks such as a Nexus Controller and external blocks which may need information pertaining to debug operation.

49.5.4.1 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 enables 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 takes effect 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.

49.5.4.2 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 e200z0h 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* causes the OnCE logic to place the CPU into Debug Mode. Once Debug Mode has been entered, the *jd_de_en* output is 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.

49.5.4.3 e200z0h OnCE debug output (*jd_debug_b*)

The e200z0h 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 e200z0h OnCE Command Register with the GO and EX bits set, and a register specified as either “No Register Selected” or the CPUSCR. This signal remains asserted even though the CPU may enter and exit the debug state for each instruction executed under control of the e200z0h OnCE controller. See *Section 49.5.5.2: e200z0h OnCE Command Register (OCMD)*, for more information on the function of the GO and EX bits. This signal is not normally used by the CPU.

49.5.4.4 e200z0h CPU clock on input (*jd_mclk_on)*

The e200z0h 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 e200z0h 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 occur to access debug resources in the *m_clk* clock domain.

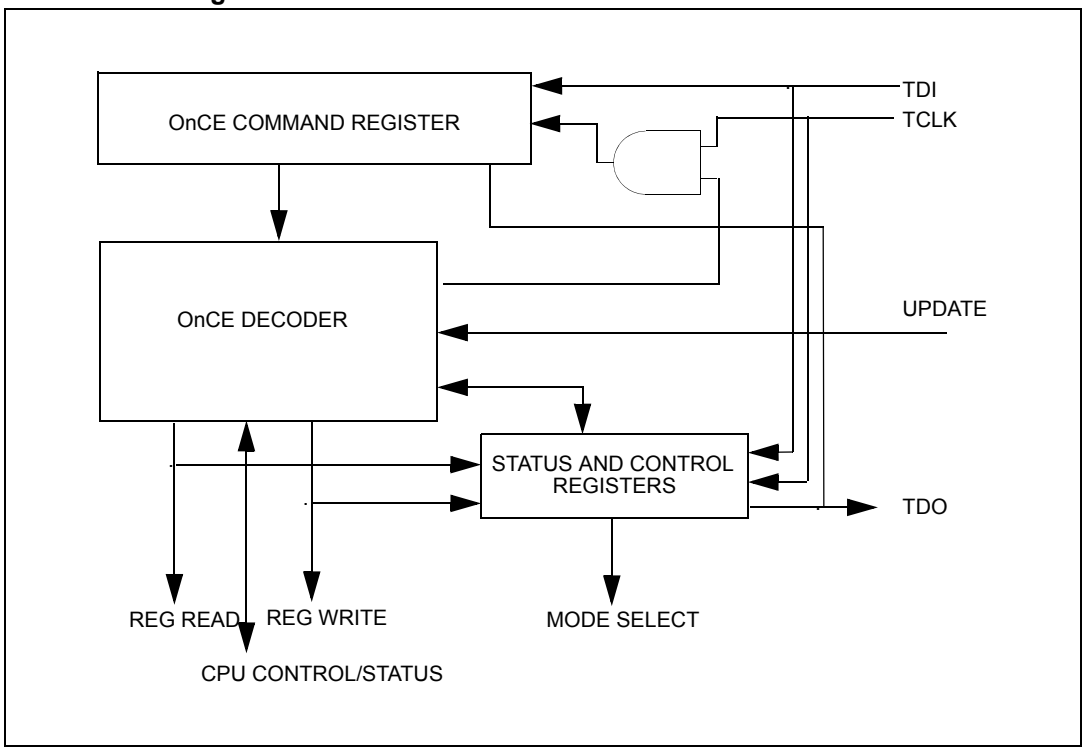
49.5.4.5 Watchpoint events (*jd_watchpt*[0:5])

The *jd_watchpt*[0:5] signals may be asserted by the e200z0h OnCE control logic to signal that a watchpoint condition has occurred. Watchpoints do not cause the CPU to be affected. They are provided to allow external visibility only. Watchpoint events are conditioned by the settings in the DBCR0, DBCR1, and DBCR2 registers.

49.5.5 e200z0h OnCE controller and serial interface

The e200z0h OnCE Controller contains the e200z0h OnCE command register, the e200z0h OnCE decoder, and the status/control register. *Figure 669* is a block diagram of the e200z0h OnCE controller. In operation, the e200z0h OnCE Command register acts as the IR for the e200z0h 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 669. e200z0h OnCE controller and serial interface



49.5.5.1 e200z0h OnCE Status Register

Status information regarding the state of the e200z0h 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.

	0	1	2	3	4	5	6	7	8	9
	MCLK	ERR	CHKSTOP	RESET	HALT	STOP	DEBUG	WAIT	0	1
Reset	0	0	0	1	0	0	0	0	0	0

Figure 670. OnCE Status register

Table 687. OnCE Status register field descriptions

Field	Description
0 MCLK	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, Alignment, Program, etc.) occurred while attempting to perform the instruction single step. In this case, the CPUSCR contains information related to the first instruction of the Interrupt handler, and no portion of the handler will have been executed.
2 CHKSTOP	CHECKSTOP Mode This bit reflects the logic level on the CPU *p_chkstop* output after capture by *j_tclk*.
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	Reserved, set to ‘0’ for 1149.1 compliance
9	Reserved, set to ‘1’ for 1149.1 compliance

49.5.5.2 e200z0h 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 e200z0h OnCE Decoder. 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 transitional through in order for an access to occur. In addition, the Update-DR state must also be transitional 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.

	0	1	2	3	4	5	6	7	8	9
	R/W	GO	EX	RS[0:6]						
Reset(1)	1	0	0	0	0	0	0	0	1	0

1. 10'b1000000010 on assertion of j_trst_b or m_por, or while in the Test_Logic_Reset state

Figure 671. OnCE Command register

Table 688. 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. 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] **Note:** The R/W bit generally ignored for read-only or write-only registers. 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.
1 GO	Go Command Bit 0 Inactive (no action taken) 1 Execute instruction in IR If the GO bit is set, the chip executes the instruction that 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 leaves 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.
2 EX	Exit Command Bit 0 Remain in debug mode 1 Leave debug mode If the EX bit is set, the processor leaves 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 leaves 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 the DBSR is set and DBCR0[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.
3:9 RS	Register Select The Register Select bits define which register is source (destination) for the read (write) operation. *Table 689* indicates the e200z0h OnCE register addresses. Attempted writes to read-only registers are ignored.

Table 689 indicates the e200z0h OnCE register addresses.

Table 689. e200z0h 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 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)
010 0101	Data Address Compare 2 (DAC2)
010 0110	Reserved (DVC1 future use)
010 0111	Reserved (DVC2 future use)
010 1000–010 1011	Reserved
010 1100	Reserved (DBCNT)
010 1101–010 1111	Reserved
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 0100–101 1111	Reserved (do not access)
110 0000–110 1110	Reserved (do not access)
110 1111	Shared Nexus Control Register Select
111 0000–111 1001	General Purpose register selects [0:9]
111 1010	Reserved
111 1011	Reserved
111 1100	Nexus2/3-Access
111 1101	Reserved
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 re-enters debug mode and await further commands. During single-stepping, exception conditions may occur if not properly masked by debug firmware (interrupts, machine checks, bus error conditions, etc.) 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 correspond to the first instruction of the exception handler.

Additionally, the DBCR0[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 is frozen from updates due to debug events regardless of DBCRO[EDM]. They may still be modified during a debug session via a single-stepped **mtspr** instruction if DBCRO[EDM] is programmed to a '0', or via OnCE access if DBCR0[EDM] is set.

49.5.5.3 e200z0h OnCE Control Register (OCR)

The e200z0h OnCE Control Register is a 32-bit register used to force the e200z0h core into debug mode and to enable / disable sections of the e200z0h OnCE control logic. It also provides control over the MMU during a debug session. The control bits are read/write. These bits are only effective while OnCE is enabled (*jd_en_once* asserted).

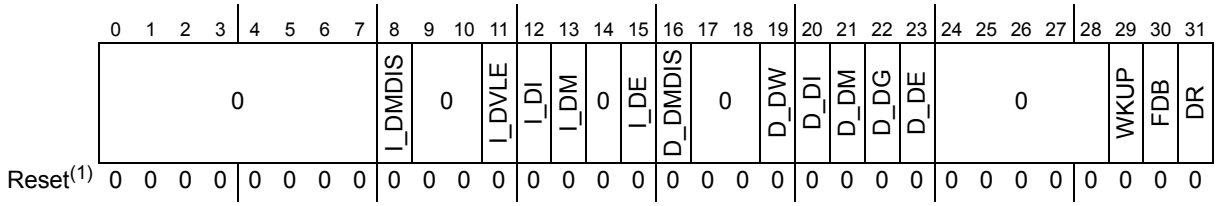


Figure 672. OnCE Control register

1. 0x0000_0000 on m_por, j_trst_b, or entering Test_logic_Reset state

Table 690. OnCE Control register field descriptions

Field	Description
0:7	Reserved
8 I_DMDIS(1)	Instruction Side Debug MMU Disable Control Bit 0 MMU not disabled for debug sessions 1 MMU disabled for debug sessions This bit may be used to control whether the MMU is enabled normally, or whether the MMU is disabled during a debug session for Instruction Accesses. When enabled, the MMU functions normally. When disabled, for Instruction Accesses, no address translation is performed (1:1 address mapping), and the TLB VLE, I,M, and E bits are taken from the OCR bits I_VLE, I_DI, I_DM, and I_DE bits. The W and G bits are assumed '0'. The SX and UX access permission control bits are set to '1' to allow full access. When disabled, no TLB miss or TLB exceptions are generated for Instruction accesses. External access errors can still occur.
9:10	Reserved

Table 690. OnCE Control register field descriptions(Continued)

Field	Description
11 I_DVLE(1)	Instruction Side Debug TLB 'VLE' Attribute Bit This bit is used to provide the 'VLE' attribute bit to be used when the MMU is disabled during a debug session.
12 I_DI(1)	Instruction Side Debug TLB 'I' Attribute Bit This bit is used to provide the 'I' attribute bit to be used for Instruction accesses when the MMU is disabled for Instruction accesses during a debug session.
13 I_DM(1)	Instruction Side Debug TLB 'M' Attribute Bit This bit is used to provide the 'M' attribute bit to be used for Instruction accesses when the MMU is disabled for Instruction accesses during a debug session.
14	Reserved
15 I_DE(1)	Instruction Side Debug TLB 'E' Attribute Bit This bit is used to provide the 'E' attribute bit to be used for Instruction accesses when the MMU is disabled for Instruction accesses during a debug session.
16 D_DMDIS(1)	Data Side Debug MMU Disable Control Bit 0 MMU not disabled for debug sessions 1 MMU disabled for debug sessions This bit may be used to control whether the MMU is enabled normally, or whether the MMU is disabled during a debug session for Data Accesses. When enabled, the MMU functions normally. When disabled, for Data Accesses, no address translation is performed (1:1 address mapping), and the TLB WIMGE bits are taken from the OCR bits D_DW, D_DI, D_DM, D_DG, and D_DE bits. The SR, SW, UR, and UW access permission control bits are set to '1' to allow full access. When disabled, no TLB miss or TLB exceptions are generated for Data accesses. External access errors can still occur.
17:18	Reserved
19 D_DW(1)	Data Side Debug TLB 'W' Attribute Bit This bit is used to provide the 'W' attribute bit to be used for Data accesses when the MMU is disabled for Data accesses during a debug session.
20 D_DI(1)	Data Side Debug TLB 'I' Attribute Bit This bit is used to provide the 'I' attribute bit to be used for Data accesses when the MMU is disabled for Data accesses during a debug session.
21 D_DM(1)	Data Side Debug TLB 'M' Attribute Bit This bit is used to provide the 'M' attribute bit to be used for Data accesses when the MMU is disabled for Data accesses during a debug session.
22 D_DG(1)	Data Side Debug TLB 'G' Attribute Bit This bit is used to provide the 'G' attribute bit to be used for Data accesses when the MMU is disabled for Data accesses during a debug session.
23 D_DE(1)	Data Side Debug TLB 'E' Attribute Bit This bit is used to provide the 'E' attribute bit to be used for Data accesses when the MMU is disabled for Data accesses during a debug session.
24:28	Reserved

Table 690. OnCE Control register field descriptions(Continued)

Field	Description
29 WKUP	Wakeup Request Bit This control bit may be used to force the e200z0h *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 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 causes the processor to enter debug mode, as if the *jd_de_b* input had been asserted. This bit is qualified with DBCR0[EDM], which must be set for FDB to take effect.
31 DR	CPU Debug Request Control Bit This control bit is used to unconditionally request the CPU to enter the Debug Mode. The CPU indicates 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 enters Debug mode at the next instruction boundary.

1. Unused by e200z0 and e200z0h

49.5.6 Access to debug resources

Resources contained in the e200z0h OnCE Module which do not require the e200z0h processor core to be halted for access may be accessed while the e200z0h core is running, and does not interfere with processor execution. Accesses to other resources such as the CPUSCR require the e200z0h 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 e200z0h 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 the e200z0h *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, because the CPU may cause multiple bits of certain registers to change state, reads of certain registers while the CPU is running (DBSR, etc.) 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 691 provides a list of access requirements for OnCE registers.

Table 691. OnCE register access requirements

Register name	Access requirements					Notes
	Requires *jd_en_once* to be asserted	Requires DBCR0 [EDM] = 1	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	Y	Y	Y	N	*(1)	—
DAC2	Y	Y	Y	N	*(1)	—
DBCR0	Y	Y	Y	N	*(1)	*DBCR0[EDM] access only requires *jd_en_once* asserted
DBCR1	Y	Y	Y	N	*(1)	—
DBCR2	Y	Y	Y	N	*(1)	—
DBSR	Y	Y	Y	N(2)	*(1)	—
IAC1	Y	Y	Y	N	*(1)	—
IAC2	Y	Y	Y	N	*(1)	—
IAC3	Y	Y	Y	N	*(1)	—
IAC4	Y	Y	Y	N	*(1)	—
JTAG ID	N	N	—	N	—	Read-only
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)(2)	Y	N	Y	Y	Y	CPU must be in debug mode with clocks running
Cache Debug Access Data (CDADATA)(2)	Y	N	Y	Y	Y	CPU must be in debug mode with clocks running
Nexus2/3-Access	Y	N	N	N	N	—
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. Not present on e200z0 or e200z0h

49.5.7 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 e200z0h Debug Mode is entered assuming the OnCE circuitry has been enabled. e200z0h OnCE operation is enabled by the assertion of the *jd_en_once* input (see *Section 49.5.4.1: OnCE enable (jd_en_once)*).

49.5.7.1 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 e200z0h 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 e200z0h core does not execute an instruction before entering Debug Mode, although the first instruction to be executed may be fetched prior to entering Debug Mode.

In this case, all values in the debug scan chain are undefined, and the external Debug Control Module is responsible for proper initialization of the chain before debug mode is exited. In particular, the exception processing associated with reset, may not be performed when the debug mode is exited, thus, the Debug controller must initialize the PC, MSR, and IR to the image that the processor would have obtained in performing reset exception processing, or must cause the appropriate reset to be re-asserted.

49.5.7.2 Debug request during RESET

Asserting a debug request by setting the DR bit in the OCR during the assertion of *p_reset_b* causes the chip to enter debug mode. In this case the chip may fetch the first instruction of the reset exception handler, but does not execute an instruction before entering debug mode. In this case, all values in the debug scan chain are undefined, and the external Debug Control Module is responsible for proper initialization of the chain before debug mode is exited. In particular, the exception processing associated with reset may not be performed when the debug mode is exited, thus, the Debug controller must initialize the PC, MSR, and IR to the image that the processor would have obtained in performing reset exception processing, or must cause the appropriate reset to be re-asserted.

49.5.7.3 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 are aborted by the interrupt processing.

49.5.7.4 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 negates the *p_waiting, p_halted* and *p_stopped* outputs. Once the debug session has ended, the CPU returns 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 is 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 remains 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.

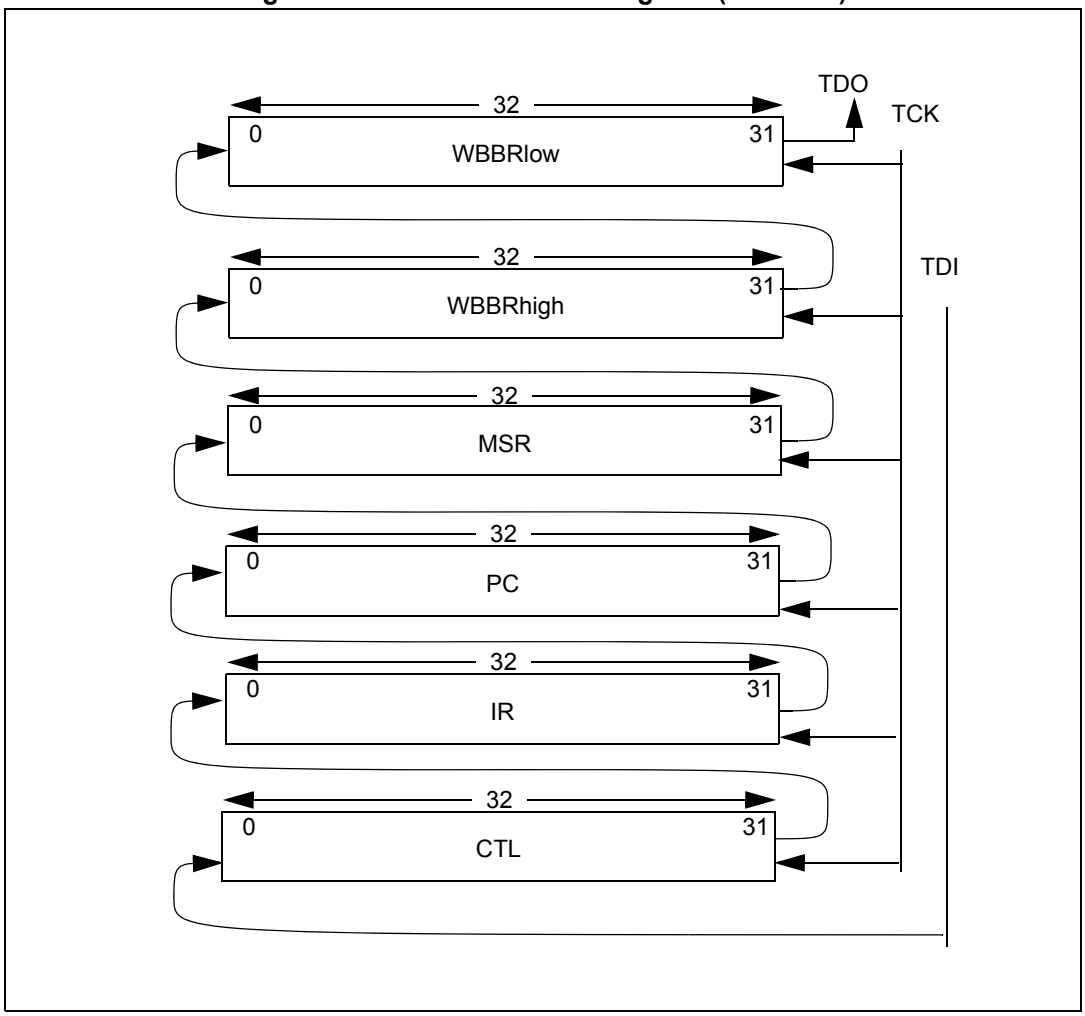
49.5.7.5 Software request during normal activity

Upon executing a "*bkpt*" pseudo-instruction (for e200z0h, 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 DBCR0[EDM] = 1, the CPU enters the debug mode after the instruction following the "*bkpt*" pseudo-instruction has entered the instruction register.

49.5.8 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 e200z0h 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 673* 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 673. CPU Scan Chain Register (CPUSCR)



49.5.8.1 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 controls 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.

49.5.8.2 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.

Access: R/W

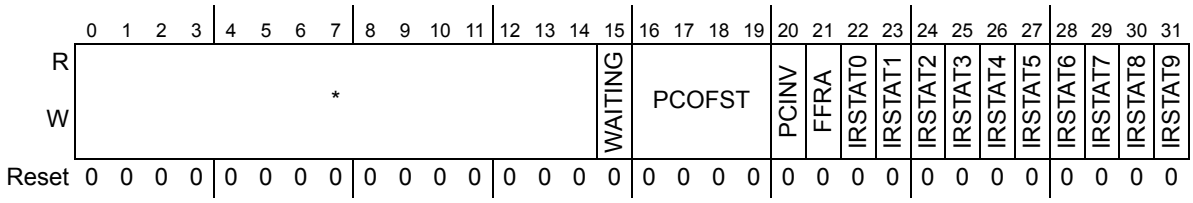


Figure 674. Control State Register (CTL)

Table 692. CTL field descriptions

Field	Description
0:14 31:17 0:15 *	Internal State Bits These control bits represent the internal processor state and should be restored to their original value after a debug session is completed, that is when an e200z0h OnCE command is issued with the GO and EX bits set and not ignored. When performing instruction execution during a debug session (see *Section 49.5.4.3: e200z0h OnCE debug output (jd_debug_b)*) which is not part of the normal program execution flow, these bits should be set to a 0.
15 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 determines 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
16:19 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

Table 692. CTL field descriptions(Continued)

Field	Description
20 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 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.
21 FFRA	Feed Forward RA Operand Bit This control bit causes the content of the $WBBR_{low}$ to be used as the RA (RS for logical and shift operations or RX for VLE se_ instructions) operand value 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 a ori Rx,Rx,0 instruction to the desired register. 0 No action. 1 Content of $WBBR_{low}$ used as RA (RS for logical and shift operations) operand value
22 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.
23 IRStat1	IR Status Bit 1 This control bit indicates a TLB Miss status for the IR. (Note: this bit is reserved.) 0 No TLB Miss occurred on the fetch of this instruction. 1 TLB Miss occurred on the fetch of this instruction.
24 IRStat2	IR Status Bit 2 This control bit indicates an Instruction Address Compare 1 event status for the IR. 0 No Instruction Address Compare 1 event occurred on the fetch of this instruction. 1 An Instruction Address Compare 1 event occurred on the fetch of this instruction.
25 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.
26 IRStat4	IR Status Bit 4 This control bit indicates an Instruction Address Compare 3 event status for the IR. 0 No Instruction Address Compare 3 event occurred on the fetch of this instruction. 1 An Instruction Address Compare 3 event occurred on the fetch of this instruction.
27 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.
28 IRStat6	IR Status Bit 6 This control bit indicates a Parity Error status for the IR. (Note: this bit is reserved.) 0 No Parity Error occurred on the fetch of this instruction. 1 Parity Error occurred on the fetch of this instruction.

Table 692. CTL field descriptions(Continued)

Field	Description
29 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 Precise External Termination Error occurred on the fetch of this instruction.
30 IRStat8	IR Status Bit 8 This control bit indicates the Power Architecture VLE status for the IR. (Note: this bit is always set on e200z0 and e200z0h.) 0 IR contains a Book E instruction. 1 IR contains a Power Architecture VLE instruction, aligned in the Most Significant Portion of IR if 16-bit.
31 IRStat9	IR Status Bit 9 This control bit indicates the Power Architecture VLE Byte-ordering Error status for the IR, or a Book E misaligned instruction fetch, depending on the state of IRStat8. (Note: this bit is reserved on e200z0 and e200z0h.) 0 IR contains an instruction without a byte-ordering error and no Misaligned Instruction Fetch Exception has occurred (no MIF). 1 If IRStat8 = '0', A Book E Misaligned Instruction Fetch Exception has occurred while filling the IR. If IRStat8 = '1', IR contains an instruction with a byte-ordering error due to mismatched VLE page attributes, or due to E indicating little-endian for a VLE page.

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 is 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 generally points to the last completed instruction, although that instruction is not re-executed. The **nop** instruction is executed instead, and instruction fetch and execution resumes at location PC+4. IRStat8 is used to determine the type of instruction present in the IR, thus should be cleared in this case.

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 IRStat2-5 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 is re-entered prior to any further instruction execution.

49.5.8.3 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** is executed, and instruction fetch and execution resumes at PC+4.

49.5.8.4 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 forces 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 lower 32 bits of processor register **r1**, an **e_ori r1,r1,0** instruction is executed, and the result value of the instruction is 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 **e_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 **e_ori** instruction, thus allowing updates to processor registers to be performed (refer to *Section 49.5.8.2: Control State Register (CTL),* 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.

49.5.8.5 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.

49.5.9 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.

49.6 Watchpoint support

e200z0h supports the generation and signalling of watchpoints when operating in Internal Debug mode (DBCR0[IDM] = 1) or in External Debug mode (DBCR0[EDM] = 1).

Watchpoints are indicated with a dedicated set of interface signals. The *jd_watchpoint*[0:5] output signals are used to indicate that a watchpoint has occurred.

Each debug address compare function (IAC1–4, DAC1–2) is 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 (except during a debug session). If not desired, 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 purposes. Watchpoints are signaled in the clock cycle following the occurrence of the actual event. The Nexus 2+ module also monitors assertion of these signals for various development control purposes.

Table 693. 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(1)	Data Address Compare 1 watchpoint Asserted whenever a DAC1 compare occurs regardless of being enabled to set DBSR status
jd_watchpt[5]	DAC2(1)	Data Address Compare 2 watchpoint Asserted whenever a DAC2 compare occurs regardless of being enabled to set DBSR status

1. If the corresponding event is completely disabled in DBCR0, either load-type or store-type data accesses are allowed to generate watchpoints, otherwise watchpoints are generated only for the enabled conditions.

49.7 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:

1. The debugger should ensure that the *jd_en_once* control signal is asserted in order to enable OnCE operation
2. 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 places the CPU in a debug state in which it is halted and awaiting single-step commands or a release to normal mode
3. 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 is shifted out during the Shift_DR state. The debugger should save the scanned-out value of CPUSCR for later restoration.
4. Select the DBCR0 register and update it with the DBCR0[EDM] bit set.
5. Clear the DBSR status bits.
6. Write appropriate values to the DBCRx, IAC, DAC registers. Note that the initial write to DBCR0 only affects 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, set the OCR[I_DMDIS] and/or OCR[I_DMDIS] control bits to ensure that no TLB misses occur while performing the debug operations
- 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 ensures 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 49.5.8.2: Control State Register (CTL)*), 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[DMDIS], OCR[DR], control bits 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 49.5.8.2: Control State Register (CTL)*), 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 value.
- The debugger should write the DBCRx registers as needed, likely clearing every enable except the DBCR0[EDM] bit
- The debugger should write the DBSR to a cleared state.
- The debugger should rewrite 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 49.5.8.2: Control State Register (CTL)*), 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.

50 Debug and Trace

50.1 Introduction

50.1.1 Features

SPC570Sx includes comprehensive features to assist in the software development process:

- Supports Nexus class 3 debug and trace
- Supports memory reconstruction[v]
- 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 trace captured using the CPU's own Nexus 2+ module
- Device data 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
- 1149.1 and 1149.7 controllers
- Debug break and cross-triggering control
- Synchronous restart control for CPUs when exiting debug mode
- Tool hot plug capability
- Security access control
- Debug reset control

50.1.2 Architecture

The top level debug and trace architecture is shown in *Figure 675*. It comprises the following modules:

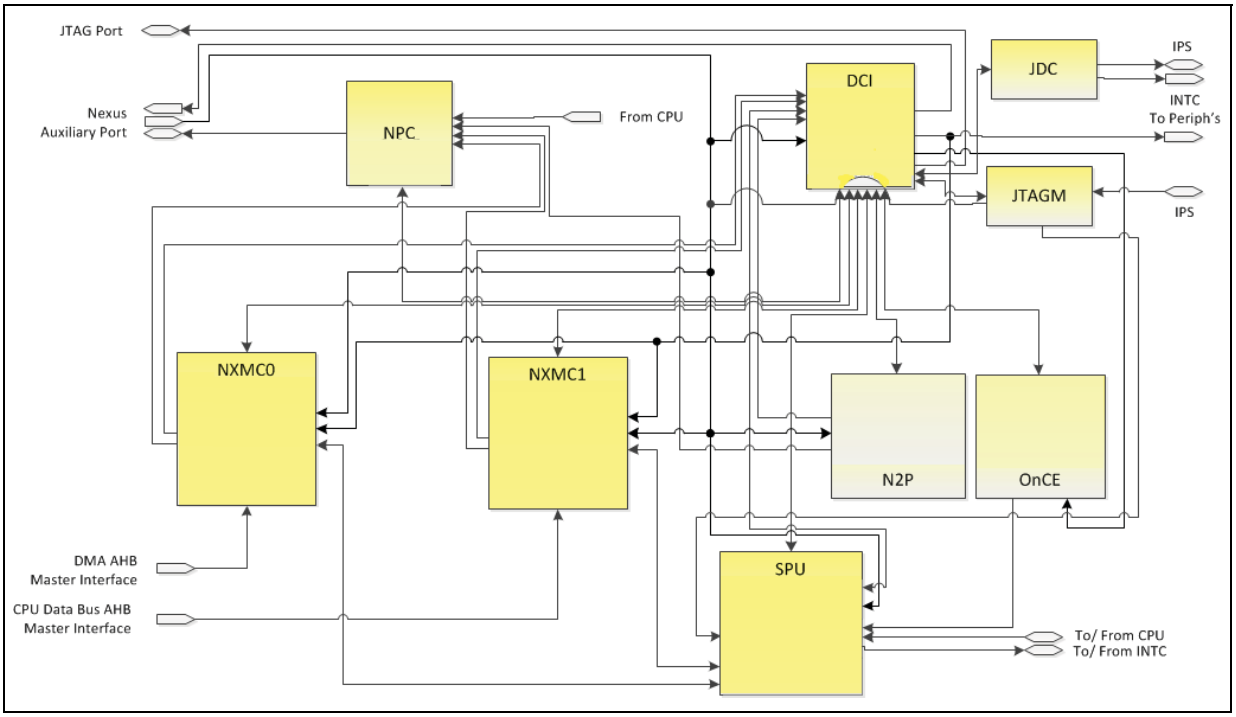
- Debug Control Interface (DCI)
- JTAG Master (JTAGM)
- 2 Nexus Crossbar Multimaster Trace Clients (NXMC)
- Nexus Port Controller (NPC)
- Sequence Processing Unit (SPU)
- JTAG Data Communication Module (JDC)

The CPUs themselves provide the Nexus 2+ module and the OnCE module. *Figure 675* 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 request coming from different Nexus clients. The Nexus scheme includes the possibility to trace the CPU data and instructions and the DMA accesses. Because the CPU has natively the instruction tracing

v. For the instrumentation use case, external reconstruction of the internal RAM content is required.

capability only, an external Nexus client (NXMC1) has been added on the CPU AHB data bus. The other external Nexus client (NXMC0) is used to trace the DMA accesses.

Figure 675. Debug architecture



50.1.3 Overview

The DCI can take the following different inputs for its JTAG signals:

- P.JTAG: JTAG signals coming from the device JTAG pads. This is the standard 4-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.

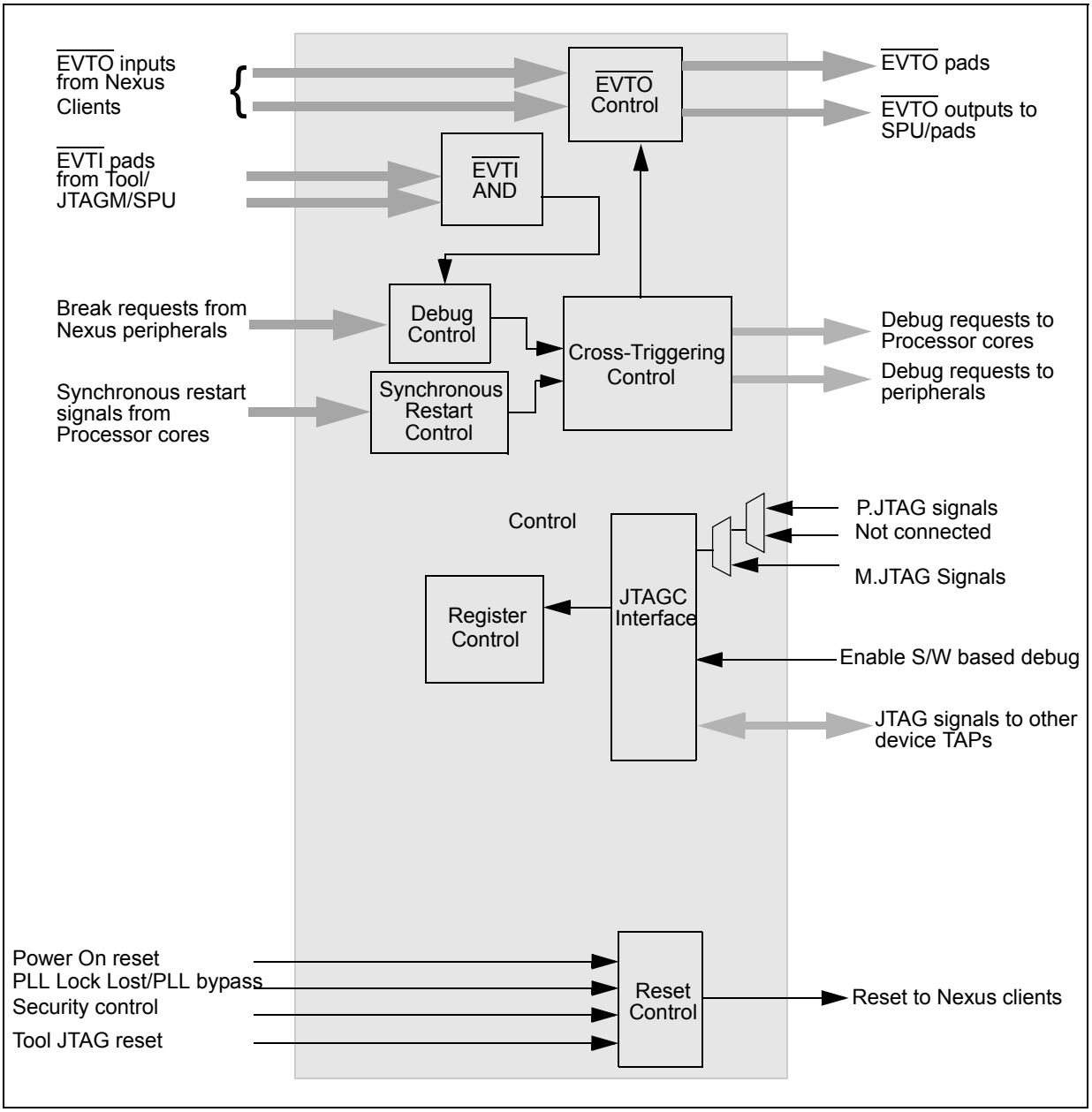
By default the DCI operates in standard 4-pin JTAG mode (IEEE 1149.1). It can be configured in 2-pin reduced pin JTAG mode (IEEE 1149.7) after starting from standard 4-pin JTAG mode.

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 the SPU. The DCI also provides the Event Out (EVTO) signals to the debug tool in case of the occurrence of any internal debug event.

Figure 676 shows the DCI block diagram.

Figure 676. DCI block diagram



50.2 Register description

The 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.

50.2.1 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.

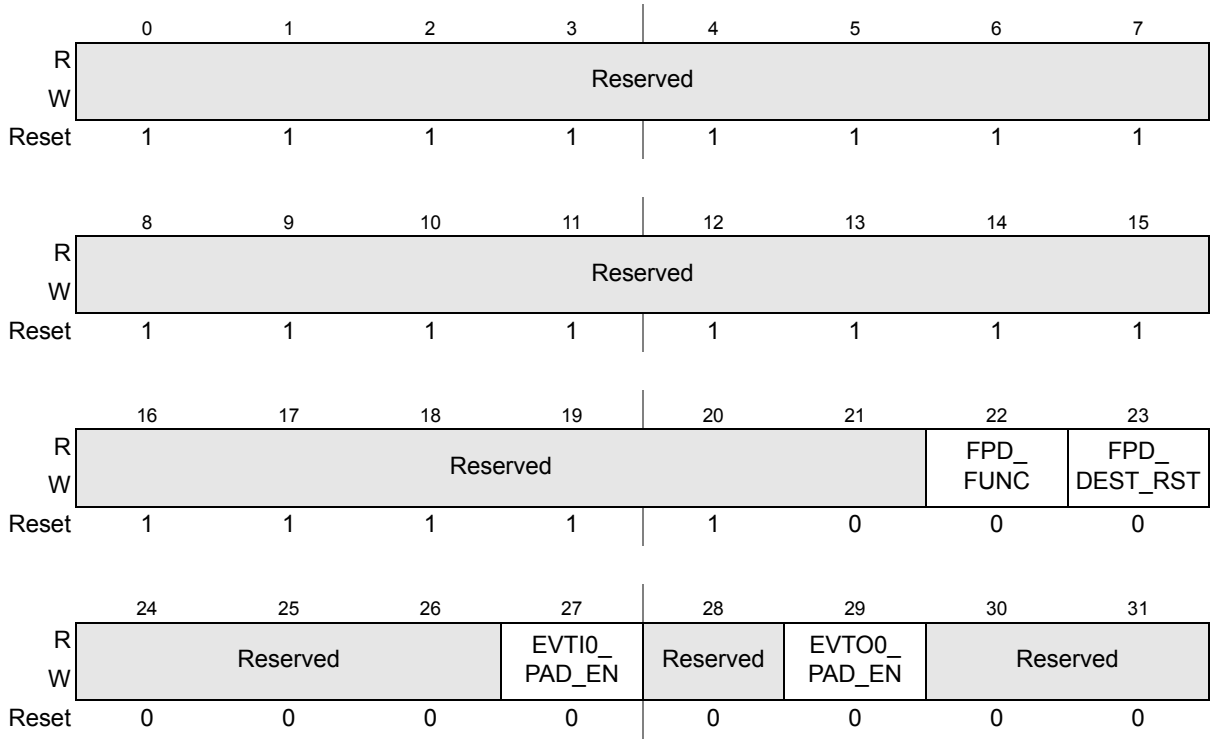


Figure 677. DCI control register (DCI_CR)

Table 694. DCI_CR field descriptions

Field	Description
22 FPD_FUNC	Force Functional reset 0 Do not force functional reset 1 Force functional reset
23 FPD_DEST_RST	Force Destructive reset 0 Do not force destructive reset 1 Force destructive reset
27 EVTI0_PAD_EN	$\overline{\text{EVTI}}$[0] feature enable 0 $\overline{\text{EVTI}}$[0] disabled(default) 1 $\overline{\text{EVTI}}$[0] enabled
29 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

50.2.2 DCI pin multiplexing control register (DCI_PINCR)

The DCI_PINCR is a 32-bit data register. This register is programmed using JTAG shift with the corresponding enable instruction described in the JTAGC chapter.

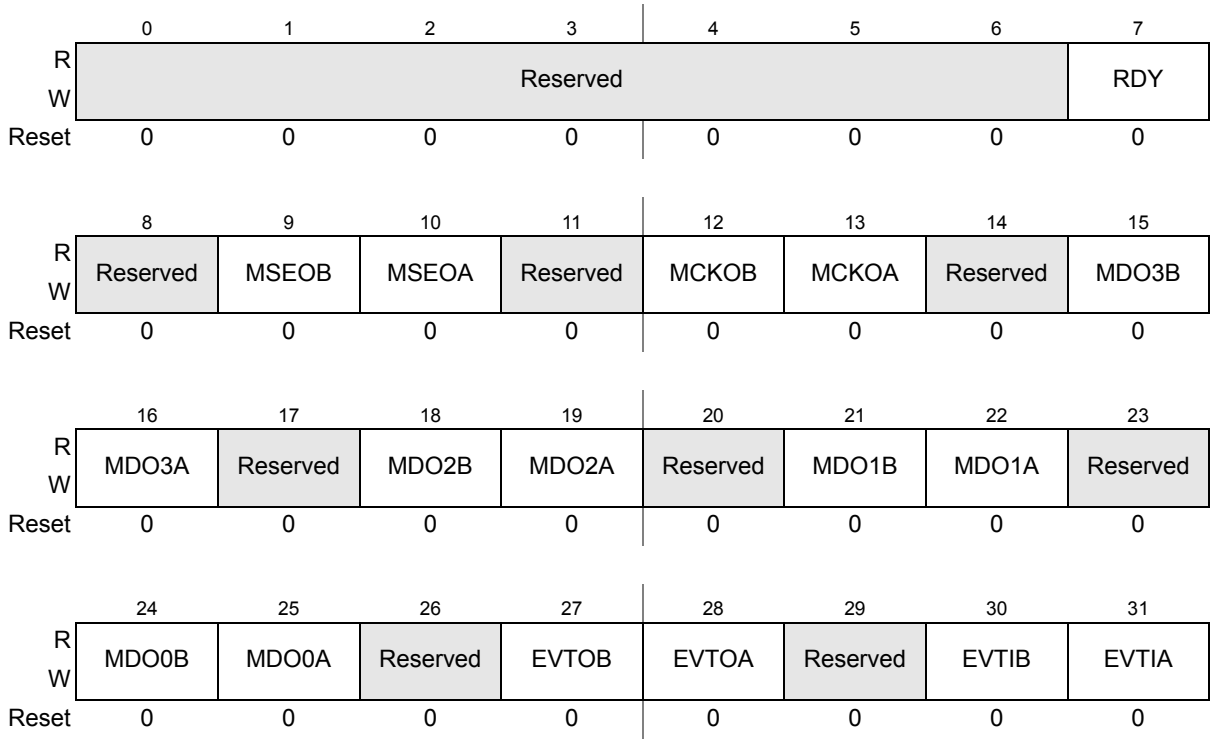


Figure 678. DCI pin multiplexing control register (DCI_PINCR)

Table 695. DCI_PINCR field descriptions

Field	Description
7 RDY	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
9 MSEOB	MSEO function on port pin PE[3] (pad 67) 0 Disables MSEO function on port pin PE[3] (pad 67) to Nexus 1 Enables MSEO function on port pin PE[3] (pad 67) to Nexus
10 MSEOA	MSEO function on port pin PB[1] (pad 17) 0 Disables MSEO function on port pin PB[1] (pad 17) to Nexus 1 Enables MSEO function on port pin PB[1] (pad 17) to Nexus
12 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
13 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
15 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
16 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 695. DCI_PINCR field descriptions(Continued)

Field	Description
18 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
19 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
21 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
22 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
24 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
25 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
27 EVTOB	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
28 EVTOA	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
30 EVTIB	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
31 EVTIA	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

50.2.3 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 696*.

Table 696. Nexus trace source identifiers

Nexus Trace Source	Source Trace ID	
e200_CORE_0	0b0000	0x0

Table 696. Nexus trace source identifiers

Nexus Trace Source	Source Trace ID	
AUX_BUS_MON_0	0b1100	0xC
AUX_BUS_MON_1	0b1101	0xD

50.2.4 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.

50.2.4.1 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.

The L1SEL0 register fields are described in *Table 697*.

Table 697. L1SEL0 register field descriptions

Field	Description
MUX 0	000 CPU0 instruction address compare 1 (watchpoint 0) 001 Reserved 010 Reserved 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 Reserved 011 Reserved 100 Reserved 101 Reserved 110 Reserved 111 Reserved

Table 697. L1SEL0 register field descriptions(Continued)

Field	Description
MUX 2	000 CPU0 instruction address compare 3 (watchpoint 2) 001 Reserved 010 Reserved 011 Reserved 100 Reserved 101 Reserved 110 Reserved 111 Reserved
MUX 3	000 CPU0 instruction address compare 4 (watchpoint 3) 001 Reserved 010 Reserved 011 Reserved 100 Reserved 101 Reserved 110 Reserved 111 Reserved
MUX 4	000 Reserved 001 Reserved 010 Reserved 011 Reserved 100 Reserved 101 Reserved 110 Reserved 111 Reserved
MUX 5	000 Reserved 001 Reserved 010 Reserved 011 Reserved 100 Reserved 101 Reserved 110 Reserved 111 Reserved
MUX 6	000 CPU0 instruction address compare 7 (watchpoint 14) 001 Reserved 010 Reserved 011 Reserved 100 Reserved 101 Reserved 110 Reserved 111 Reserved
MUX 7	000 CPU0 instruction address compare 8 (watchpoint 15) 001 Reserved 010 Reserved 011 Reserved 100 Reserved 101 Reserved 110 Reserved 111 Reserved

The L1SEL1 register fields are described in *Table 698*.

Table 698. L1SEL1 register field descriptions

Field	Description
MUX 8	000 CPU0 instruction address compare 1 (watchpoint 0) 001 Reserved 010 Reserved 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 Reserved 011 Reserved 100 Reserved 101 Reserved 110 Reserved 111 Reserved
MUX 10	000 CPU0 instruction address compare 3 (watchpoint 2) 001 Reserved 010 Reserved 011 Reserved 100 Reserved 101 Reserved 110 Reserved 111 Reserved
MUX 11	000 CPU0 instruction address compare 4 (watchpoint 3) 001 Reserved 010 Reserved 011 Reserved 100 Reserved 101 Reserved 110 Reserved 111 Reserved
MUX 12	000 Reserved 001 Reserved 010 Reserved 011 Reserved 100 Reserved 101 Reserved 110 CPU0 debug event output 1 (watchpoint 13) 111 Reserved
MUX 13	000 Reserved 001 Reserved 010 Reserved 011 Reserved 100 Reserved 101 Reserved 110 Reserved 111 Reserved

Table 698. L1SEL1 register field descriptions(Continued)

Field	Description
MUX 14	000 CPU0 instruction address compare 7 (watchpoint 14) 001 Reserved 010 Reserved 011 Reserved 100 Reserved 101 Reserved 110 Reserved 111 Reserved
MUX 15	000 CPU0 instruction address compare 8 (watchpoint 15) 001 Reserved 010 Reserved 011 Reserved 100 Reserved 101 Reserved 110 Reserved 111 Reserved

The L1SEL2 register fields are described in *Table 699*.

Table 699. L1SEL2 register field descriptions

Field	Description
MUX16	000 CPU0 data address compare 1 (watchpoint 4) 001 Reserved 010 Reserved 011 Reserved 100 Reserved 101 Reserved 110 Reserved 111 Reserved
MUX 17	000 CPU0 data address compare 2 (watchpoint 5) 001 Reserved 010 Reserved 011 Reserved 100 Reserved 101 Reserved 110 Reserved 111 Reserved
MUX 18	000 Reserved 001 Reserved 010 Reserved 011 Reserved 100 Reserved 101 Reserved 110 Reserved 111 Reserved

Table 699. L1SEL2 register field descriptions(Continued)

Field	Description
MUX 19	000 Reserved 001 Reserved 010 Reserved 011 Reserved 100 Reserved 101 Reserved 110 Reserved 111 Reserved
MUX 20	000 CPU0 data address compare 1 (watchpoint 4) 001 Reserved 010 Reserved 011 Reserved 100 Reserved 101 Reserved 110 Reserved 111 Reserved
MUX 21	000 CPU0 data address compare 2 (watchpoint 5) 001 Reserved 010 Reserved 011 Reserved 100 Reserved 101 Reserved 110 Reserved 111 Reserved
MUX 22	000 Reserved 001 Reserved 010 Reserved 011 Reserved 100 Reserved 101 Reserved 110 Reserved 111 Reserved
MUX 23	000 Reserved 001 Reserved 010 Reserved 011 Reserved 100 Reserved 101Reserved 110 Reserved 111 Reserved

The L1SEL3 register fields are described in *Table 700*.

Table 700. L1SEL3 register field descriptions

Field	Description
MUX 24	000 CPU0 data address compare 1 (watchpoint 4) 001 Reserved 010 Reserved 011 Reserved 100 Reserved 101 Reserved 110 Reserved 111 Reserved
MUX 25	000 CPU0 data address compare 2 (watchpoint 5) 001 Reserved 010 Reserved 011 Reserved 100 Reserved 101 Reserved 110 Reserved 111 Reserved
MUX 26	000 Reserved 001 Reserved 010 Reserved 011 Reserved 100 Reserved 101 Reserved 110 Reserved 111 Reserved
MUX 27	000 Reserved 001 Reserved 010 Reserved 011 Reserved 100 Reserved 101 Reserved 110 Reserved 111 Reserved
MUX 28	000 CPU0 data address compare 1 (watchpoint 4) 001 Reserved 010 Reserved 011 Reserved 100 Reserved 101 Reserved 110 Reserved 111 Reserved
MUX 29	000 CPU0 data address compare 2 (watchpoint 5) 001 Reserved 010 Reserved 011 Reserved 100 Reserved 101 Reserved 110 Reserved 111 Reserved

Table 700. L1SEL3 register field descriptions(Continued)

Field	Description
MUX 30	000 Reserved 001 Reserved 010 Reserved 011 Reserved 100 Reserved 101 Reserved 110 Reserved 111 Reserved
MUX 31	000 Reserved 001 Reserved 010 Reserved 011 Reserved 100 Reserved 101Reserved 110 Reserved 111 Reserved

The L1SEL4 register fields are described in *Table 701*.

Table 701. L1SEL4 register field descriptions

Field	Description
MUX 32	000 Reserved 001 Reserved 010 Reserved 011 Reserved 100 Reserved 101 Reserved 110 Reserved 111 Reserved
MUX 33	000 Reserved 001 Reserved 010 Reserved 011 Reserved 100 Reserved 101 Reserved 110 Reservedl 111 Reserved
MUX 34	000 CPU0 performance monitor counter 2 (watchpoint 25) 001Reserved 010 Reserved 011 Reserved 100 Reserved 101 Reserved 110 Reserved 111 Reserved

Table 701. L1SEL4 register field descriptions(Continued)

Field	Description
MUX 35	000 CPU0 performance monitor counter 3 (watchpoint 26) 001 Reserved 010 Reserved 011 Reserved 100 Reserved 101 Reserved 110 Reserved 111 Reserved
MUX 36	000 Reserved) 001 Reserved 010 CPU0 performance monitor event input (watchpoint 22) 011 Reserved 100 Reserved 101 Reserved 110 Reserved 111 Reserved
MUX 37	000 Reserved 001 Reserved 010 Reserved 011 Reserved 100 Reserved 101 Reserved 110 Reserved 111 Reserved
MUX 38	000 Reserved 001 Reserved 010 Reserved 011 Reserved 100 Reserved 101 Reserved 110 Reserved 111 Reserved
MUX 39	000 Reserved 001 Reserved 010 Reserved 011 Reserved 100 Reserved 101 Reserved 110 Reserved 111 Reserved

The L1SEL5 register fields are described in *Table 702*.

Table 702. L1SEL5 register field descriptions

Field	Description
MUX 40	000 CPU0 interrupt (watchpoint 16) 001 CPU0 critical interrupt (watchpoint 18) 010 Reserved 011 Reserved 100 Reserved 101 Reserved 110 SPU counter event 0 111 Reserved
MUX 41	000 Reserved 001 Reserved 010 Reserved 011 Reserved 100 Reserved 101 Reserved 110 SPU counter event 1 111 Reserved
MUX 42	000 Reserved 001 Reserved 010 Reserved 011 Reserved 100 Reserved 101 Reserved 110 SPU counter event 2 111 Reserved
MUX43	000 Reserved 001 Reserved 010 Reserved 011 Reserved 100 Reserved 101 Reserved 110 SPU counter event 3 111 Reserved
MUX 44	000 CPU0 return (watchpoint 17) 001 CPU0 critical return (watchpoint 19) 010 Reserved 011 Reserved 100 Reserved 101 Reserved 110 SPU counter event 4 111 Reserved
MUX 45	000 Reserved 001 Reserved 010 Reserved 011 Reserved 100 Reserved 101 Reserved 110 SPU counter event 5 111 Reserved

Table 702. L1SEL5 register field descriptions(Continued)

Field	Description
MUX 46	000 Reserved 001 Reserved 010 Reserved 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

The L1SEL6 register fields are described in *Table 703*.

Table 703. L1SEL6 register field descriptions

Field	Description
MUX 48	000 Reserved 001 Reserved 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 Reserved 001 Reserved 010 Reserved 011 NXMC0 watchpoint 2 100 Reserved 101 Reserved 110 SPU counter event 10 111 Reserved

Table 703. L1SEL6 register field descriptions(Continued)

Field	Description
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 Reserved 010 Reserved 011 Reserved 100 Reserved 101 Reserved 110 SPU counter event 12 111 Reserved
MUX 53	000 Reserved 001 Reserved 010 Reserved 011 Reserved 100 Reserved 101 Reserved 110 SPU counter event 13 111 Reserved
MUX 54	000 Reserved 001 Reserved 010 Reserved 011 Reserved 100 Reserved 101 Reserved 110 SPU counter event 14 111 Reserved
MUX 55	000 Reserved 001 Reserved 010 Reserved 011 Reserved 100 Reserved 101 Reserved 110 SPU counter event 15 111 Reserved

The L1SEL7 register fields are described in *Table 704*.

Table 704. L1SEL7 register field descriptions

Field	Description
MUX 56	000 Reserved 001 Reserved 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 Reserved 001 Reserved 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 Reserved 010 Reserved 011 Reserved 100 Reserved 101 Reserved 110 Reserved 111 DCI system halt
MUX 61	000 Reserved 001 Reserved 010 Reserved 011 Reserved 100 Reserved 101 Reserved 110 Reserved 111 Reserved

Table 704. L1SEL7 register field descriptions(Continued)

Field	Description
MUX 62	000 Reserved 001 Reserved 010 Reserved 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 Reserved

50.3 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.

50.3.1 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.

50.3.1.1 JTAG port

The SPC570Sx features a JTAG port with a 4-pin interface. The supported signals are TDI, TDO, TCK and TMS. TRST is not used.

Table 705. 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

50.3.1.2 Nexus auxiliary port

Table 706. Nexus signals

Channel	Functionality	Direction	Width
EVTI	Event Trigger Input	Input	1
EVTO	Event Trigger Output	Output	1
MSEO	MDO Start/End Output	Output	1
MCKO	MDO and MSEO signal clock	Output	1
MDO(0..N(1))	Trace	Output	N(1)

1. Four MDO pins configuration for QFP64/QFP100; 12 MDO pins configuration for QFP144

*Note: The **nex_rdy_b** signal is not used. In order to accomplish a read/write operation via Nexus RWCS register, the external tool polls the RWCS.DV and RWCS.ERR bits.*

50.3.2 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 (NEXUS2/3-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 (see *Section 50.2.2*) to select which pads are to be used as Nexus pads

50.4 Functionality

50.4.1 Debug Control Interface (DCI)

The DCI is the control interface for the on-chip debug and trace resources. It includes a 1149.1/1149.7 compatible JTAG interface which is used to connect with an external JTAG tool, but also features support for SW debug control by the MCU without the need for an external tool.

The supported functionality is:

- Debug mode enable control for connected tool or software
- Debug break and cross-triggering control
- Debug reset control
- Synchronous restart control
- Security access control

50.4.1.1 DCI JTAG inputs

The DCI takes the following inputs for its JTAG signals:

- JTAG signals coming from the device JTAG pads
- JTAG signals generated by the device JTAGM module in SW debug mode

50.4.1.2 Break control

The DCI provides central break control for CPU and non-CPU debug modules. Break signals are generated by:

- SPU
- JTAGM
- Nexus clients

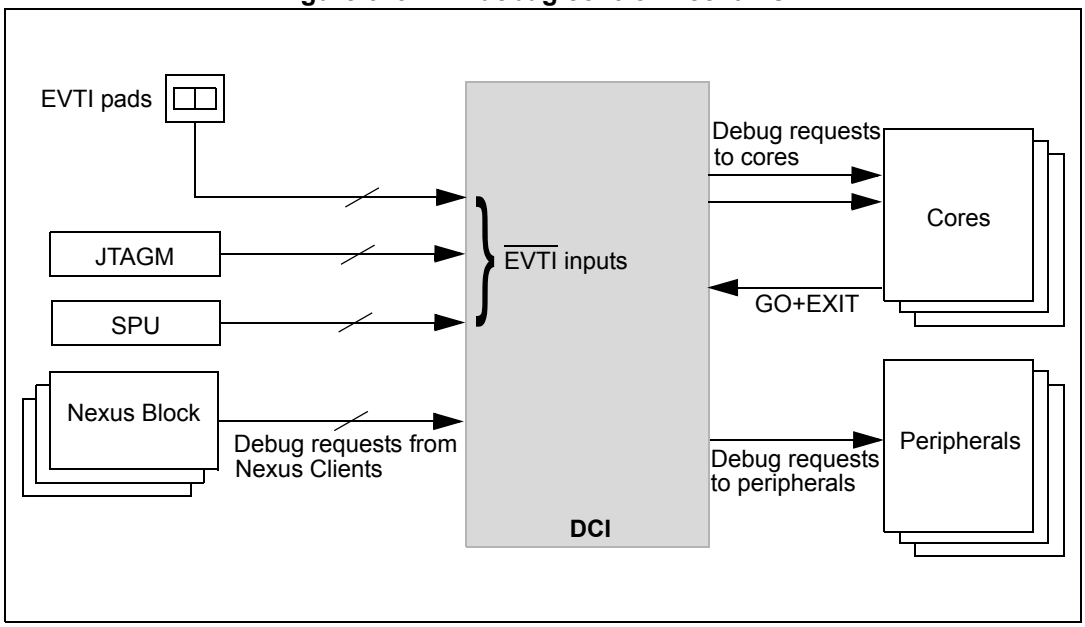
There are various different input sources that provide debug requests:

- $\overline{\text{EVTI}}$[1:0] signals from the Sequence Processing Unit (SPU)
- $\overline{\text{EVTI}}$[1:0] signals from JTAGM
- $\overline{\text{EVTI}}$[0] signal from EVTI pads

The DCI generates the 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 679. DCI debug control mechanism



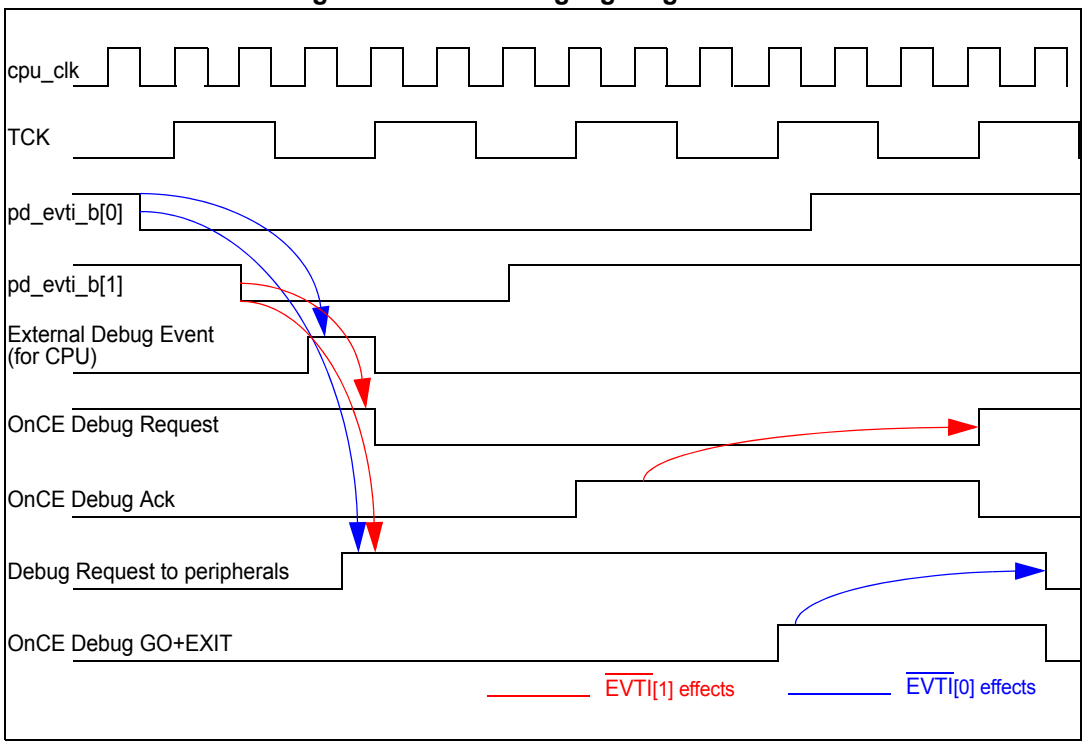
The various sources of $\overline{\text{EVTI}}$[1:0] inputs are synchronized and logically ANDed in the DCI to generate two debug requests DCI_$\overline{\text{EVTI}}$[1:0] for the processor cores and non-CPU peripherals. Of these two inputs, $\overline{\text{EVTI}}$[0] is used to generate debug requests for non-CPU peripherals and for processor cores through the external debug request signal. The other input, $\overline{\text{EVTI}}$[1], 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 pads is enabled for debug control using the 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 680 shows the timing sequence.

Figure 680. DCI debug signal generation



50.4.1.3 EVTO management

The DCI provides EVTO management. The DCI collects and samples EVTO signals coming from the following sources:

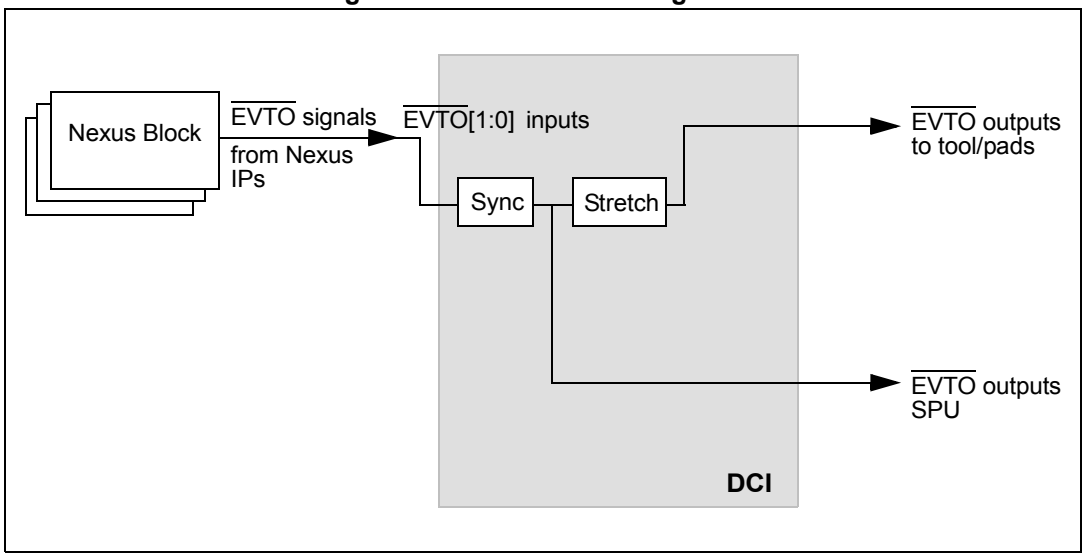
- CPU
- SPU
- NXMC clients

After receiving any EVTO signal, the DCI asserts EVTO outputs going to:

- SPU
- EVTO pads

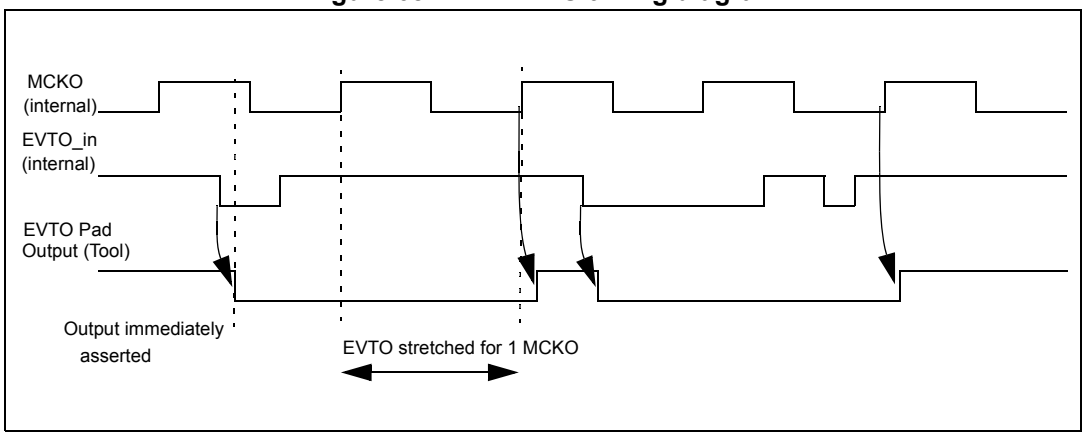
The EVTO management block diagram is presented in *Figure 681*.

Figure 681. DCI EVTO management



The DCI module controls sharing of the EVTO outputs between all Nexus Clients including the cores that produce EVTO[1:0] signals. The DCI uses a high frequency clock to sample EVTO signals coming from all modules which might generate EVTO outputs at their own clock frequency. After receiving a single clock period of asserted EVTO[1:0] signals from any Nexus client, the DCI immediately asserts EVTO[1:0] outputs and stretches it for a minimum of one output clock period for the output to the tool (*Figure 682*). The DCI also generates EVTO outputs in other device clock domains so that other Nexus clients can receive EVTO inputs without the need of any synchronization. EVTO function is active as long as the DCI is not in reset.

Figure 682. DCI EVTO timing diagram



50.4.1.4 Software enabled debug and trace mode

When enabled in the JTAGM module, the DCI is configured to receive JTAG signals which are driven by the JTAGM module. In this case software can take control of the DCI features like any other JTAG source.

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 trace mode via a memory-mapped register. It is done first by setting a control bit in the JTAGM (JTAGM_MCR[DTM]). For more details, refer to *Section 53.4.1.1: Module configuration register (JTAGM_MCR)*. 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 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, which takes all debug modules out of reset.

The bit JTAGM_MCR[jtagm_JCOMP], when programmed to 0, asserts reset for the JTAGC block, thus putting the debug and trace logic in reset state. This bit is used during software based debug to initialize the debug and trace logic.

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.

50.4.1.5 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 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.

50.4.1.6 JTAG Master (JTAGM)

The SPC570Sx 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.

50.4.1.7 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.

- CPU Data Bus NXMC: Since the e200z0hn2p CPU does not support data trace an NXMC block has to be added externally.
- 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.

50.4.1.8 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.

50.4.1.9 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 N2P block of the CPU
- The DMA master port NXMC client
- The CPU data bus 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.

50.4.1.10 Sequence Processing Unit (SPU)

The SPU is able to generate complex debug events, based upon input events from sources throughout the SOC.

On SPC570Sx the required client interfaces are:

- One e200z0h 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

50.5 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 707 shows a summary of the configuration options.

Table 707. DCI JTAG source selection

JTAGM_MCR[DTM]	DCI source
0	P.JTAG (Standard JTAG)
1	M.JTAG (Software)

The reset signal to the DCI block consists of the following:

- Power-on reset, Low/High voltage detection on IO/core supplies

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 708 contains the operating mode descriptions.

Table 708. 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.

50.5.1 Nexus reset control

The DCI provides the reset signal for all Nexus blocks. This reset is controlled using the following signals:

- Power-on reset/Low voltage detect
- SoC Debug_Enable signal
- System PLL is out of lock state and not bypassed

Asserting power-on reset, 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. 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 module (PASS), 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 709. Nexus reset truth table

POR	Debug_Enable	PLL lock	Nexus reset
Assert	X	X	Assert
X	X	X	Assert
X	Disable	X	Assert
X	X	Unlocked	Assert
Negate	Enable	Locked	Negate

50.5.1.1 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.

50.5.1.2 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 or Power-on-reset.

51 JTAG Controller (JTAGC)

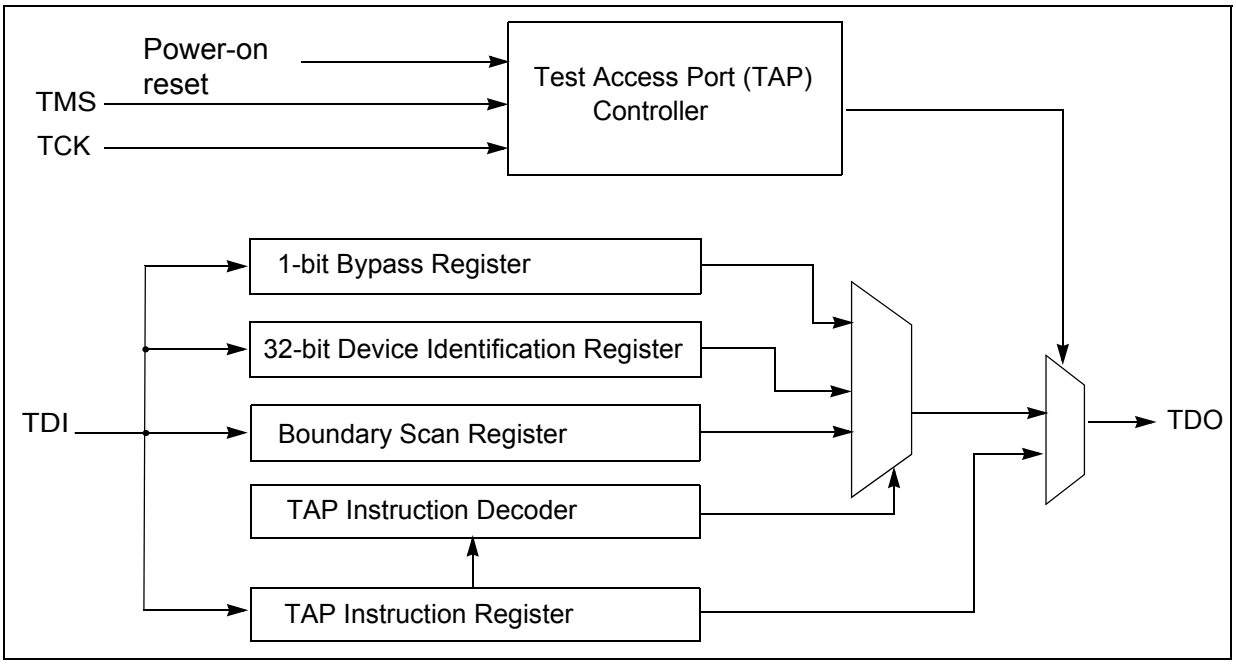
51.1 Introduction

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.

51.1.1 Block Diagram

The following is a block diagram of the JTAG Controller (JTAGC) block.

Figure 683. JTAG (IEEE 1149.1) block diagram



51.1.2 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)
- Instruction register that supports several IEEE 1149.1-2001 defined instructions as well as several public and private device-specific instructions. Refer to *Table 713* for a list of supported instructions.
- Sharing of the TAP with other TAP controllers via ACCESS_AUX_x instructions
- JTAG_PASSWORD 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.

51.1.3 Modes of operation

The JTAGC block uses 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.

51.1.3.1 Reset

The JTAGC block is placed in reset when either power-on reset is asserted, 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 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

51.1.3.2 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 51.4.4*.

51.1.3.3 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.

51.2 External signal description

The JTAGC consists of four 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 710. 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

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.

51.2.1 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.

51.2.2 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.

51.2.3 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 51.4.3*.

51.2.4 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.

51.3 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.

51.3.1 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

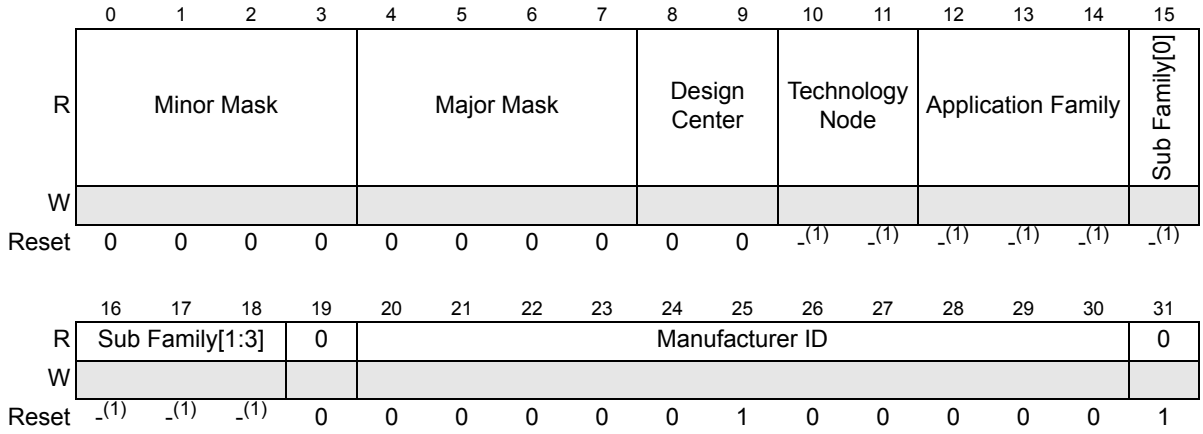
Figure 684. 6-bit instruction register

51.3.2 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.

51.3.3 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 711*

Figure 685. Device identification register

The following table describes the device identification register functions.

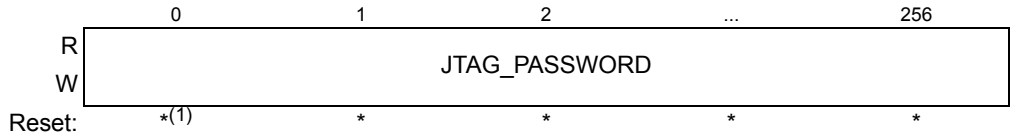
Table 711. Device identification register field descriptions

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 0001 SPC570S50
20:30 Manufacturer ID	Manufacturer Identity Code These bits contain the reduced Joint Electron Device Engineering Council (JEDEC) ID 00000100000: STMicroelectronics

51.3.4 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 686. JTAG_PASSWORD register

The following table describes the JTAG_PASSWORD register functions.

Table 712. JTAG_PASSWORD register field descriptions

Field	Description
JTAG_PASSWORD	JTAG Password. The JTAG_PASSWORD bits are used to provide the JTAG password for security.

51.3.5 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 51.4.5*. The size of the boundary scan register and bit ordering is device-dependent and can be found in the device BSDL file.

51.4 Functional description

This section explains the JTAGC functional description.

51.4.1 JTAGC reset configuration

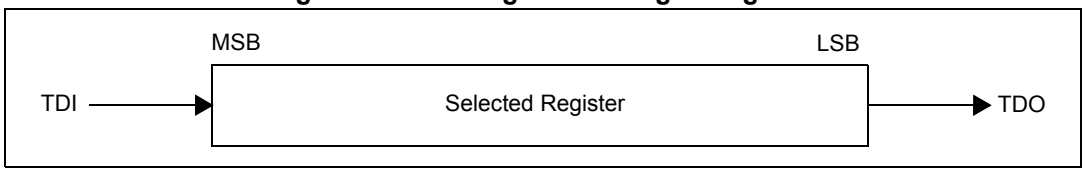
While in reset, the TAP controller is forced into the Test-Logic-Reset state, thus disabling the test logic and allowing normal operation of the on-chip system logic. In addition, the instruction register is loaded with the IDCODE instruction.

51.4.2 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 51.4.4.8: ACCESS_AUX_x instructions*.

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.

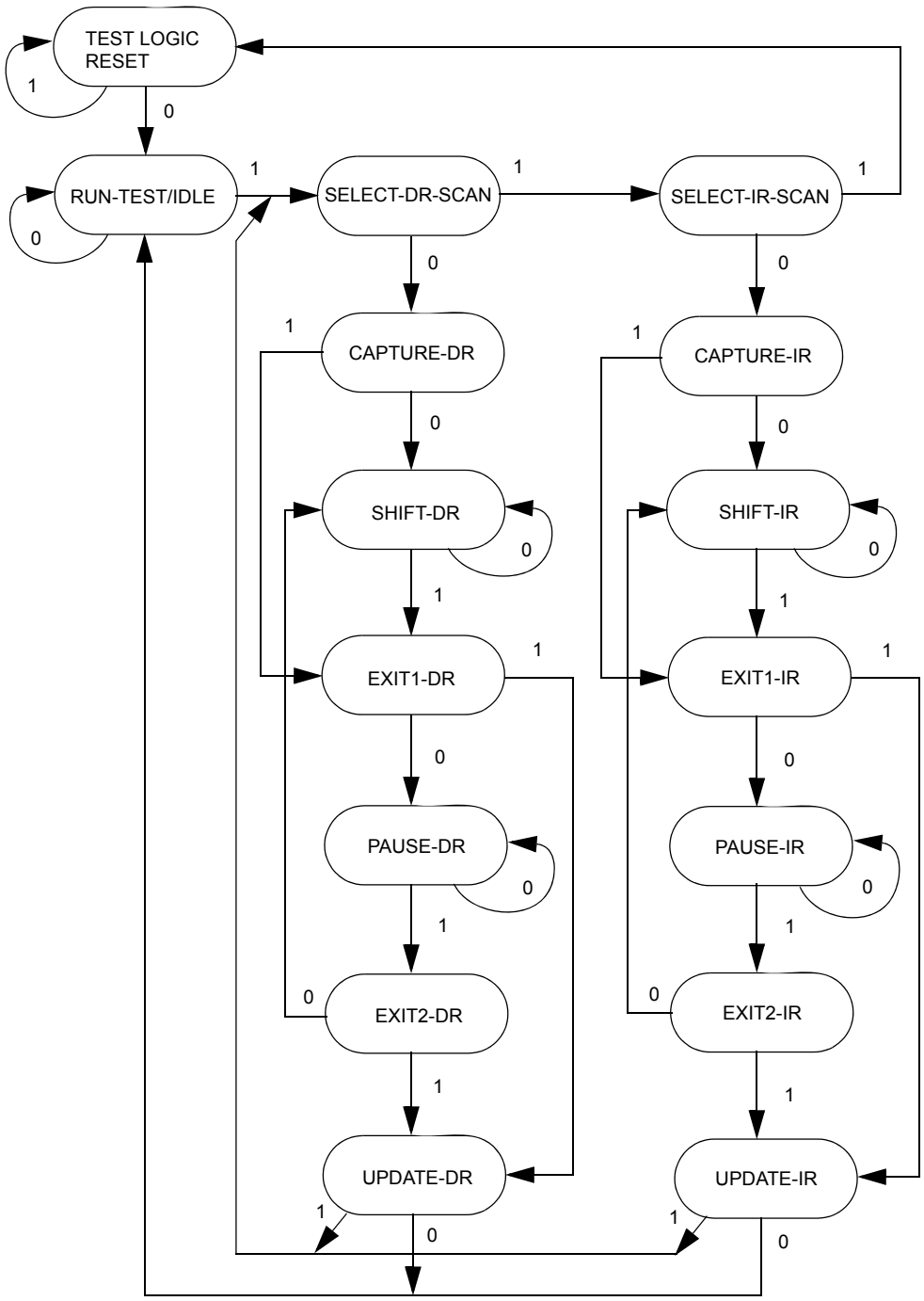
Figure 687. Shifting data through a register



51.4.3 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 688. 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.

51.4.3.1 Enabling the TAP controller

The JTAGC TAP controller is enabled by setting the JTAGC enable to a logic 1 value.

51.4.3.2 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.

51.4.4 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 713. 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
Reserved	000101	Reserved
Reserved	000110	Reserved
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_DCI_PINCR	001111	Enables access to DCI PINCR register
ACCESS_AUX_x	100000-111110	Grants one of the auxiliary TAP controllers ownership of the TAP as shown in the cells below.
Reserved	100000	Reserved
ACCESS_AUX_NPC_PD	100001	Enables access to the NPC TAP controller on production device

Table 713. 6-bit JTAG instructions(Continued)

Instruction	Code[5:0]	Instruction Summary
Reserved	100010	Reserved
Reserved	100011	Reserved
Reserved	100100	Reserved
Reserved	100101	Reserved
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
ACCESS_AUX_CORE_0s	101010	Enables access to E200 Core 0s
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
Reserved	111001	Reserved
ACCESS_AUX_SPU_A	111010	Enables access to SPU Client TAP controller
Reserved	111011	Reserved
ACCESS_AUX_PARALLEL(1)	111100	Enables access to all cores in parallel
Reserved	111101	Reserved
Reserved	111110	Reserved
BYPASS	111111	Selects bypass register for data operations
Reserved(2)	All other opcodes	Decoded to select bypass register

1. 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
2. The manufacturer reserves the right to change the decoding of reserved instruction codes in the future

51.4.4.1 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.

51.4.4.2 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.

51.4.4.3 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.

51.4.4.4 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.

51.4.4.5 ENABLE_JTAG_PASSWORD instruction

The ENABLE_JTAG_PASSWORD instruction selects the JTAG_PASSWORD register for connection as the shift path between TDI and TDO.

51.4.4.6 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 (e.g., high impedance). HIGHZ also asserts the internal system reset for the MCU to force a predictable internal state.

51.4.4.7 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.

51.4.4.8 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.

51.4.4.9 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.

51.4.5 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.

51.5 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, the following sequence is required:

1. Place the JTAGC in reset through TAP controller state machine transitions controlled by TMS
2. Load the appropriate instruction for the test or action to be performed

52 JTAG Data Communication (JDC)

52.1 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.

52.2 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.

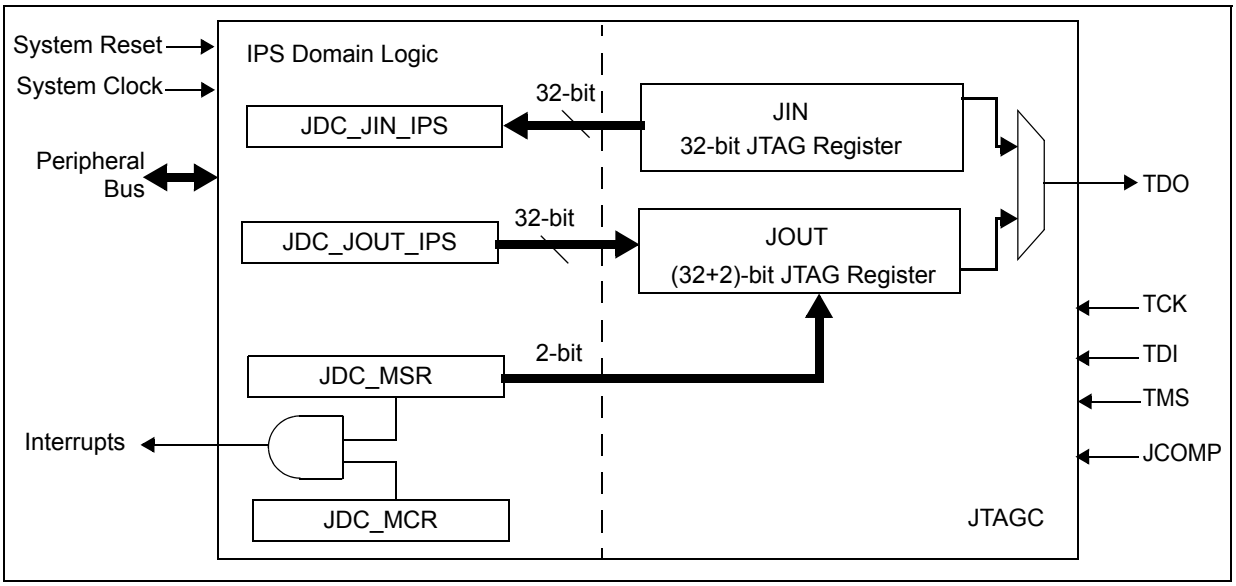
The JDC implements the following IPS data registers, occupying separate memory space:

- 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 689*.

Figure 689. JDC block diagram



52.3 Register definition

Table 714 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 714. JDC memory map

Address offset	Register	Access	Reset value	Location
0x0	Module Configuration Register (JDC_MCR)	R/W	0x0000_0000	*Section 52.3.1.1*
0x4	Module Status Register (JDC_MSR)	W1C	0x0000_0000	*Section 52.3.1.2*
0x8	JTAG Output IPS Data Register (JDC_JOUT_IPS)	R/W	0x0000_0000	*Section 52.3.1.3*
0xC	JTAG Input IPS Data Register (JDC_JIN_IPS)	Read Only	0x0000_0000	*Section 52.3.1.4*

52.3.1 Register descriptions

52.3.1.1 Module configuration register (JDC_MCR)

Figure 690 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 690. Module Configuration Register (JDC_MCR)

The JDC_MCR is described in *Table 715*.

Table 715. 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

52.3.1.2 Module status register (JDC_MSR)

Figure 691 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 691. Module Status Register (JDC_MSR)

The JDC_MSR is described in *Table 716*.

Table 716. 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

52.3.1.3 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 692* 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 692. JDC_JOUT_IPS register

The JDC_JOUT_IPS register is described in *Table 717*.

Table 717. JDC_JOUT_IPS register field descriptions

Field	Description
0:31 DATA	Contains JOUT_IPS data.

52.3.1.4 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 693* shows the format of the JDC_JIN_IPS register. This register is reset by system destructive reset.

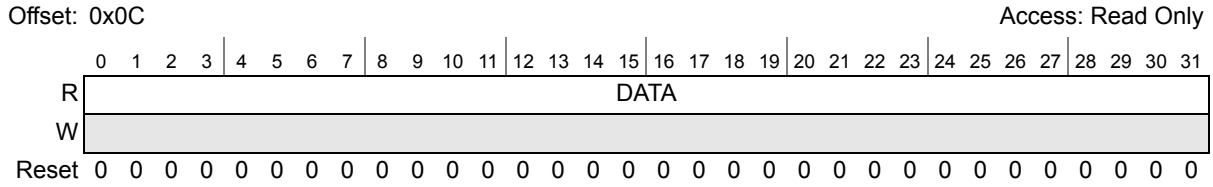


Figure 693. JDC_JIN_IPS register

The JDC_JIN_IPS register is described in *Table 718*.

Table 718. JDC_JIN_IPS register field descriptions

Field	Description
0:31 DATA	Contains JIN_IPS data.

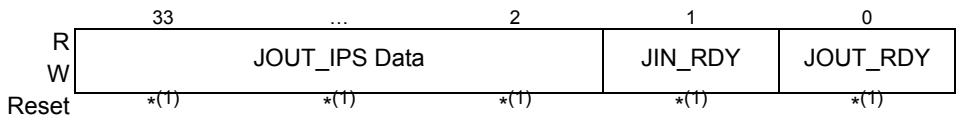
52.3.2 Non-Memory mapped register definition

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.

52.3.2.1 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 694* shows the format of the JOUT register. This register is reset by system destructive reset and JTAG reset.

Figure 694. JOUT JTAG register



1. The reset value of the JOUT register is 34'b0.

The JOUT register is described in *Table 719*.

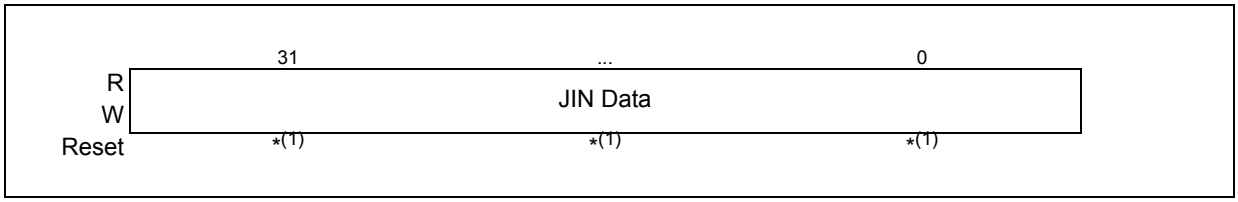
Table 719. JOUT JTAG register field descriptions

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

52.3.2.2 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 695* shows the format of the JIN register. This register is reset by system destructive reset and JTAG reset.

Figure 695. JIN JTAG Register



1. The reset value of the JIN register is 32'b0.

The JIN register is described in *Table 720*.

Table 720. 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.

52.4 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.

52.4.1 IPS register access

IPS access is available to the registers described in *Section 52.3.1*.

52.4.2 JTAG register access

Refer to the JTAGC documentation for information on how to access the JTAG registers.

52.4.2.1 JDC block instructions

The JDC block implements the IEEE 1149.1-2001 defined instructions listed in *Table 721*. 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 721. 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

53 JTAG Master (JTAGM)

53.1 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.

53.1.1 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:

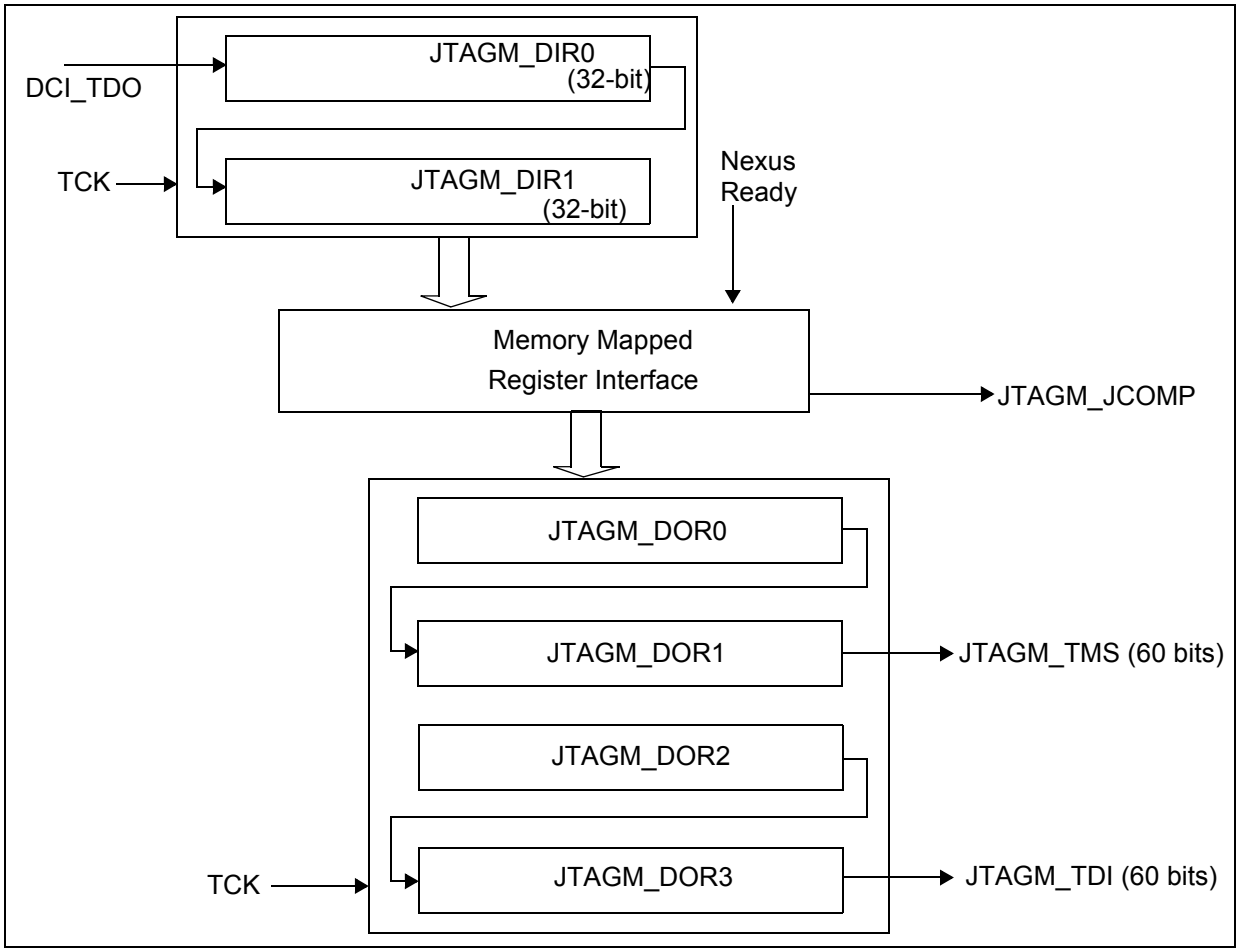
- Configuration register
- Status register
- Four data output registers
- Two data input registers

All these registers, described in detail in *Section 53.4*, 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 696 shows the block diagram of the JTAGM.

Figure 696. JTAGM block diagram



53.1.2 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

53.2 Functional description

53.2.1 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.

53.2.2 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.

53.2.2.1 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.

53.2.2.2 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.

53.2.3 $\overline{\text{EVTO}}$ 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.

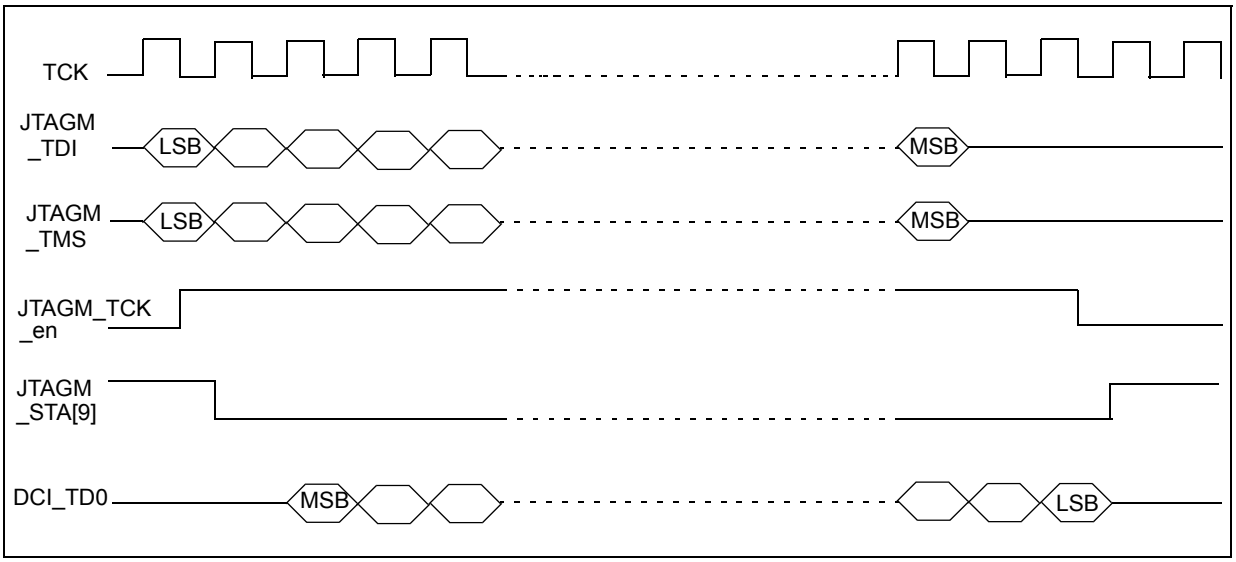
53.2.4 JTAGM configuration and status monitoring

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 53.2.2.1: Status register ready bit*). The status register also includes status bits provided by the DCI module.

53.2.5 JTAGM to DCI serial interface

Figure 697 shows the data transfer timings between JTAGM and DCI.

Figure 697. JTAGM to DCI serial interface diagram



53.2.6 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.

53.3 Modes of operation

There are no special modes of operation. The JTAGM works normally in the debug mode.

53.4 Memory map and register definition

Table 722 contains the memory map of the JTAGM registers.

Table 722. Memory map

Offset	Register	Location
0x00	Module Configuration Register (JTAGM_MCR)	*Section 53.4.1.1*
0x04	Status Register (JTAGM_SR)	*Section 53.4.1.2*

Table 722. Memory map(Continued)

Offset	Register	Location
0x08	Data Out Register 0 (JTAGM_DOR0)	*Section 53.4.1.3*
0x0C	Data Out Register 1 (JTAGM_DOR1)	*Section 53.4.1.4*
0x10	Data Out Register 2 (JTAGM_DOR2)	*Section 53.4.1.5*
0x14	Data Out Register 3 (JTAGM_DOR3)	*Section 53.4.1.5*
0x18	Reserved	
0x1C	Data Input Register 0 (JTAGM_DIR0)	*Section 53.4.1.7*
0x20	Data Input Register 1 (JTAGM_DIR1)	*Section 53.4.1.8*

53.4.1 Register descriptions

53.4.1.1 Module configuration register (JTAGM_MCR)

This register contains the status bits for the current transfer. *Figure 698* shows the format of the JTAGM_MCR.

Note: Bit-set operation on this register is prohibited as it can lead to device going to debug mode.

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 698. JTAGM_MCR register

The JTAGM_MCR register is described in *Table 723*.

Table 723. 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
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

Table 723. JTAGM_MCR register field descriptions(Continued)

Field	Description
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.

53.4.1.2 Status register (JTAGM_SR)

Figure 699 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 699. JTAGM_SR register

The JTAGM_SR register is described in *Table 724*.

Table 724. 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{EVTO}}$0 1 Activity defined by evto0_sense detected on $\overline{\text{EVTO}}$0
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{EVTO}}$1 1 Activity defined by evto1_sense detected on $\overline{\text{EVTO}}$1
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.
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.

53.4.1.3 Data out register 0 (JTAGM_DOR0)

Figure 700 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 700. JTAGM_DOR0 register

The JTAGM_DOR0 register is described in *Table 725*.

Table 725. 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

53.4.1.4 Data out register 1 (JTAGM_DOR1)

Figure 701 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 701. JTAGM_DOR1 register

The JTAGM_DOR1 register is described in *Table 726*.

Table 726. 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

53.4.1.5 Data out register 2 (JTAGM_DOR2)

Figure 702 shows the format of the JTAGM_DOR2.

Offset 0x10 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	TDI_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 702. JTAGM_DOR2 register

The JTAGM_DOR2 register is described in *Table 727*.

Table 727. 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

53.4.1.6 Data out register 3 (JTAGM_DOR3)

Figure 703 shows the format of the JTAGM_DOR3.

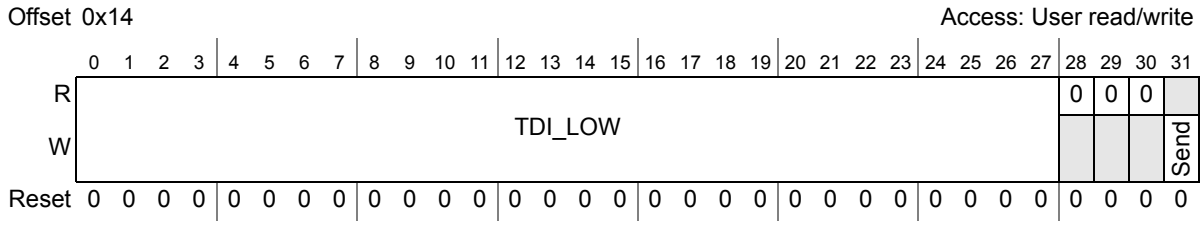


Figure 703. JTAGM_DOR3 register

The JTAGM_DOR3 register is described in *Table 728*.

Table 728. 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

53.4.1.7 Data input register 0 (JTAGM_DIR0)

Figure 704 shows the format of the JTAGM_DIR0.

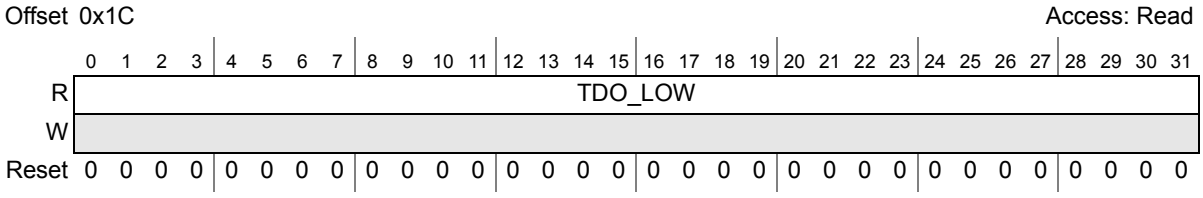


Figure 704. JTAGM_DIR0 register

The JTAGM_DIR0 register is described in *Table 729*.

Table 729. JTAGM_DIR0 register field descriptions

Field	Description
0:31 TDO_LOW	Lower word of data received on TDO, bits 0–31

53.4.1.8 Data input register 1 (JTAGM_DIR1)

Figure 705 shows the format of the JTAGM_DIR1.

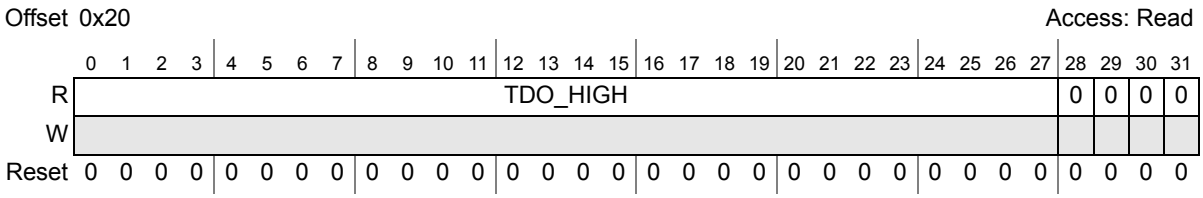


Figure 705. JTAGM_DIR1 register

The JTAGM_DIR1 register is described in *Table 730*.

Table 730. JTAGM_DIR1 register field descriptions

Field	Description
0:27 TDO_HIGH	Higher word of data received on TDO, bits 32–59

54 Nexus Port Controller (NPC)

54.1 Information specific to this device

This section presents device-specific parameterization, customization, and feature availability information not specifically referenced in the remainder of this chapter.

54.1.1 Parameter values

The following table provides the parameter values for this device.

Table 731. Device parameter values

Parameter	Value
Number of MDO pins available in reduced-port mode	4
Number of MDO pins available in full-port mode	12
Number of blocks that input an $\overline{\text{EVTO}}$	1
Number of $\overline{\text{MSEO}}$ pins available	1

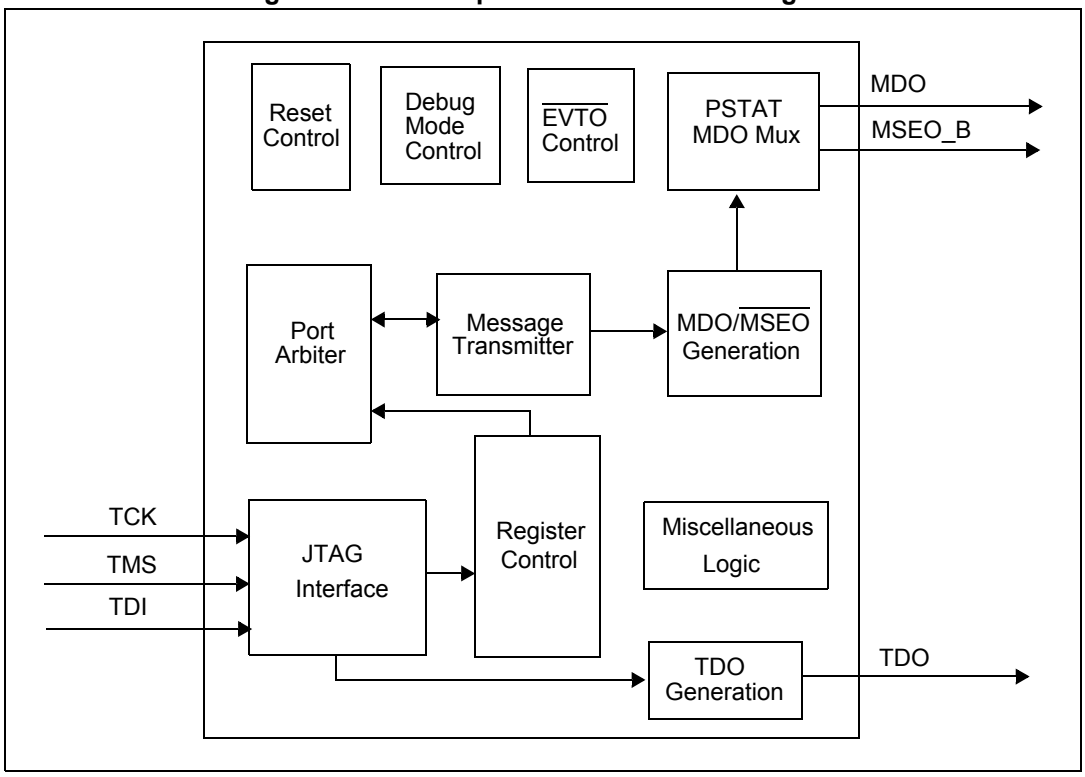
54.1.2 Unavailable features

This device does not support an MCKO division factor of 3. Therefore, the setting PCR[MCKO_DIV] = 2 (see *Table 736*) is not available on this device.

54.2 Introduction

The following figure shows the Nexus Port Controller (NPC) block diagram.

Figure 706. Nexus port controller block diagram



54.2.1 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 5001-2003 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.

54.2.2 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
- Generates power-on reset status
- Provides Nexus support for censorship mode
- $\overline{\text{RDY}}$ pin support to increase the transfer rate of the IEEE 1149.1 port for large data read requests

54.2.3 Modes of operation

The NPC block uses the censorship status 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.

54.2.3.1 Reset

The NPC block is asynchronously placed in reset when either power-on reset is asserted, the device enters censored mode, or the TAP controller state machine is in 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. 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 censored mode only)
- Registers default back to their reset values

54.2.3.2 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 $\overline{\text{EVTO}}$.

54.2.3.3 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.

54.2.3.4 Reduced-Port mode

Reduced-port mode (RPM) is entered by asserting the MCKO_EN bit and deasserting 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.

54.2.3.5 Censored mode

When the device is in censored mode, reading the contents of internal flash externally is not allowed. To prevent Nexus modules from violating censorship, the NPC is held in reset when in censored mode, asynchronously holding all other Nexus modules in reset as well. This prevents Nexus read/write to memory mapped resources and the transmission of Nexus trace messages.

54.3 External signal description

54.3.1 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 the following table.

Table 732. NPC signal properties

Name	Port	Function	Reset state	Pull(1)
EVTO	Auxiliary	Event Out pin	0b1	—
MDO	Auxiliary	Message Data Out pins	0	—
MSEO	Auxiliary	Message Start/End Out pin	0b1	—
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
RDY	JTAG	Data ready for transfer to/from NRRs	—	—

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.

54.3.2 Detailed signal descriptions

This section describes each of the signals listed in *Table 732* in more detail. 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.

54.3.2.1 $\overline{\text{EVTO}}$ - 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.

54.3.2.2 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.

54.3.2.3 MSEO_B - Message Start/End Out

Message Start/End Out ($\overline{\text{MSEO}}$) is one 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.

54.3.2.4 TCK - Test Clock Input

Test Clock Input (TCK) pin is used to synchronize the test logic and control register access through the JTAG port.

54.3.2.5 TDI - Test Data Input

Test Data Input (TDI) pin receives serial test instruction and data. TDI is sampled on the rising edge of TCK.

54.3.2.6 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.

54.3.2.7 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.

54.3.2.8 $\overline{\text{RDY}}$ - Data ready for transfer

The $\overline{\text{RDY}}$ pin exists to increase the transfer rate of the IEEE 1149.1 port. It is used to signal when data are ready to be transferred to and from NRRs. This may eliminate the need to poll NRRs for status information for synchronization purposes. This capability becomes especially important when performing read/write access transfers to different speed target memories.

The $\overline{\text{RDY}}$ pin asserts (asynchronously) a logic low whenever the read/write access transfer has completed without error and then deasserts when the IEEE 1149.1 state machine has reached the CAPTURE_DR state.

54.4 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 733 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 733. NPC registers

Index	Register
0	Device ID Register (DID)
127	Port Configuration Register (PCR)

54.4.1 Register descriptions

This section contains the NPC register descriptions.

54.4.1.1 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.

54.4.1.2 Instruction Register

The NPC block uses a 4-bit instruction register as shown in *Figure 707*. 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.

	0	1	2	3
R	Previous Instruction Opcode			
W	Instruction Opcode			
Reset:	BYPASS Instruction Opcode (0xF)			

Figure 707. 4-bit Instruction Register

54.4.1.3 Nexus Device ID Register (DID)

The device identification register allows identification information 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	0	Manufacturer ID											0	Sub Family		
W																
Reset	0	0	0	0	0	0	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	Sub Family	Application Family			Technology Node		Design Center		Minor Mask				Major Mask			
W																
Reset	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0

Figure 708. Nexus Device ID Register

Table 734. DID field descriptions

Field	Description
0	Reserved
1:11 Manufacturer ID	These bits contain the Joint Electron Device Engineering Council (JEDEC) manufacturer's identification code. 00000100000: STMicroelectronics
12	Reserved
13:16 Sub Family	0001 SPC570Sx
17:19 Application Family	101 SPC57S - Chassis & Safety 55nm 110 SPC58S - Chassis & Safety 40nm
20:21 Technology Node	00 55nm 01 40nm
22:23 Design Center	These bits indicate the device design center.
24:27 Minor Mask	Minor mask revision
28:31 Major Mask	Major mask revision

54.4.1.4 Port Configuration Register (PCR)

The PCR 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.

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.

Register index: 127

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	FPM	MCKO_GT	MCKO_EN	MCKO_DIV[2:0]			Reserved	DDR_EN	Reserved	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0(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	Reserved	0	0	0	0	0	0	0	0	0	0	0	0	0	0	PSTAT_EN
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 its value has no impact.

Figure 709. Port Configuration Register (PCR)

Table 735. PCR field descriptions

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
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[2:0]	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 736* shows the meaning of MCKO_DIV values. In this table, SYS_CLK represents the system clock frequency.
7 DDR_EN	Double Data Rate Mode Enable(1) This bit enables Nexus Double Data Rate (DDR) mode. In DDR mode, message data is updated on both rising and falling edges of MCKO, effectively doubling message throughput. 1 DDR mode enabled 0 DDR mode disabled
16	Not functionally used but this bit can be written by software.
31 PSTAT_EN	Processor Status Mode Enable(2) This bit enables processor status (PSTAT) mode. In PSTAT mode, all auxiliary output port MDO pins are used to transmit processor status information, and Nexus messaging is unavailable. 1 PSTAT mode enabled 0 PSTAT mode disabled

1. DDR not supported on this device. Do not write to this bit; value must be kept at 0.
2. PSTAT mode is intended for factory processor debug only. The PSTAT_EN bit should be written to disable PSTAT mode if Nexus messaging is desired. No Nexus messages are transmitted under any circumstances when PSTAT mode is enabled.

Table 736. PCR[MCKO_DIV] values

MCKO_DIV[2:0]	MCKO frequency
0d (000b)	SYS_CLK(1)
1d (001b)	SYS_CLK/2
2d (010b)	SYS_CLK/3(2)
3d (011b)	SYS_CLK/4
4d (100b)	Reserved
5d (101b)	Reserved
6d (110b)	Reserved
7d (111b)	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.
2. SYS_CLK/3 setting must not be used in DDR mode.

54.5 Functional description

54.5.1 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.

The following table describes the NPC reset configuration options.

Table 737. NPC reset configuration options

NPC_PCR[MCKO_EN]	NPC_PCR[FPM]	Configuration
X	X	Reset
0	X	Disabled
1	1	Full-Port mode
1	0	Reduced-Port mode

54.5.2 Auxiliary output port

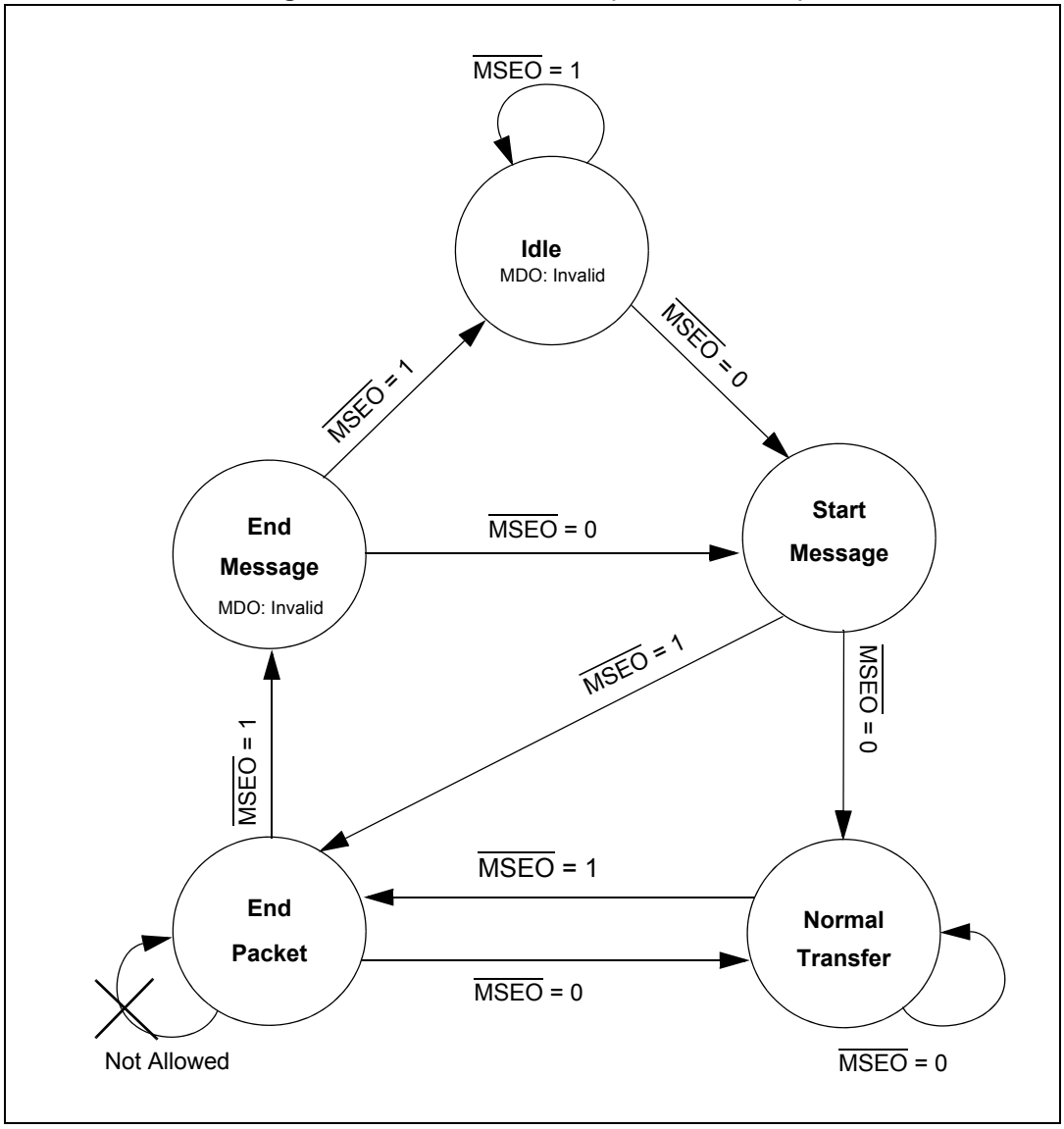
The auxiliary output port is shared by each of the Nexus modules on the device. The NPC communicates with each of the Nexus modules and arbitrates for access to the port.

54.5.2.1 Output Message Protocol

The protocol for transmitting messages via the auxiliary port is accomplished with the MSEO functions. The MSEO pin is used to signal the end of variable-length packets and the end of messages. It is not required to indicate the end of fixed-length packets. MDO and MSEO are sampled on the rising edge of MCKO.

Figure 710 illustrates the state diagram for $\overline{\text{MSEO}}$ transfers. All transitions not included in the figure are reserved, and must not be used.

Figure 710. $\overline{\text{MSEO}}$ transfers (for 1-bit $\overline{\text{MSEO}}$)



54.5.2.2 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 and the port replacement output message on the MDO pins. The device ID message can also be sent out serially through TDO.

Table 738 describes the device ID and port replacement output messages that the NPC can transmit on the auxiliary port. The TCODE is the first packet transmitted.

Table 738. NPC output messages

Message name	Packet				
	Size (bits)		Type	Name	Description
	Min	Max			
Device ID Message	6	6	fixed	TCODE	Value = 1
	32	32	fixed	ID	DID register contents

Figure 711 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).

Figure 711. Message field sizes

Message	TCODE	Field #1	Field #2	Field #3	Field #4	Field #5	Size (bits)	
							Min(1)	Max(2)
Device ID Message	1	Fixed = 32	—	—	—	—	38	38

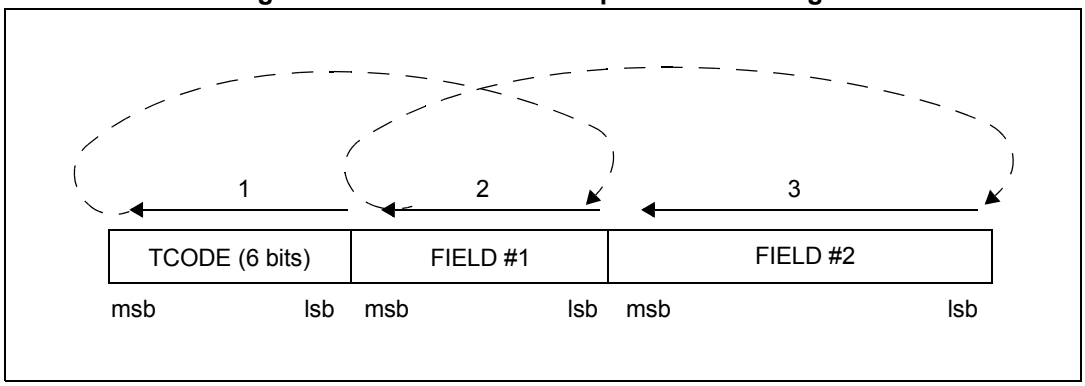
1. Minimum information size. The actual number of bits transmitted depends on the number of MDO pins.
2. Maximum information size. The actual number of bits transmitted depends on the number of MDO pins.

The double edges in *Figure 711* 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.

54.5.2.3 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 712* shows the transmission sequence of a message that is made up of a TCODE followed by two fields.

Figure 712. Transmission sequence of messages



54.5.3 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 714*. The Nexus controller state machine is defined by the IEEE-ISTO 5001-2001 standard. It is shown in *Figure 715*.

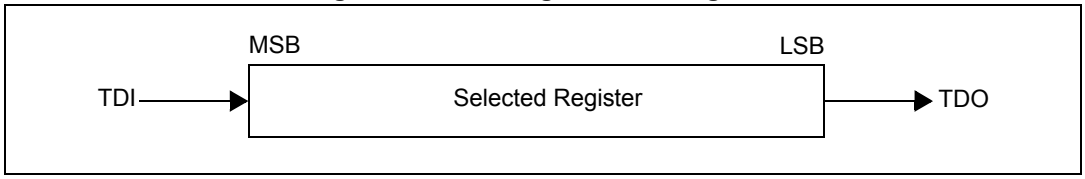
The instructions implemented by the NPC TAP controller are listed in *Table 739*. The value of the NEXUS-ENABLE instruction is 0b0000. Each unimplemented instruction acts like the BYPASS instruction. The size of the NPC instruction register is 4 bits.

Table 739. 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 713*. This applies for the instruction register and all Nexus tool-mapped registers.

Figure 713. Shifting data into register

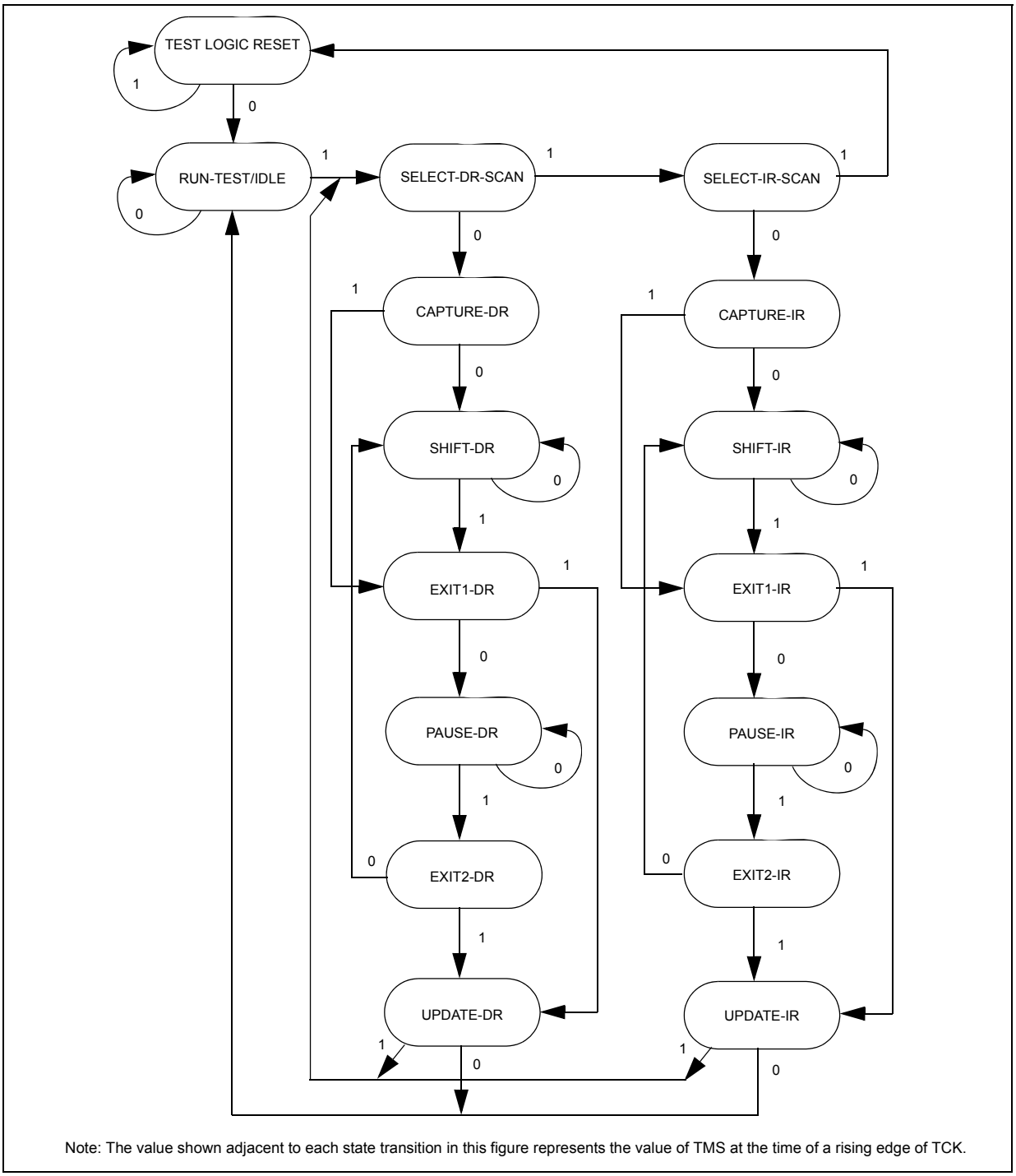


54.5.3.1 Enabling the NPC TAP controller

Assertion of the power-on reset signal or entry into censored mode resets the NPC TAP controller.

When not in power-on reset or censored mode, the NPC TAP controller is enabled exiting the Test-Logic-Reset state loading the NEXUS-ENABLE instruction in NPC TAP controller.

Figure 714. 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.

54.5.3.2 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 54.5.3.4: Selecting a Nexus client register*.

54.5.3.3 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 715*, transitions to the REG_SELECT state. The Nexus controller has three states: idle, register select, and data access. *Table 740* illustrates the IEEE 1149.1 sequence to load the NEXUS-ENABLE instruction.

Figure 715. NEXUS controller state machine

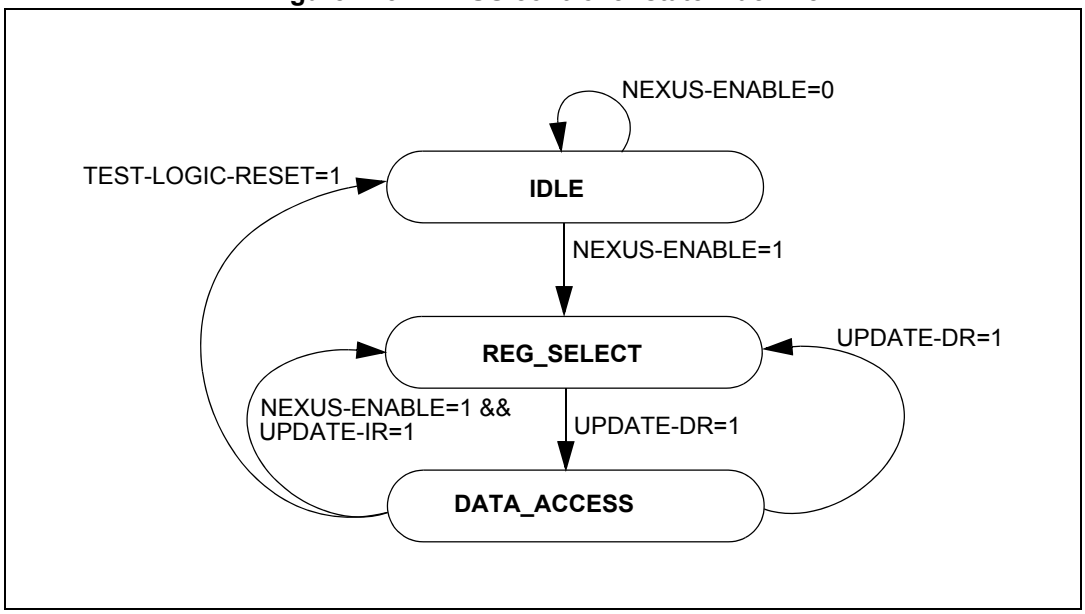


Table 740. Loading NEXUS-ENABLE instruction

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

Table 740. Loading NEXUS-ENABLE instruction(Continued)

Clock	TMS	IEEE 1149.1 state	Nexus state	Description
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

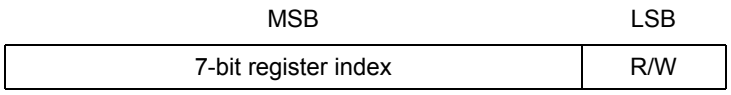
54.5.3.4 Selecting a Nexus client register

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 716*. The read/write control bit is set to '1' for writes and '0' for reads.

Figure 716. 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.

The following table illustrates a sequence which writes a 32-bit value to a register.

Table 741. 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

Table 741. Write to a 32-bit Nexus Client Register(Continued)

Clock	TMS	IEEE 1149.1 state	Nexus state	Description
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				
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.

54.5.4 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.

54.5.5 MCKO and ipg_sync_mcko

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.

54.5.6 $\overline{\text{EVTO}}$ 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.

54.5.7 Nexus reset control

Internal Nexus reset is asserted when the device is in censored mode.

54.6 Initialization/Application information

54.6.1 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 5001-2003 standard for more detail.

55 Sequence Processing Unit (SPU)

55.1 Introduction

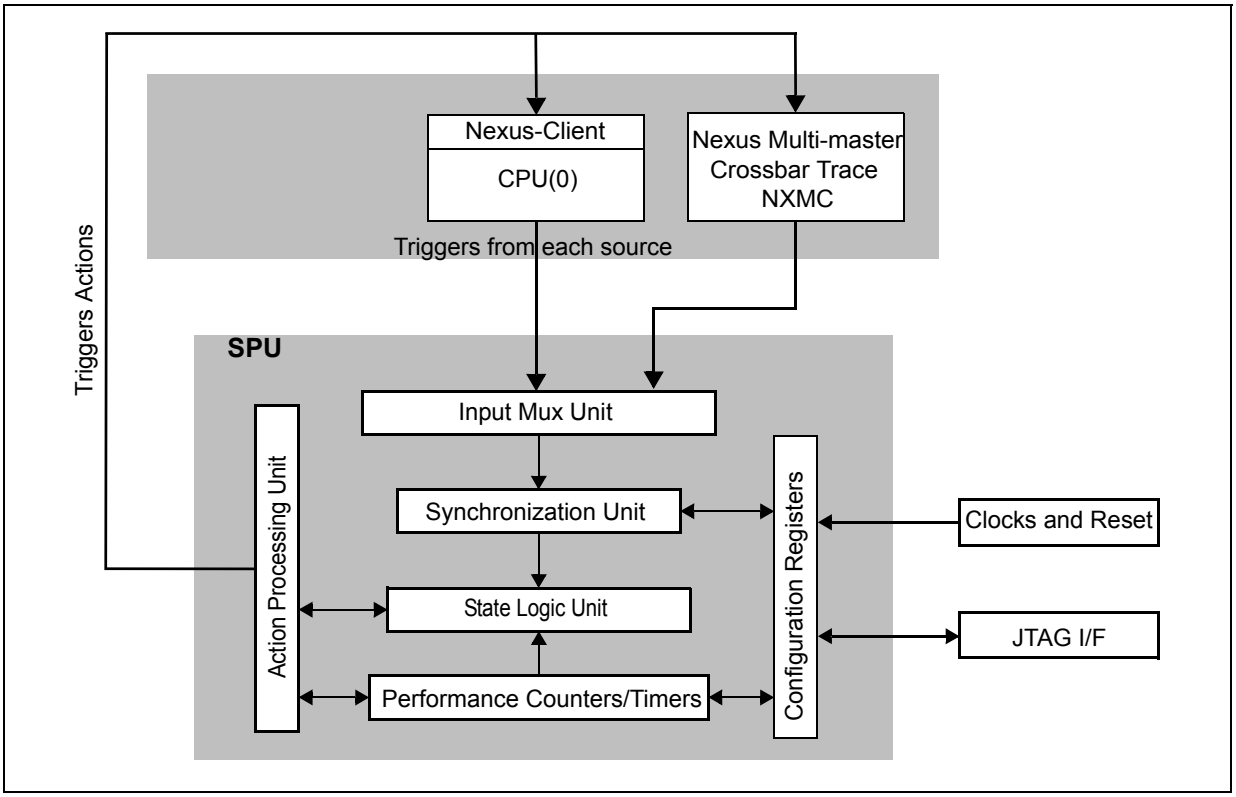
The Sequence Processing Unit (SPU) provides an on-device trigger functions similar to those found on a logic analyzer. In a complex SoC, performance monitor functions are available at two levels: system level and CPU level. Complex trigger and system performance monitor functions are implemented in the 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.

55.1.1 SPU block diagram

Figure 717 shows a top-level block diagram of the SPU.

Figure 717. SPU block diagram



55.1.2 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 752*.

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 753*. 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 754* and *Figure 755*. 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 CPU0 writes to VariableB. If CPU0 writes a value to VariableB which is 0x100, then enable trace for the halt CPU0.

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 752*.
- 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 753*.
- 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.

55.1.3 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 55.2*

55.2 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
- 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.

55.3 Operation mode

55.3.1 SPU enable/disable

By default, the SPU is in disabled mode. In this state:

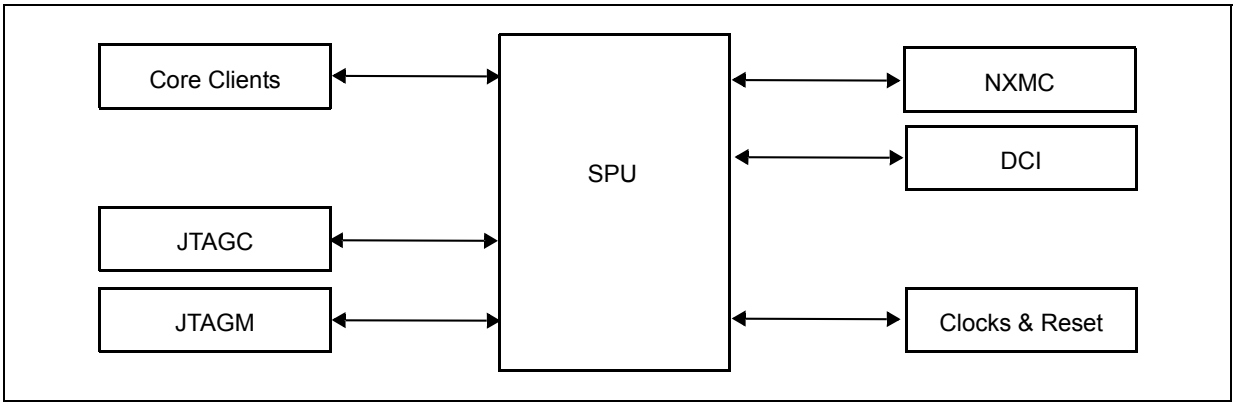
- All events/inputs are disabled such that no counting or other actions occur
- All the SPU control registers are accessible

The user must configure the control registers before enabling the SPU via the SPU Enable/Disable register.

55.4 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 718*.

Figure 718. SPU module interface



55.5 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 742 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 742. SPU register summary

Offset	Register	Location
0x01	Level1 Mux Selection 0 (L1SEL0)	*55.5.1.1.1 on page 1312*
0x02	Level1 Mux Selection 1 (L1SEL1)	*55.5.1.1.2 on page 1312*
0x03	Level1 Mux Selection 2 (L1SEL2)	*55.5.1.1.3 on page 1312*
0x04	Level1 Mux Selection 3 (L1SEL3)	*55.5.1.1.4 on page 1313*
0x05	Level1 Mux Selection 4(L1SEL4)	*55.5.1.1.5 on page 1313*
0x06	Level1 Mux Selection 5 (L1SEL5)	*55.5.1.1.6 on page 1314*
0x07	Level1 Mux Selection 6 (L1SEL6)	*55.5.1.1.7 on page 1314*
0x08	Level1 Mux Selection 7 (L1SEL7)	*55.5.1.1.8 on page 1315*
0x09–0x0A	Reserved for future use	
0x0B	Input Trigger Level Detection 2 (ITLD2)	*55.5.1.1.9 on page 1315*
0x0C	Input Trigger Level Detection 3 (ITLD3)	*55.5.1.1.10 on page 1318*
0x0D	CPU0 Processor Except Vector Prefix (C0PEVP)	*55.5.1.1.11 on page 1320*
0x0E	Reserved	
0x0F	Reserved	
0x10–0x11	Reserved for future use	
0x12	CPU0 Interrupt Priority Selection (C0PIS)	*55.5.1.1.12 on page 1321*
13	Reserved	
14	Reserved	
0x15–0x16	Reserved for future use	
0x17	Level2 Mux State0 Selection 0 (L20SEL0)	*55.5.1.2.1 on page 1322*

Table 742. SPU register summary(Continued)

Offset	Register	Location
0x18	Level2 Mux State0 Selection 1 (L20SEL1)	55.5.1.2.2 on page 1324
0x19	Level2 Mux State0 Selection 2 (L20SEL2)	55.5.1.2.3 on page 1325
0x1A	Level2 Mux State0 Selection 3 (L20SEL3)	55.5.1.2.4 on page 1326
0x1B	Level2 Mux State1 Selection 0 (L21SEL0)	55.5.1.2.1 on page 1322
0x1C	Level2 Mux State1 Selection 1 (L21SEL1)	55.5.1.2.2 on page 1324
0x1D	Level2 Mux State1 Selection 2 (L21SEL2)	55.5.1.2.3 on page 1325
0x1E	Level2 Mux State1 Selection 3 (L21SEL3)	55.5.1.2.4 on page 1326
0x1F	Level2 Mux State2 Selection 0 (L22SEL0)	55.5.1.2.1 on page 1322
0x20	Level2 Mux State2 Selection 1 (L22SEL1)	55.5.1.2.2 on page 1324
0x21	Level2 Mux State2 Selection 2 (L22SEL2)	55.5.1.2.3 on page 1325
0x22	Level2 Mux State2 Selection 3 (L22SEL3)	55.5.1.2.4 on page 1326
0x23	Level2 Mux State3 Selection 0 (L23SEL0)	55.5.1.2.1 on page 1322
0x24	Level2 Mux State3 Selection 1 (L23SEL1)	55.5.1.2.2 on page 1324
0x25	Level2 Mux State3 Selection 2 (L23SEL2)	55.5.1.2.3 on page 1325
0x26	Level2 Mux State3 Selection 3 (L23SEL3)	55.5.1.2.4 on page 1326
0x27	Level2 Mux State4 Selection 0 (L24SEL0)	55.5.1.2.1 on page 1322
0x28	Level2 Mux State4 Selection 1 (L24SEL1)	55.5.1.2.2 on page 1324
0x29	Level2 Mux State4 Selection 2 (L24SEL2)	55.5.1.2.3 on page 1325
0x2A	Level2 Mux State4 Selection 3 (L24SEL3)	55.5.1.2.4 on page 1326
0x2B	Level2 Mux State5 Selection 0 (L25SEL0)	55.5.1.2.1 on page 1322
0x2C	Level2 Mux State5 Selection 1 (L25SEL1)	55.5.1.2.2 on page 1324

Table 742. SPU register summary(Continued)

Offset	Register	Location
0x2D	Level2 Mux State5 Selection 2 (L25SEL2)	55.5.1.2.3 on page 1325
0x2E	Level2 Mux State6 Selection 3 (L25SEL3)	55.5.1.2.4 on page 1326
0x2F	Level2 Mux State6 Selection 0 (L26SEL0)	55.5.1.2.1 on page 1322
0x30	Level2 Mux State6 Selection 1 (L26SEL1)	55.5.1.2.2 on page 1324
0x31	Level2 Mux State6 Selection 2 (L26SEL2)	55.5.1.2.3 on page 1325
0x32	Level2 Mux State6 Selection 3 (L26SEL3)	55.5.1.2.4 on page 1326
0x33	Level2 Mux State7 Selection 0 (L27SEL0)	55.5.1.2.1 on page 1322
0x34	Level2 Mux State7 Selection 1 (L27SEL1)	55.5.1.2.2 on page 1324
0x35	Level2 Mux State7 Selection 2 (L27SEL2)	55.5.1.2.3 on page 1325
0x36	Level2 Mux State7 Selection 3 (L27SEL3)	55.5.1.2.4 on page 1326
0x37	Input/Output Inversion Control State0 (IOIC0)	55.5.1.3.1 on page 1328
0x38	Input/Output Inversion Control State1 (IOIC1)	55.5.1.3.1 on page 1328
0x39	Input/Output Inversion Control State2 (IOIC2)	55.5.1.3.1 on page 1328
0x3A	Input/Output Inversion Control State3 (IOIC3)	55.5.1.3.1 on page 1328
0x3B	Input/Output Inversion Control State4 (IOIC4)	55.5.1.3.1 on page 1328
0x3C	Input/Output Inversion Control State5 (IOIC5)	55.5.1.3.1 on page 1328
0x3D	Input/Output Inversion Control State6 (IOIC6)	55.5.1.3.1 on page 1328
0x3E	Input/Output Inversion Control State7 (IOIC7)	55.5.1.3.1 on page 1328
0x3F	Sequence Control (SCTRL)	55.5.1.3.2 on page 1329
0x40	State0 Goto Control (S0GC)	55.5.1.3.3 on page 1331
0x41	State1 Goto Control (S1GC)	55.5.1.3.3 on page 1331

Table 742. SPU register summary(Continued)

Offset	Register	Location
0x42	State2 Goto Control (S2GC)	55.5.1.3.3 on page 1331
0x43	State3 Goto Control (S3GC)	55.5.1.3.3 on page 1331
0x44	State4 Goto Control (S4GC)	55.5.1.3.3 on page 1331
0x45	State5 Goto Control (S5GC)	55.5.1.3.3 on page 1331
0x46	State6 Goto Control (S6GC)	55.5.1.3.3 on page 1331
0x47	State7 Goto Control (S7GC)	55.5.1.3.3 on page 1331
0x48	SLU Status (SS)	55.5.1.4 on page 1333
0x49	SPU Enable (SE)	55.5.1.5 on page 1336
0x01	State0 True Action 0 (S0TA0)	55.5.1.6.1 on page 1338
0x02	State0 True Action 1 (S0TA1)	55.5.1.6.2 on page 1339
0x03	State1 True Action 0 (S1TA0)	55.5.1.6.1 on page 1338
0x04	State1 True Action 1 (S1TA1)	55.5.1.6.2 on page 1339
0x05	State2 True Action 0 (S2TA0)	55.5.1.6.1 on page 1338
0x06	State2 True Action 1 (S2TA1)	55.5.1.6.2 on page 1339
0x07	State3 True Action 0 (S3TA0)	55.5.1.6.1 on page 1338
0x08	State3 True Action 1 (S3TA1)	55.5.1.6.2 on page 1339
0x09	State4 True Action 0 (S4TA0)	55.5.1.6.1 on page 1338
0x0A	State4 True Action 1 (S4TA1)	55.5.1.6.2 on page 1339
0x0B	State5 True Action 0 (S5TA0)	55.5.1.6.1 on page 1338
0x0C	State5 True Action 1(S5TA1)	55.5.1.6.2 on page 1339
0x0D	State6 True Action 0 (S6TA0)	55.5.1.6.1 on page 1338

Table 742. SPU register summary(Continued)

Offset	Register	Location
0x0E	State6 True Action 1(S6TA1)	55.5.1.6.2 on page 1339
0x0F	State7 True Action 0 (S7TA0)	55.5.1.6.1 on page 1338
0x10	State7 True Action 1(S7TA1)	55.5.1.6.2 on page 1339
0x11	State0 False Action 0 (S0FA0)	55.5.1.6.3 on page 1339
0x12	State0 False Action 1 (S0FA1)	55.5.1.6.4 on page 1340
0x13	State1 False Action 0 (S1FA0)	55.5.1.6.3 on page 1339
0x14	State1 False Action 1(S1FA1)	55.5.1.6.4 on page 1340
0x15	State2 False Action 0 (S2FA0)	55.5.1.6.3 on page 1339
0x16	State2 False Action 1(S2FA1)	55.5.1.6.4 on page 1340
0x17	State3 False Action 0 (S3FA0)	55.5.1.6.3 on page 1339
0x18	State3 False Action 1(S3FA0)	55.5.1.6.4 on page 1340
0x19	State4 False Action 0 (S4FA0)	55.5.1.6.3 on page 1339
0x1A	State4 False Action 1 (S4FA1)	55.5.1.6.4 on page 1340
0x1B	State5 False Action 0 (S5FA0)	55.5.1.6.3 on page 1339
0x1C	State5 False Action 1 (S5FA1)	55.5.1.6.4 on page 1340
0x1D	State6 False Action 0 (S6FA0)	55.5.1.6.3 on page 1339
0x1E	State6 False Action 1 (S6FA1)	55.5.1.6.4 on page 1340
0x1F	State7 False Action 0 (S7FA0)	55.5.1.6.3 on page 1339
0x20	State7 False Action 1 (S7FA1)	55.5.1.6.4 on page 1340
0x21	DTM Trace Group Configuration (DTMTGC)	55.5.1.6.5 on page 1341
0x22	PTM Trace Group Configuration (PTMTGC)	55.5.1.6.5 on page 1341

Table 742. SPU register summary(Continued)

Offset	Register	Location
0x23	OTM Trace Group Configuration (OTMTGC)	55.5.1.6.5 on page 1341
0x24	WTM Trace Group Configuration (WTMTGC)	55.5.1.6.5 on page 1341
0x25	Interrupt Status (INTS)	55.5.1.6.6 on page 1342
0x26	Counter Control 0 (CCTRL0)	55.5.1.7 on page 1342
0x27	Counter Control 1 (CCTRL1)	55.5.1.7 on page 1342
0x28	Counter Control 2 (CCTRL2)	55.5.1.7 on page 1342
0x29	Counter Control 3 (CCTRL3)	55.5.1.7 on page 1342
0x2A	Counter Control 4 (CCTRL4)	55.5.1.7 on page 1342
0x2B	Counter Control 5 (CCTRL5)	55.5.1.7 on page 1342
0x2C	Counter Control 6 (CCTRL6)	55.5.1.7 on page 1342
0x2D	Counter Control 7 (CCTRL7)	55.5.1.7 on page 1342
0x2E	Counter Control 8 (CCTRL8)	55.5.1.7 on page 1342
0x2F	Counter Control 9 (CCTRL9)	55.5.1.7 on page 1342
0x30	Counter Control 10 (CCTRL10)	55.5.1.7 on page 1342
0x31	Counter Control 11 (CCTRL11)	55.5.1.7 on page 1342
0x32	Counter Control 12 (CCTRL12)	55.5.1.7 on page 1342
0x33	Counter Control 13 (CCTRL13)	55.5.1.7 on page 1342
0x34	Counter Control 14 (CCTRL14)	55.5.1.7 on page 1342
0x35	Counter Control 15 (CCTRL15)	55.5.1.7 on page 1342
0x36	Counter Compare 0(CCMP0)	55.5.1.8 on page 1344
0x37	Counter Compare 1 (CCMP1)	55.5.1.8 on page 1344

Table 742. SPU register summary(Continued)

Offset	Register	Location
0x38	Counter Compare 2 (CCMP2)	55.5.1.8 on page 1344
0x39	Counter Compare 3 (CCMP3)	55.5.1.8 on page 1344
0x3A	Counter Compare 4 (CCMP4)	55.5.1.8 on page 1344
0x3B	Counter Compare 5 (CCMP5)	55.5.1.8 on page 1344
0x3C	Counter Compare 6 (CCMP6)	55.5.1.8 on page 1344
0x3D	Counter Compare 7 (CCMP7)	55.5.1.8 on page 1344
0x3E	Counter Compare 8 (CCMP8)	55.5.1.8 on page 1344
0x3F	Counter Compare 9 (CCMP9)	55.5.1.8 on page 1344
0x40	Counter Compare 10 (CCMP10)	55.5.1.8 on page 1344
0x41	Counter Compare 11 (CCMP11)	55.5.1.8 on page 1344
0x42	Counter Compare 12 (CCMP12)	55.5.1.8 on page 1344
0x43	Counter Compare 13 (CCMP13)	55.5.1.8 on page 1344
0x44	Counter Compare 14 (CCMP14)	55.5.1.8 on page 1344
0x45	Counter Compare 15 (CCMP15)	55.5.1.8 on page 1344
0x46	Counter Compare Status (CCOMS)	55.5.1.9.1 on page 1344
0x47	Counter Overflow Status (COS)	55.5.1.9.2 on page 1346
0x48	Counter Capture Status (CCAPS)	55.5.1.9.3 on page 1348

55.5.1 Registers description

55.5.1.1 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 752*.

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).

55.5.1.1.1 Level1 Mux Selection 0 (L1SEL0)

Figure 719 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 the device-specific 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 719. Level1 Mux Selection 0 (L1SEL0)

55.5.1.1.2 Level1 Mux Selection 1 (L1SEL1)

Figure 720 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 the device-specific 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 720. Level1 Mux Selection 1 (L1SEL1)

55.5.1.1.3 Level1 Mux Selection 2 (L1SEL2)

Figure 721 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 the device-specific 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 721. Level1 Mux Selection 2 (L1SEL2)

55.5.1.1.4 Level1 Mux Selection 3 (L1SEL3)

Figure 722 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 the device-specific 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 722. Level1 Mux Selection 3 (L1SEL3)

55.5.1.1.5 Level1 Mux Selection 4 (L1SEL4)

Figure 723 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 the device-specific 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 723. Level1 Mux Selection 4 (L1SEL4)

55.5.1.1.6 Level1 Mux Selection 5 (L1SEL5)

Figure 724 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 the device-specific 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 724. Level1 Mux Selection 5 (L1SEL5)

55.5.1.1.7 Level1 Mux Selection 6 (L1SEL6)

Figure 725 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 the device-specific 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 725. Level1 Mux Selection 6 (L1SEL6)

55.5.1.1.8 Level1 Mux Selection 7 (L1SEL7)

Figure 726 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 the device-specific 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 726. Level1 Mux Selection 7 (L1SEL7)

55.5.1.1.9 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 727* 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 W	Mux47		Mux46		Mux45		Mux44		Mux43		Mux42		Mux41		Mux40	
Reset	0	0	0	0	0	0	0	0	0	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	Mux39		Mux38		Mux37		Mux36		Mux35		Mux34		Mux33		Mux32	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 727. Input triggers level detection control 2 (ITLD2)

The ITLD2 registers Mux fields are described in *Table 743*.

Table 743. 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
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

Table 743. ITLD2 register field descriptions(Continued)

Field	Description
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
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

Table 743. ITLD2 register field descriptions(Continued)

Field	Description
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

55.5.1.1.10 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 728* 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

Figure 728. Input triggers level detection control 3 (ITLD3)

The ITLD3 registers Mux fields are described in *Table 744*.

Table 744. 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

Table 744. ITLD3 register field descriptions(Continued)

Field	Description
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
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

Table 744. ITLD3 register field descriptions(Continued)

Field	Description
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

55.5.1.1.11 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 768*. The SPU compares this captured vector with the value in the C0PEVP register. *Figure 729* 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 729. CPU0 processor exception vector prefix (C0PEVP)

The C0PEVP register is described in *Table 745*.

Table 745. C0PEVP register field descriptions

Field	Description
16:31 CPU0	CPU0 processor exception vector Program this field with one of the vector values from *Table 768* to select the corresponding processor exception. This field must use the byte swapped values in the CnPEVP register CPUn field value column in *Table 768*.

55.5.1.1.12 CPU0 interrupt priority selection (C0PIS)

Figure 730 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 730. CPU0 interrupt priority selection (C0PIS)

The C0PIS register field is described in *Table 746*.

Table 746. C0PIS register field descriptions

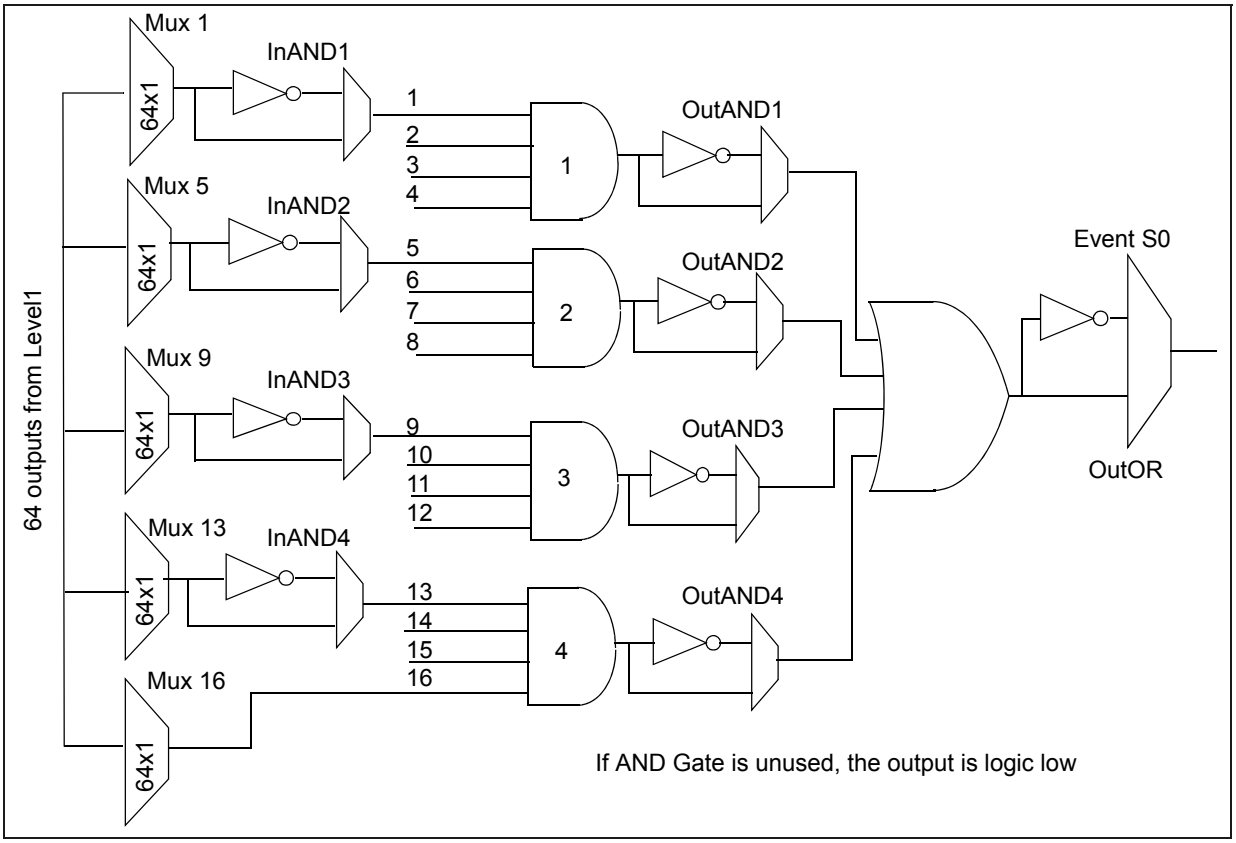
Field	Description
24:31 CPU0	CPU0 Priority Interrupt Selection

55.5.1.2 Level2 input Mux configurations registers

At level1, the set of input Mux control registers (described in *Section 55.5.1.1: Level1 input Mux configurations registers*) 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 752* for State0. Similar logic is provided for States 1 to 7.

Figure 731. 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.

55.5.1.2.1 Level2 Mux state n selection 0 (L2nSEL0)

Figure 732 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 732. Level2 Mux state n selection 0 (L2nSEL0)

The L2nSEL0 register fields are described in *Table 747*.

Table 747. 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
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

55.5.1.2.2 Level2 Mux state n selection 1 (L2nSEL1)

Figure 733 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 733. Level2 Mux state n selection 1 (L2nSEL1)

The L2nSEL1 register fields are described in *Table 748*.

Table 748. 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

Table 748. L2nSEL1 register field descriptions(Continued)

Field	Description
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

55.5.1.2.3 Level2 Mux state n selection 2 (L2nSEL2)

Figure 734 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 734. Level2 Mux state n selection 2 (L2nSEL2)

The L2nSEL2 register fields are described in *Table 749*.

Table 749. 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

55.5.1.2.4 Level2 Mux state n selection 3 (L2nSEL3)

Figure 735 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 735. Level2 Mux state n selection 3 (L2nSEL3)

The L2nSEL3 register fields are described in *Table 750*.

Table 750. 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

55.5.1.3 Sequence unit registers

55.5.1.3.1 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 736* 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 736. Input/output Inversion control state n (IOICn)

The IOICn register fields are described in *Table 751*.

Table 751. 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 751. 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

55.5.1.3.2 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 737* 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 737. Sequence control (SCTRL)

The SCTRL register fields are described in *Table 752*.

Table 752. 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 tate 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 752. 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
28:29 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.

55.5.1.3.3 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 738 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 738. State n goto control (SnGC)

The SnGC register fields are described in *Table 753*.

Table 753. 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 753. 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

55.5.1.4 SLU status register (SS)

The SLU status register indicates the states that have been hit/encountered in any of the active sequences. *Figure 739* 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 [15]
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 [16:17]		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 739. SLU status register (SS)

The SS register fields are described in *Table 754*.

Table 754. 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 occured
11 EM0	Exception Match 0 This bit indicates the status of the processor exception vector match value of CPU0. 0 No match 1 Match occured
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 754. 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

55.5.1.5 SPU enable register (SE)

The SPU enable/disable register is used to enable and disable the SPU. If disabled, all inputs are masked. *Figure 740* 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 740. SPU enable register (SE)

The SE register fields are described in *Table 755*.

Table 755. 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

55.5.1.6 SLU action unit registers

The action unit of the SPU is responsible for generating the action triggers supported by various actions, listed in *Section 55.2*. 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 756* shows the bit mapping for all the possible actions divided in three groups: client based, counter based, system related.

Table 756. 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	—	—	Valid for Core, AHB and trace group actions	
		00010	Reserved	—	—		
		00011–00111	Reserved	0011	—		
		01000	Reserved	0100	—		
		01001–01111	Reserved	0101	—	—	0= Stop 1= Start
		10000	NXMC1	0110	—		
		10001	NXMC2	0111	—		
		10101–10111	Reserved	1000	—		
		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 756. 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.

55.5.1.6.1 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 756*. 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 741 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 741. State n true action 0 (SnTA0)

The SnTA0 register fields are described in *Table 757*.

Table 757. 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 756*.
20:31 ACTION1	Action 1 to be performed for State n true condition The decoding of these bits is defined in *Table 756*.

55.5.1.6.2 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 756*. 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 742* 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 742. State n true action 1 (SnTA1)

The SnTA1 register fields are described in *Table 758*.

Table 758. 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 756*.
20:31 ACTION3	Action 3 to be performed for State n true condition The decoding of these bits is defined in *Table 756*.

55.5.1.6.3 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 756*. 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 743* 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 743. State n false action 0 (SnFA0)

The SnFA0 register fields are described in *Table 759*.

Table 759. 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 756*.
20:31 ACTION1	Action 1 to be performed for State n false condition The decoding of these bits is defined in *Table 756*.

55.5.1.6.4 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 756*. 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 744* 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 744. State n false action 1 (SnFA1)

The SnFA1 register fields are described in *Table 760*.

Table 760. 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 756*.
20:31 ACTION3	Action 3 to be performed for State n false condition The decoding of these bits is defined in *Table 756*.

55.5.1.6.5 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 745 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 745. Trace group configuration register format

The trace group configuration register fields are described in *Table 761*.

Table 761. Trace group configuration register 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

55.5.1.6.6 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 746* 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 746. Interrupt status (INTS)

The INTS register fields are described in *Table 762*.

Table 762. INTS register field descriptions

Field	Description
31 INT	1 Interrupt status bit is cleared from JTAGM

55.5.1.7 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 747* 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 747. Counters control n registers (CCTRLn)

The CCTRLn register fields are described in *Table 763*.

Table 763. 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.
30 MODE	Mode Select Select counter or timer operation. 0 Counter 1 Timer
31 ENABLE	Enable 0 Counter disabled 1 Counter enabled

55.5.1.8 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 748* 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

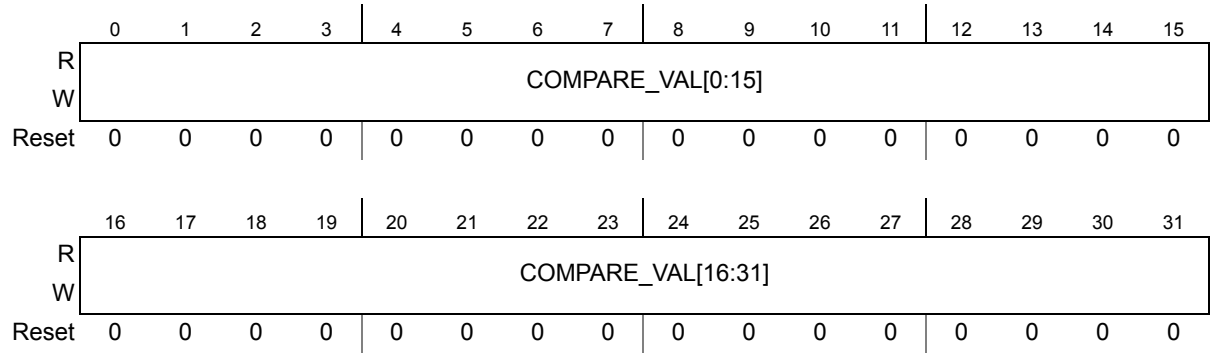


Figure 748. Counters compare n registers (CCMPn)

The CCMPn register fields are described in *Table 764*.

Table 764. 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.

55.5.1.9 Status registers

55.5.1.9.1 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 749* shows the format of the CCOMS register. All of the CC fields in the CCOMS register have the same description as shown in *Table 765*.

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 749. Counter compare status (CCOMS)

The CCOMS register fields are described in *Table 765*.

Table 765. 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 765. 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 ompare 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

55.5.1.9.2 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 750* shows the format of the COS register. All of the CO fields in the COS register have the same description as shown in *Table 766*.

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 750. Counter overflow status (COS)

The COS register fields are described in *Table 766*.

Table 766. COS register field descriptions

Field	Description
16 CO15	Counter 15 Overflow 0 Counter did not overflow 1 Counter overflow occured
17 CO14	Counter 14 Overflow 0 Counter did not overflow 1 Counter overflow occured
18 CO13	Counter 13 Overflow 0 Counter did not overflow 1 Counter overflow occured
19 CO12	Counter 12 Overflow 0 Counter did not overflow 1 Counter overflow occured
20 CO11	Counter 11 Overflow 0 Counter did not overflow 1 Counter overflow occured
21 CO10	Counter10 Overflow 0 Counter did not overflow 1 Counter overflow occured
22 CO9	Counter 9 Overflow 0 Counter did not overflow 1 Counter overflow occured
23 CO8	Counter 8 Overflow 0 Counter did not overflow 1 Counter overflow occured
24 CO7	Counter 7 Overflow 0 Counter did not overflow 1 Counter overflow occured
25 CO6	Counter 6 Overflow 0 Counter did not overflow 1 Counter overflow occured
26 CO5	Counter 5 Overflow 0 Counter did not overflow 1 Counter overflow occured
27 CO4	Counter 4 Overflow 0 Counter did not overflow 1 Counter overflow occured
28 CO3	Counter 3 Overflow 0 Counter did not overflow 1 Counter overflow occured
29 CO2	Counter 2 Overflow 0 Counter did not overflow 1 Counter overflow occured

Table 766. COS register field descriptions(Continued)

Field	Description
30 CO1	Counter 1 Overflow 0 Counter did not overflow 1 Counter overflow occured
31 CO0	Counter 0 Overflow 0 Counter did not overflow 1 Counter overflow occured

55.5.1.9.3 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 751* 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 751. Counter capture status (CCAPS)

The CCAPS register fields are described in *Table 767*.

Table 767. 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 767. 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

55.6 Functional description

55.6.1 Input Mux unit

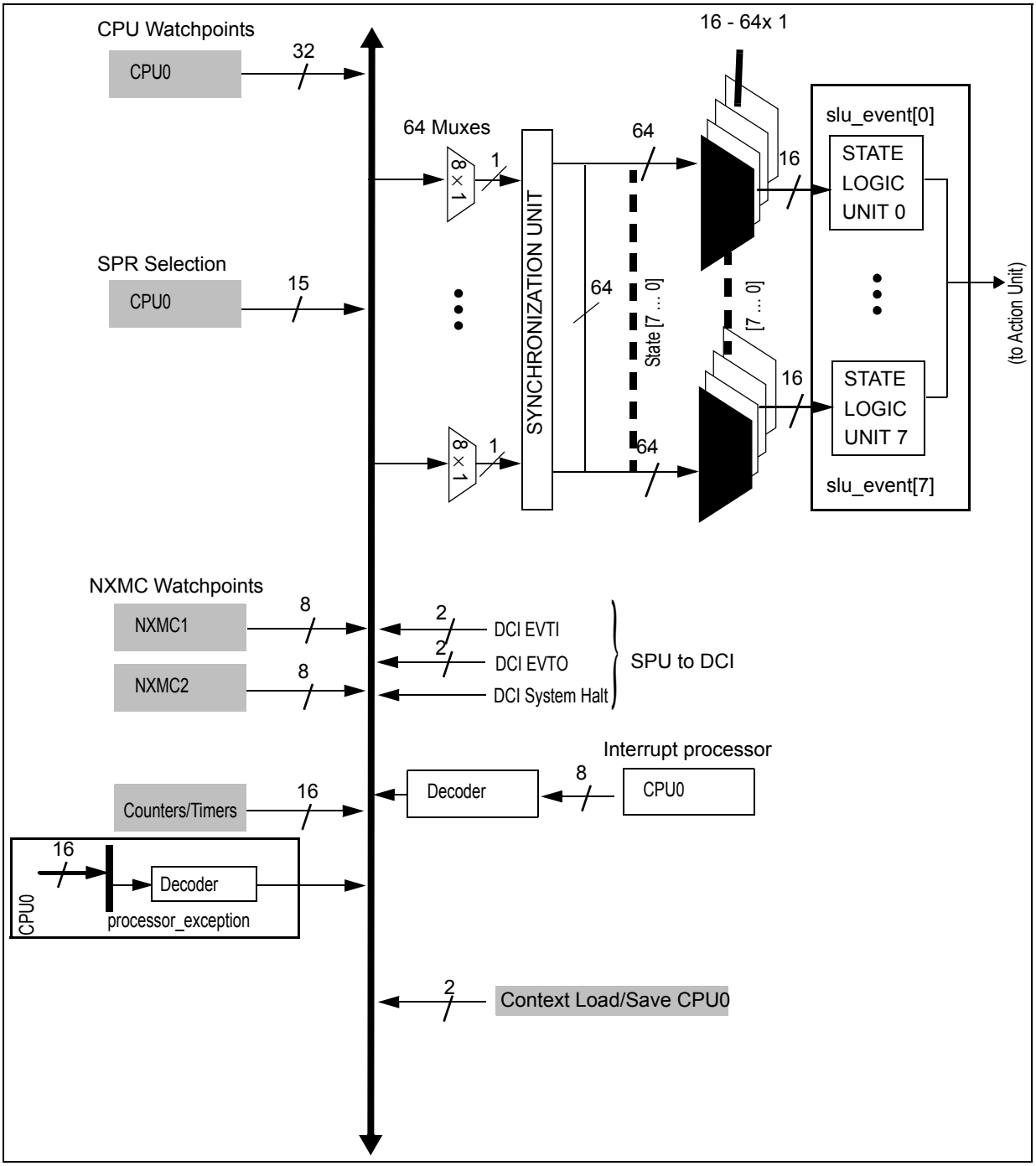
The SPU receives numerous inputs in the form of complex triggers from various clients to create a predefined sequence. One state logic can accommodate 16 inputs at one point of time. Therefore, an input Muxing unit becomes essential for reducing the debug and performance events to a manageable set of 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 752*.

- 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 752*.
- 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 752*.

Figure 752. Input Mux unit block diagram



55.6.2 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 768*. 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 768*.

Table 768. Processor exception processing conditions

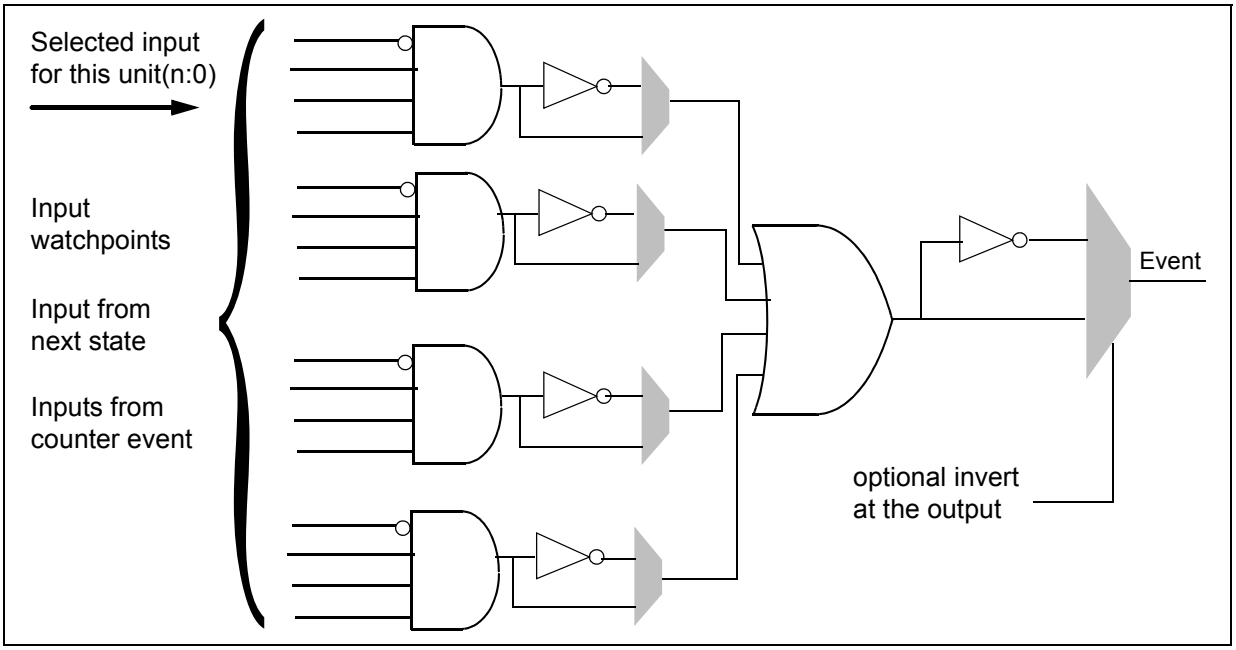
CPU exception vector	CnPEVP register CPUn field value	Exception condition
0x0000	0x0000	Critical Input: autovectored
0x0010	0x1000	Machine check
0x0020	0x2000	Data storage
0x0030	0x3000	Instruction storage
0x0040	0x4000	External input: autovectored
0x0050	0x5000	Alignment
0x0060	0x6000	Program trap
0x0061	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

55.6.3 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 753*.

Figure 753. 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.

55.6.4 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.

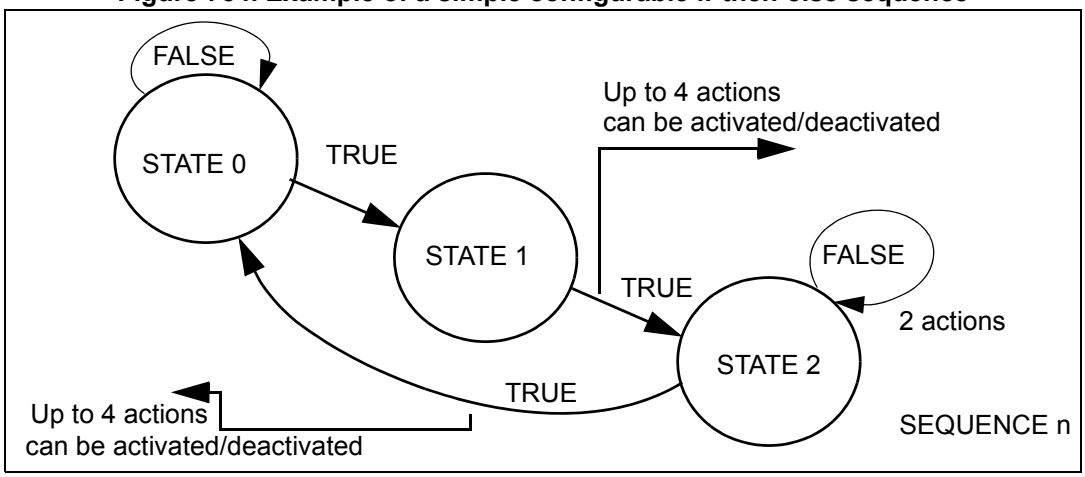
For example, the following state configurations are possible:

- 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 754*:

- 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

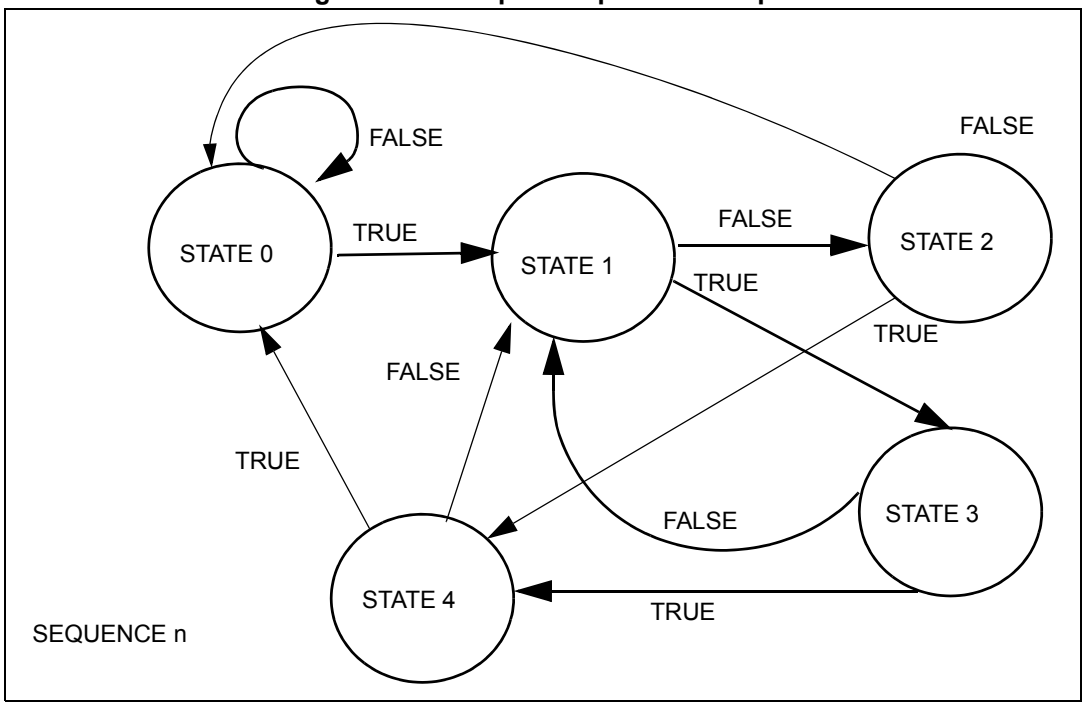
Figure 754. Example of a simple configurable if-then-else sequence



55.6.4.1 Complex sequence example

A user can create a complex sequence using five states as shown in the state transition diagram in *Figure 755*.

Figure 755. Complex sequence example



55.6.4.2 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 occurance 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

55.6.4.3 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:

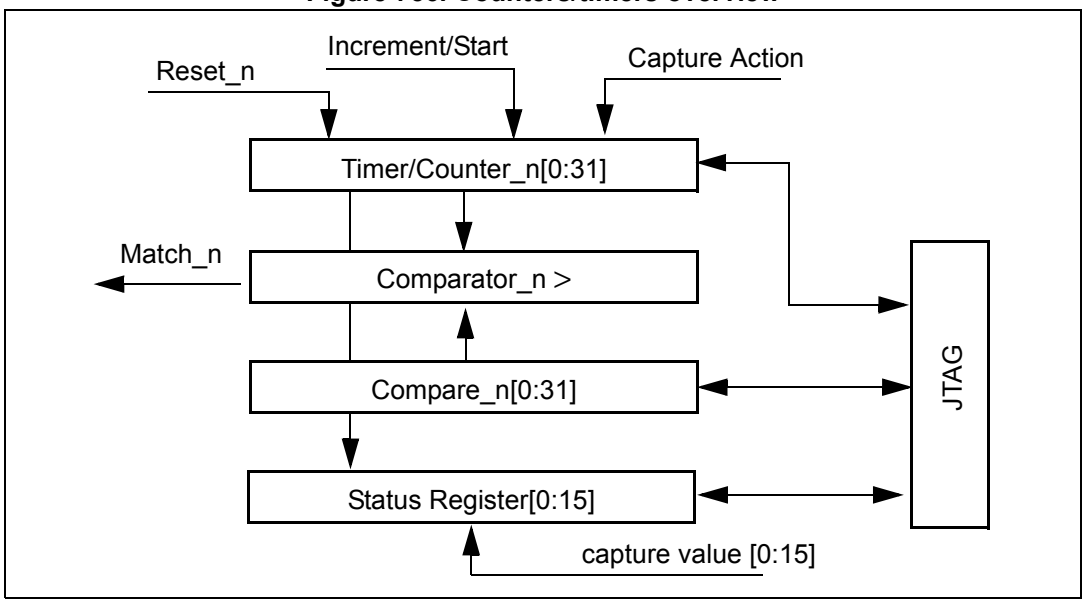
- States: each state has a sticky bit that is set upon a true match of the state logic. This sticky bit is cleared upon a write to this register
- Sequence status: each sequence has a field indicating the active state of the state machine. This is a read only field

55.6.5 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 756*. 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 756. Counters/timers overview



55.6.5.1 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:

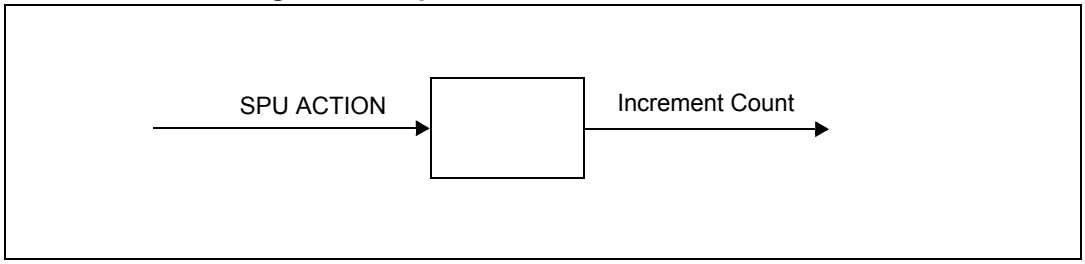
- Actions: each timer/counter has an interface from the action unit to start/increment and reset as appropriate. The start/increment signal is a single signal, with the operation selected in the SPU configuration registers. The generation of the appropriate count/reset signal is generated in the SPU.

  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.

55.6.5.2 Counter input selection at SPU level

There is a main input that is considered to increment the count of a counter. Counter increment inputs derive from the SPU action, as shown in *Figure 757*. These inputs include signals from the primary SPU muxes or SPU events of counters.

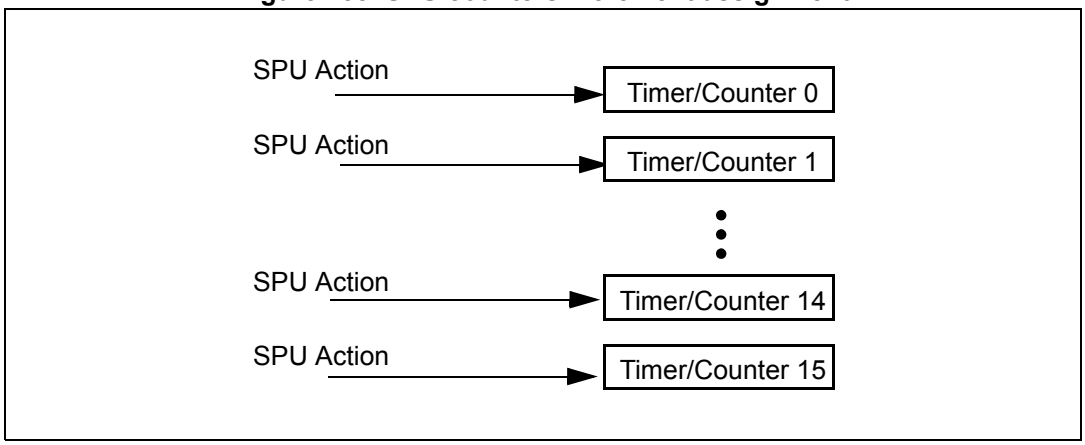
Figure 757. Input selection for count increment



55.6.5.3 Counters increment assignment

When configured as a counter, the counter has an increment signal. This can be routed from the SPU action unit 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 758*.

Figure 758. SPU counters increment assignment



55.6.6 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 55.2*. 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.

55.6.7 Optional trace group

User defined trace group can be defined through the configuration register. Configuration registers, defined in *Table 761*, are available where the user can group the different traces from same or different clients.

56 e200z0h Nexus 2+ Module

The e200z0h Nexus 2+ module provides real-time development capabilities for corresponding core processors in compliance with the IEEE-ISTO 5001-2010 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 5001-2010 standard defines an extensible auxiliary port which is used in conjunction with the JTAG port in these processors.

56.1 Introduction

56.1.1 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 5001-2010 standard. The development features supported are Program Trace, Watchpoint Messaging, Ownership Trace, Data Acquisition Messaging, and Read/Write Access via the JTAG interface. The Nexus 2+ module also supports two Class 4 features: Watchpoint Triggering, and Processor Overrun Control.

56.1.2 Terms and Definitions

Table 769 contains a set of terms and definitions associated with the Nexus 2+ module.

Table 769. 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 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.
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.

Table 769. Terms and Definitions(Continued)

Term	Description
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.

56.1.3 Feature List

The Nexus 2+ module is compliant with Class 2 of the IEEE-ISTO 5001 standard, with additional Class 3 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.
- 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 Trace Messaging
- Data Acquisition Messaging (DQM) allows code to be instrumented to export customized information to the Nexus Auxiliary Output Port.
- External hardware trigger inputs to independently enable/disable Program 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) 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, Ownership Trace and Watchpoint Trigger.
- All features controllable and configurable via the JTAG port

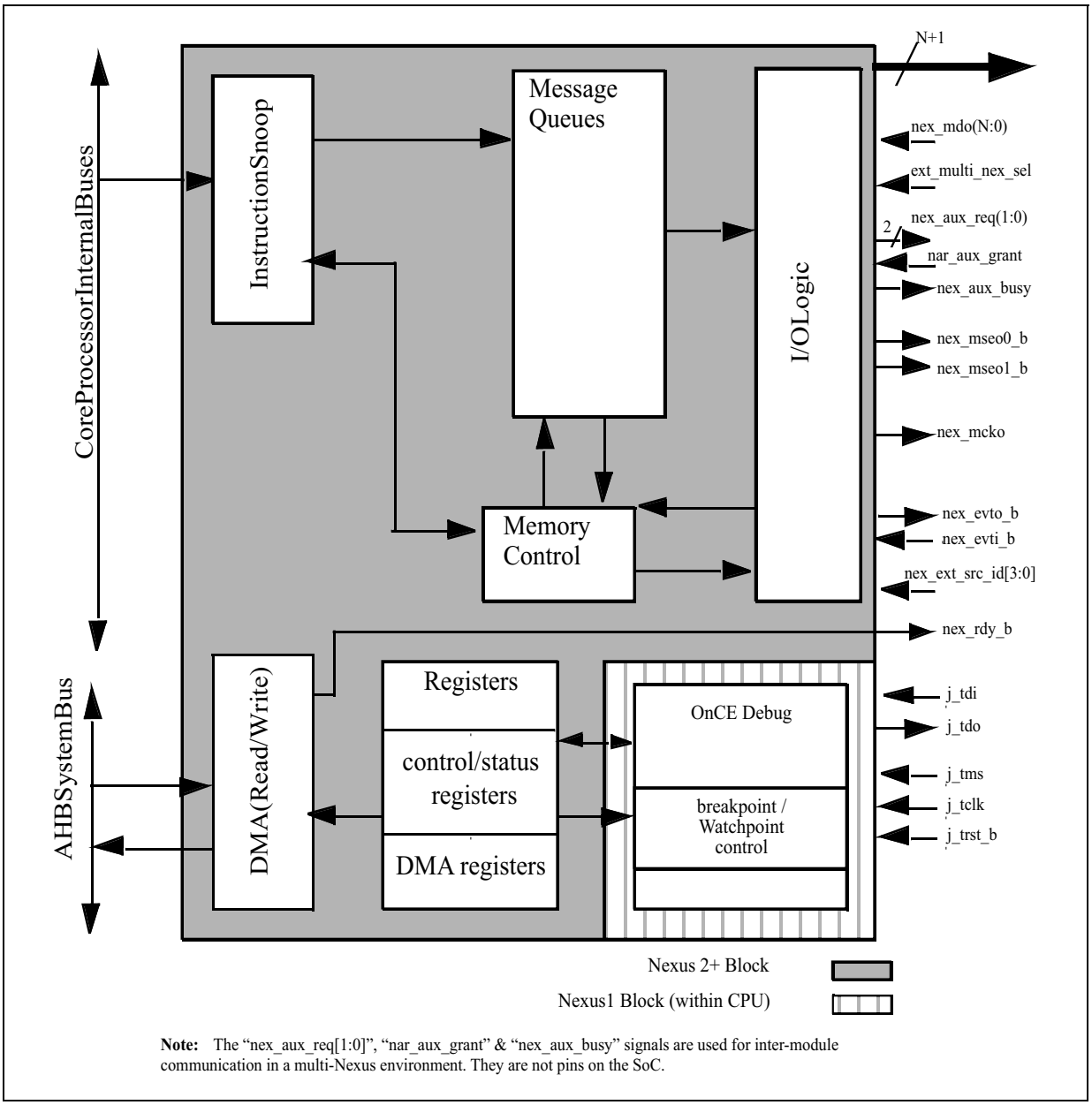
Note: For multi-Nexus implementations, the configuration of the Message Data Out pins is controlled by the Port Control Register (@ the SoC level). For single Nexus implementations (not normally implemented on an SoC), this configuration is controlled by Development Control Register 1 (DC1) within the Nexus 2+ 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.

56.1.4 Functional Block Diagram

Figure 759. Nexus 2+ 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.

56.2 Enabling Nexus 2+ Operation

The Nexus module is enabled by loading a single instruction (*NEXUS2-ACCESS*) into the JTAG Instruction Register (IR) (OnCE OCMD register). For the Nexus2+ 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 2+ module automatically enables the generation of Debug Status Messages.

The Nexus 2+ 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 2+ 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.

56.3 TCODEs supported

The Nexus 2+ 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 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 2+ block supports the TCODEs shown in *Table 770*.

Table 770. 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 taken branch
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

Table 770. Supported TCODEs(Continued)

Message Name	Minimum Field Size (bits)	Maximum Field Size (bits)	Field Name	Field Type	Field Description
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: 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 taken branch
	1	32	F-ADDR	variable	full target address (leading zeros truncated)
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)
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 773*): indicates which resource is the cause of this message
	1	32	RDATA	variable	branch / predicate instruction history (see *Section 56.11.4*.)

Table 770. 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 56.11.1*)
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 56.11.1*.)
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 774*)
	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 56.11.5*.)
	0	32	CDATA2	variable	correlation data field 2- PID info (see *Section 56.11.5*.)

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 56.11.1. If the Branch History method is selected, the shaded TCODES above will not be messaged out.

Table 771 shows the error code encodings used when reporting an error via the Nexus 2+ Error Message.

Table 771. Error Code (ECODE) Encoding (TCODE = 8)

Error Code	Description
xxxxxxx1	Watchpoint Trace Message(s) Lost
xxxxxx1x	Reserved
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 772 shows the error type encodings used when reporting an error via the Nexus 2+ Error Message.

Table 772. 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	Read/write access error
0100	Reserved
0101	Invalid access opcode (Nexus Register unimplemented)
0110-1111	Reserved

Table 773 shows the encodings used for resource codes for certain messages.

Table 773. 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 774 shows the event code encodings used for certain messages.

Table 774. 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
0101-1001	Reserved for future functionality
1010	Branch and link occurrence (direct branch function call)(1)
1011-1111	Reserved for future functionality

1. Only used for Program Trace - History Mode

56.4 Nexus 2+ Programmer's Model

This section describes the Nexus 2+ programmers model. Nexus 2+ registers are accessed using the JTAG/OnCE port in compliance with IEEE 1149.1. See *Section 56.5* for details on Nexus 2+ register access.

Note: Nexus 2+ 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 5001-2010 standard.

Table 775 details the register map for the Nexus 2+ module.

Table 775. Nexus 2+ Register Map

Nexus Register	Nexus Access Opcode	Read/ Write	Read Address	Write Address
Client Select Control (CSC)(1)	0x1	R	0x02	—
Port Configuration Register (PCR)(2)	PCR_INDEX(1)	R/W	—	—
Development Control 1 (DC1)	0x2	R/W	0x04	0x05
Development Control 2 (DC2)	0x3	R/W	0x06	0x07
Development Control 3 (DC3)	0x4	R/W	0x08	0x09
Development Control 4 (DC4)	0x5	R/W	0x0A	0x0B
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
Reserved	0xC-0x2F	-	0x18-0x5E	0x19-0x5F
Development Status (DS)	0x30	R	0x60	—

Table 775. Nexus 2+ Register Map(Continued)

Nexus Register	Nexus Access Opcode	Read/ Write	Read Address	Write Address
Reserved	0x31	R/W	0x62	0x63
Overrun Control (OVCR)	0x32	R/W	0x64	0x65
Watchpoint Mask (WMSK)	0x33	R/W	0x66	0x67
Reserved	0x34	-	0x68	0x69
Program Trace Start Trigger Control (PTSTC)	0x35	R/W	0x6A	0x6B
Program Trace End Trigger Control (PTETC)	0x36	R/W	0x6C	0x6D
Reserved	0x37-0x3F	—	0x6E-0x7E	0x6F-0x7F

1. 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 2+ module. The SoC's CSC Register is readable through Nexus, but the PCR is shown for reference only here.
2. The "PCR_INDEX" is a parameter determined by the SoC.

56.4.1 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 2+ controller to select one of multiple on-chip Nexus 2+ units.

Offset: Refer to *Table 775* 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 760. Client Select Control register (CSC)

Table 776. CSC Register field descriptions

Field	Description
4:7 CS	Client Select Control xxxx Nexus client (SoC level)

56.4.2 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.

Offset: Refer to *Table 775* 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 761. Port Configuration Register (PCR)

Table 777. 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 2+ module is the only Nexus module, these registers are not implemented and the Nexus 2+ defined Development Control Register 1 (DC1) is used to control the SoC-level Nexus port functionality.

56.4.3 Nexus Development Control Register 1 (DC1)

Nexus Development Control Register 1 is used to control the basic development features of the Nexus 2+ module. Development Control Register 1 is shown in *Figure 762* and its fields are described in *Table 778*.

Offset: Refer to *Table 775* 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	0	0	0	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 762. Development Control register 1 (DC1)

Table 778. 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
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

Table 778. DC1 Register field descritions(Continued)

Field	Description
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 Reserved 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 56.4.7* for more information on watchpoint triggering. Refer to *Section 56.4.7* 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.

56.4.4 Nexus Development Control Registers 2-3 (DC2-3)

Nexus Development Control Registers 2 and 3 are used to control output signaling on the Nexus 2+ module. A table of watchpoints can be found in the Core (e200z0h) Core Debug Support chapter.

Development Control Register 2 is shown in *Figure 763* and its fields are described in *Table 779*.

Offset: Refer to *Table 775* 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 763. Development Control register 2 (DC2)

Table 779. 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 764* and its fields are described in *Table 780*.

Offset: Refer to *Table 775* 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	0	0	0	0	0	0	0	0	EWC					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 764. Development Control register 3 (DC3)

Table 780. DC3 Register field descriptions

Field	Description
26:31 EWC	EVTO Watchpoint Configuration(1) 00000000000000 No Watchpoints #16-#29 trigger **nex_evto_b** xxxxx1 Watchpoint #16 triggers **nex_evto_b** xxxx1x Watchpoint #17 triggers **nex_evto_b** xxx1xx Watchpoint #18 triggers **nex_evto_b** xx1xxx Watchpoint #19 triggers **nex_evto_b** x1xxxx Watchpoint #20 triggers **nex_evto_b** 1xxxxx Watchpoint #21 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.

56.4.5 Nexus Development Control Register 4 (DC4)

Nexus Development Control Register 4 is used to control mark selection for Program Messaging, as well as masking of events which initiate Program Correlation Messages on the Nexus 2+ module.

Development Control Register 4 is shown in *Figure 765* and its fields are described in *Table 781*.

Offset: Refer to *Table 775* Access: Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	PTMARK	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	EVCDM															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 765. Development Control register 4 (DC4)

Table 781. 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'
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 774* for implemented EVCODEs.

56.4.6 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 775* Access: Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DBG	LPS			LPC		CHK	0	0	0	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 766. Development Status Register (DS)

Table 782. 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)
6 CHK	CHK Zen CPU Checkstop Status 0 = CPU not in Checkstop state 1 = CPU in Checkstop state (p_chkstop signal asserted)

56.4.7 Watchpoint Trigger Registers (WT, PTSTC, PTETC)

The Watchpoint Trigger Registers allows the watchpoints defined within the Nexus1 logic to trigger actions. These watchpoints can control Program 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) 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 in DC1 will override these controls, and trace will remain enabled until another direct write to DC1 to disable program 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 Debug Support chapter.

Offset: Refer to *Table 775* Access: Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	PTS				PTE				0	0	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 767. Watchpoint Trigger register (WT)

Table 783 details the Watchpoint Trigger register fields.

Table 783. 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

For extended Program Trace start trigger control, the PTSTC register is used.

Offset: Refer to *Table 775* 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	PTST[21:16]					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	PTST[15:0]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 768. Program Trace Start Trigger Control register (PTSTC)

Table 784 details the PTSTC register fields.

Table 784. PTSTC Register field descriptions

Field	Description
10:31 PTST	Program Trace Start Trigger Control 00000000000000000000 Trigger disabled xxxxxxxxxxxxxxxxxxx1 Use Watchpoint #0 xxxxxxxxxxxxxxxxxx1x Use Watchpoint #1 xxxxxxxxxxxxxxxxx1xx Use Watchpoint #2 xxxxxxxxxxxxxxxx1xxx Use Watchpoint #3 xxxxxxxxxxxxxxx1xxxx Use Watchpoint #4 xxxxxxxxxxxxxx1xxxxx Use Watchpoint #5 xxxxxxxxxxxxx1xxxxxx Use Watchpoint #6 xxxxxxxxxxxx1xxxxxxx Use Watchpoint #7 xxxxxxxxxxx1xxxxxxxx Use Watchpoint #8 xxxxxxxxxx1xxxxxxxxx Use Watchpoint #9 xxxxxxxxx1xxxxxxxxx Use Watchpoint #10 xxxxxxxx1xxxxxxxxxx Use Watchpoint #11 xxxxxxx1xxxxxxxxxxx Use Watchpoint #12 xxxxxx1xxxxxxxxxxxx Use Watchpoint #13 xxxxx1xxxxxxxxxxxxx Use Watchpoint #14 xxxx1xxxxxxxxxxxxxx Use Watchpoint #15 xxx1xxxxxxxxxxxxxxx Use Watchpoint #16 xx1xxxxxxxxxxxxxxxx Use Watchpoint #17 x1xxxxxxxxxxxxxxxxx Use Watchpoint #18 1xxxxxxxxxxxxxxxxxx Use Watchpoint #19 Use Watchpoint #20 Use Watchpoint #21

For extended Program Trace end trigger control, the PTETC register is used.

Offset: Refer to *Table 775* 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	PTET[21:16]					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	PTET[15:0]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 769. Program Trace End Trigger Control register (PTETC)

Table 785 details the PTETC register fields.

Table 785. PTETC Register field descriptions

Field	Description
10:31 PTET	Program Trace End Trigger Control 00000000000000000000 Trigger disabled xxxxxxxxxxxxxxxxxxx1 Use Watchpoint #0 xxxxxxxxxxxxxxxxxx1x Use Watchpoint #1 xxxxxxxxxxxxxxxxx1xx Use Watchpoint #2 xxxxxxxxxxxxxxxx1xxx Use Watchpoint #3 xxxxxxxxxxxxxxx1xxxx Use Watchpoint #4 xxxxxxxxxxxxxx1xxxxx Use Watchpoint #5 xxxxxxxxxxxxx1xxxxxx Use Watchpoint #6 xxxxxxxxxxxx1xxxxxxx Use Watchpoint #7 xxxxxxxxxxx1xxxxxxxx Use Watchpoint #8 xxxxxxxxxx1xxxxxxxx Use Watchpoint #9 xxxxxxxxx1xxxxxxxxx Use Watchpoint #10 xxxxxxxx1xxxxxxxxxx Use Watchpoint #11 xxxxxxx1xxxxxxxxxxx Use Watchpoint #12 xxxxxx1xxxxxxxxxxxx Use Watchpoint #13 xxxxx1xxxxxxxxxxxxx Use Watchpoint #14 xxxx1xxxxxxxxxxxxxx Use Watchpoint #15 xxx1xxxxxxxxxxxxxxx Use Watchpoint #16 xx1xxxxxxxxxxxxxxxx Use Watchpoint #17 x1xxxxxxxxxxxxxxxxx Use Watchpoint #18 1xxxxxxxxxxxxxxxxxx Use Watchpoint #19

56.4.8 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 775* 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	WEM[21:16]					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	WEM[15:0]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 770. Watchpoint Mask register (WMSK)

Table 786 details the Watchpoint Trigger register fields.

enabled for WTM

Table 786. WMSK Register field descriptions

Field	Description
10:31 WEM	Watchpoint Enable for Messaging 00000000000000000000 No Watchpoints enabled for Watchpoint Trace Messaging xxxxxxxxxxxxxxxxxxx1 Watchpoint #0 enabled for WTM xxxxxxxxxxxxxxxxxxx1x Watchpoint #1 enabled for WTM xxxxxxxxxxxxxxxxxx1xx Watchpoint #2 enabled for WTM xxxxxxxxxxxxxxxxx1xxx Watchpoint #3 enabled for WTM xxxxxxxxxxxxxxxx1xxxx Watchpoint #4 enabled for WTM xxxxxxxxxxxxxxx1xxxxx Watchpoint #5 enabled for WTM xxxxxxxxxxxxxx1xxxxxx Watchpoint #6 enabled for WTM xxxxxxxxxxxxx1xxxxxxx Watchpoint #7 enabled for WTM xxxxxxxxxxxx1xxxxxxxx Watchpoint #8 enabled for WTM xxxxxxxxxxx1xxxxxxxxx Watchpoint #9 enabled for WTM xxxxxxxxxx1xxxxxxxxxx Watchpoint #10 enabled for WTM xxxxxxxxx1xxxxxxxxxxx Watchpoint #11 enabled for WTM xxxxxxxx1xxxxxxxxxxx Watchpoint #12 enabled for WTM xxxxxxx1xxxxxxxxxxx Watchpoint #13 enabled for WTM xxxxxxx1xxxxxxxxxxx Watchpoint #14 enabled for WTM xxxxxx1xxxxxxxxxxxx Watchpoint #15 enabled for WTM xxxxx1xxxxxxxxxxxx Watchpoint #16 enabled for WTM xxxx1xxxxxxxxxxxxx Watchpoint #17 enabled for WTM xxx1xxxxxxxxxxxxx Watchpoint #18 enabled for WTM xx1xxxxxxxxxxxxx Watchpoint #19 enabled for WTM x1xxxxxxxxxxxxx Watchpoint #20 enabled for WTM 1xxxxxxxxxxxxx Watchpoint #21 enabled for WTM

56.4.9 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 775* 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 771. Nexus Overrun Control Register (OVCR)

Table 787. 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 Reserved 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

56.4.10 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 789*.

Offset: Refer to *Table 775* 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		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	CNT														ERR	DV
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 772. Read/Write Access Control/Status register (RWCS)

Table 788. 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 Reserved 100 Reserved (default to word) 101 Reserved (default to word) 110 Reserved (default to word) 111 Reserved (default to word)
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

Table 788. RWCS register field descriptions(Continued)

Field	Description
16:29 CNT	Access Control Count 0xXXXX Number of accesses of word size SZ
30 ERR	Read/Write Access Error (see *Table 789*)
31 DV	Read/Write Access Data Valid (see *Table 789*)

1. The priority functionality is not currently implemented.

Table 789. 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

56.4.11 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 775* 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 773. 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 790* shows the proper placement of data into the RWD. Note that doubleword transfers require two passes through RWD.

Table 790. 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

1. "X" indicates byte lanes with valid data
"—" indicates byte lanes which will contain unused data.

Table 791 shows the mapping of RWD bytes to byte lanes of the AHB read and write data buses.

Table 791. 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]
Half @000	0 0 0	—	—	AHB[15:8]	AHB[7:0]
Half @010	0 1 0	—	—	AHB[31:24]	AHB[23:16]
Word @000	0 0 0	AHB[31:24]	AHB[23:16]	AHB[15:8]	AHB[7:0]

1. "-" indicates byte lanes which will contain unused data.

56.4.12 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 775* 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 774. Read/Write Access Address register (RWA)

56.5 Nexus 2+ Register Access via JTAG/OnCE

Access to Nexus 2+ register resources is enabled by loading a single instruction ("*NEXUS2-ACCESS*") into the JTAG Instruction Register (IR) (OnCE OCMD register). For the Nexus 2+ block, the OCMD value is 0b0001111100.

Once the NEXUS2-ACCESS instruction has been loaded, the JTAG/OnCE port allows tool/target communications with all Nexus 2+ registers according to the register map in *Table 775*.

Reading/writing of a Nexus 2+ register then requires two (2) passes through the Data-Scan (DR) path of the JTAG state machine. (See *Section 56.20*.)

1. The first pass through the DR selects the Nexus 2+ register to be accessed by providing an index (see *Table 775*), 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 775. JTAG DR format

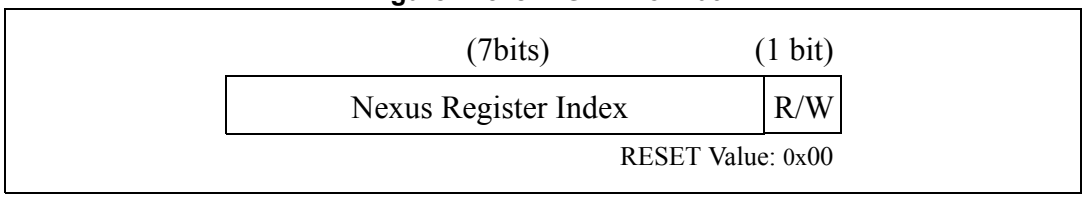


Table 792. JTAG DR field descriptions

Nexus Register Index	Selected from values in *Table 775*
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.

56.6 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 sub-sections describe a subset of the message field types.

56.6.1 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 5001-2010.

56.6.2 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 2+ module implements a 4-bit fixed length Source ID field consisting of a Client Source ID.

56.6.3 Relative Address Field (U-ADDR)

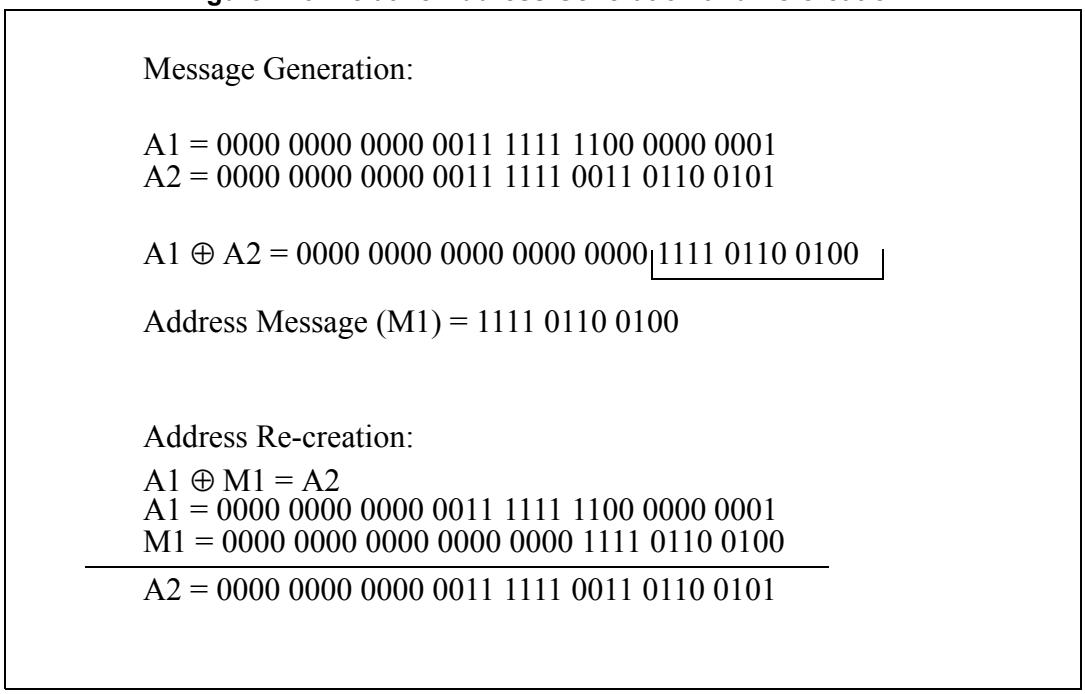
The non-sync forms of the Program Trace messages include addresses which are relative to the address that was transmitted in the previous Program Trace message respectively. The relative address format is compliant with IEEE-ISTO 5001-2010 and is designed to reduce the number of bits transmitted for 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.

Previous Address (A1) =0x0003FC01, New Address (A2) = 0x0003F365

Figure 776. Relative Address Generation and Re-creation



56.6.4 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.

56.6.5 Address Space Indication Field (MAP)

Indirect-type program trace messages provide the address space status (IS value) in the address space (MAP) field. For Program Trace, the MAP field is used to indicate the future space used for instruction execution (new value of MSRIS). A change in instruction address space will only occur on reset, on an exception, or via a mtmsr, se_rfi, se_rfci, or se_rfdi instruction. A potential change in address space via an exception or via an se_rfi, se_rfci, or se_rfdi instruction will cause a program trace indirect branch message to be generated indicating the new address space (IS) value, along with ICNT and HIST information for instructions executed up to the change (including the se_rfi, se_rfci, or se_rfdi). A change in address space via a mtmsr instruction will cause a program correlation message to be generated indicating the new address space (IS) value, along with ICNT and HIST information for instructions executed prior to the change (including the mtmsr).

56.7 Nexus Message Queues

The Nexus 2+ 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.

56.7.1 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.

56.7.2 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 56.4.9* for complete programming options.

56.7.3 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 56.7.1*. 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.

56.7.4 Nexus Message Priority

Nexus messages may be lost due to contention with other message types under the following circumstances:

- More than two messages are generated in the same cycle

Table 793 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 793*.

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 -> OTM -> BTM)-> Instruction1 (newer instruction) (WPM -> DQM -> OTM -> BTM). Up to two messages may be simultaneously queued. Note that for the cycle following a dropped PTM, non-periodic OTM, or DQM message, only one other message 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 793. 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)

Table 793. Message Type Priority and Message Dropped Responses(Continued)

Message Type	Message	Priority	Message Dropped Response
DQ(Data Acquisition)	DQM (Data Acquisition Message)	2	DQM Error Message
OT(Ownership)	OTM: PID/NPIDR update (Ownership Trace Message)	2	OTM Error Message
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
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.

56.7.5 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.

56.7.6 Ownership Trace Message Priority Loss Response

If an Ownership Trace message (OTM) loses arbitration due to contention with higher priority messages, an error message will be generated to indicate that a OTM has been lost due to contention.

56.7.7 Program Trace Message Priority Loss Response

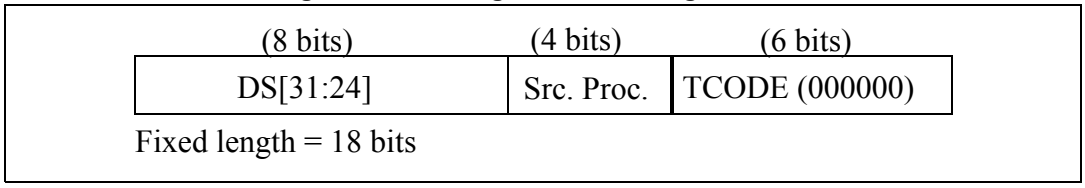
If a Program Trace message (PTM) loses arbitration due to contention with higher priority messages, and the discarded PTM is a Program Correlation message, a Resource Full message for instruction count or history buffer, or a Branch Trace message, then an Error message is generated to indicate that branch trace information has been lost, and the next Branch Trace message will be upgraded to a sync-type message.

56.8 Debug Status Messages

Debug Status Messages report low power mode and debug status. Debug Status Messages are enabled when Nexus 2+ 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 777. Debug Status Message Format



56.9 Error Messages

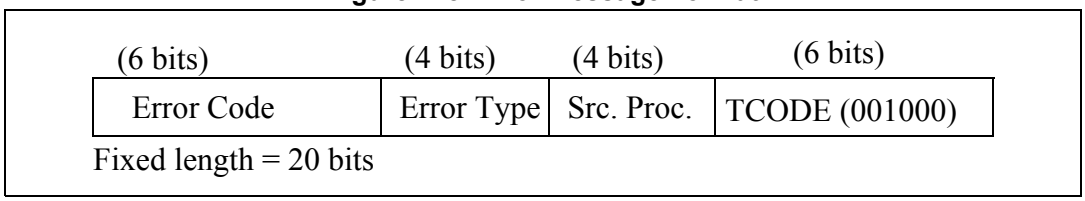
Error messages are enabled whenever the debug logic is enabled. There are two conditions that will produce an error message, each receiving a separate error type designation:

- 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 771* and *Table 772*):

Figure 778. Error Message Format



56.10 Ownership Trace

This section details the ownership trace features of the Nexus 2+ module.

56.10.1 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.

56.10.2 Ownership Trace Messaging (OTM)

Ownership trace information is messaged via the auxiliary port using an Ownership Trace Message (OTM). Core (e200z0h) processors contain a *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.

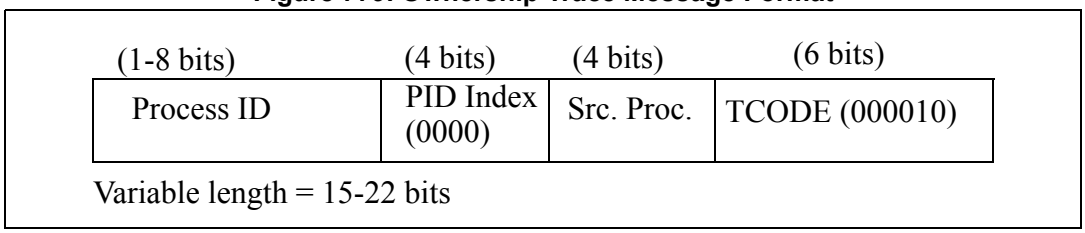
Note: The CPU includes a Process ID register (PID0), thus the Nexus UBA functionality is not implemented.

There are two conditions which will cause an Ownership Trace Message when Ownership Trace is enabled:

- When new information is updated in the Process ID (PID0) register by the Zen processor, the data is latched within Nexus, and is messaged out via the auxiliary port, allowing development tools to trace ownership flow.
- 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 sub-field that PID0 status is being reported and the most recent value of the PID0 register will be conveyed in the Process ID value sub-field. These periodic Ownership Trace message events can be disabled by setting $DC1_{POTD}$.

Ownership trace information is messaged out in the following format:

Figure 779. Ownership Trace Message Format



56.11 Program Trace

This section details the program trace mechanism supported by Nexus2+ for the e200z0h processor. Program trace is implemented via Branch Trace Messaging (BTM) as per the IEEE-ISTO 5001 standard definition. Branch Trace Messaging for Core (e200z0h) processors is accomplished by snooping the internal address bus (between the CPU and MMU), attribute signals, and CPU Status (**p_pstat[0:5]**).

56.11.1 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. Messaging for taken indirect branches and exceptions also include the newly established value of the MSRIS bit in the MAP field if the indirect branch message is due to an exception or **se_rfi, se_rfci, or se_rfdi** class instruction. For all other indirect branches, the MAP field will reflect the current value of MSR_{IS}.

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. Messaging for taken indirect branches and exceptions also include the newly established value of the MSRIS bit in the MAP field if the indirect branch message is due to an exception or **se_rfi, se_rfci, or se_rfdi class** instruction. For all other indirect branches, the MAP field will reflect the current value of MSR_{IS}.

56.11.1.1 Indirect Branch Message Instructions

Table 794 shows the types of instructions and events which cause Indirect Branch Messages or Branch History Messages to be encoded.

Table 794. Indirect Branch Message Sources

Source of Indirect Branch Message	Instructions / Detail
Taken branch relative to a register value	**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

56.11.1.2 Direct Branch Message Instructions

Table 795 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 795. Direct Branch Message Sources

Source of Direct Branch Message	Instructions
Taken direct branch instructions Instruction Synchronize	**se_b. se_bc, se_bl, e_b, e_bc, e_bl, e_bcl, se_isync**

56.11.1.3 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.

56.11.1.4 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.

56.11.2 BTM Message Formats

The Nexus 2+ 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.

56.11.2.1 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 780. 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

56.11.2.2 Indirect Branch Messages (Traditional)

If $DC1_{PTM}$ is cleared to ‘0’, indirect branch information is messaged out in the following format:

Figure 781. Indirect Branch Message Format

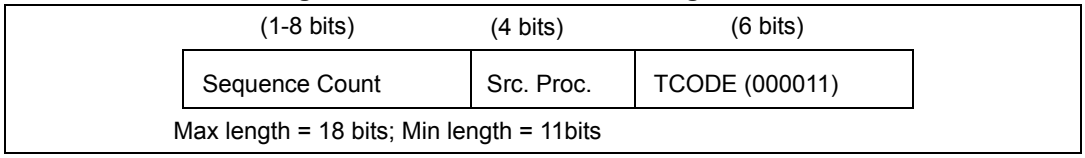
(1-32 bits)	(1-8 bits)	(1 bit)	(4 bits)	(6 bits)
Relative Address	Sequence Count	(0)	Source Proc.	TCODE (000100)

Max length = 51 bits; Min length = 13 bits

56.11.2.3 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 782. Direct Branch Message Format



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.

56.11.3 Program Trace Message Fields

The following subsections describe specific fields used for Program Trace messages.

56.11.3.1 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.

56.11.3.2 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 796. Branch/Predicate History Events

Branch/Predicate History Event	History Bit(s)	Relevant Instructions
Not taken register indirect branches	0	**se_bctr, se_bctrl, se_blr, se_blrl**
Not taken direct branches	0	**se_b. se_bc, se_bl, e_b, e_bc, e_bl, e_bcl**
Taken direct branches	1	**se_b. se_bc, se_bl, e_b, e_bc, e_bl, e_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.

56.11.3.3 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. Note that when a branch results in an execution mode switch, the program trace message resulting from that branch will indicate the previous execution state. The new state will not be signaled until the next program trace message. Note that for Zen z0Hn2p, no execution mode switch will occur, since all instructions are executed in a VLE context.

In some cases, a Program Correlation Message is generated to indicate execution mode status. Refer to *Section 56.11.5: Program Correlation Messages* for more information on these cases.

56.11.4 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 783. Resource Full Message Format

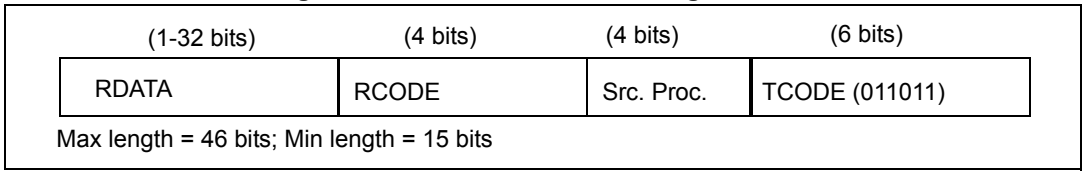


Table 797 shows the RCODE encodings and RDATA information used for Resource Full messages.

Table 797. 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.

56.11.5 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)

Refer to *Table 774* 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 784. Program Correlation Message Formats

(1-32 bits)	(1-8 bits)	(2 bits)	(4 bits)	(4 bits)	(6 bits)
Branch History	Sequence Count	CDF*	EVCODE	Src. Proc.	TCODE (100001)

Max length = 56 bits; Min length = 18 bits

* - CDF=01,

56.11.6 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.

56.11.7 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 5001-2010.

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 798*):

- Initial Program Trace Message upon the first direct/indirect branch after exit from system reset or whenever program trace is enabled.
- Upon direct/indirect branch after returning from a CPU Low Power state.
- Upon direct/indirect branch after returning from Debug Mode.
- Upon direct/indirect branch after occurrence of queue overrun (can be caused by any trace message), provided Program Trace is enabled.
- Upon direct/indirect branch after the periodic program trace counter has expired indicating 255 without-sync Program Trace Messages have occurred since the last with-sync message occurred.
- Upon direct/indirect branch after assertion of the Event In (nex_evti_b) pin if the EIC bits within the DC1 Register have enabled this feature.
- Upon direct/indirect branch after the sequential instruction counter has expired indicating 255 instructions have occurred since the last change of flow.
- Upon direct/indirect branch after a BTM Message was lost due to a collision while attempting to enter the message queue.

Note that the ICNT and History information for the first message will not be meaningful for some of these cases, 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 785. Direct/Indirect Branch with Sync Message Format

(1-32 bits)	(1-8 bits)	(1 bit)	(4 bits)	(6 bits)
Full Target Address	Sequence Count	Inst Space	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 786. 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	Inst Space	Source Proc.	TCODE (011101)

Max length = 83 bits; Min length = 14 bits

Exception conditions that result in Program Trace Synchronization are summarized in *Table 798*.

Table 798. 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 2+ 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.
Periodic Program Trace Synchronization	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.
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.

Table 798. Program Trace Exception Summary(Continued)

Exception Condition	Exception Handling
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: WPM -> OTM -> BTM. A BTM Message which attempts to enter the queue at the same time as a Watchpoint Message and Ownership Trace Message 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.

56.11.8 Enabling Program Trace

Program Trace Messaging can be enabled in one of two 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 (Zen watchpoints are configured within the CPU)

56.11.9 Program Trace Timing Diagrams (2 MDO / 1 MSEO configuration)

Figure 787. Program Trace - Indirect Branch Message (Traditional)

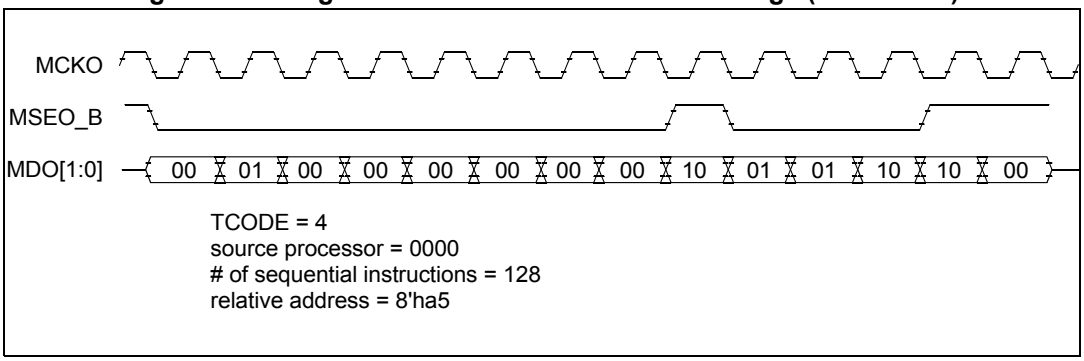


Figure 788. Program Trace - Indirect Branch Message (History)

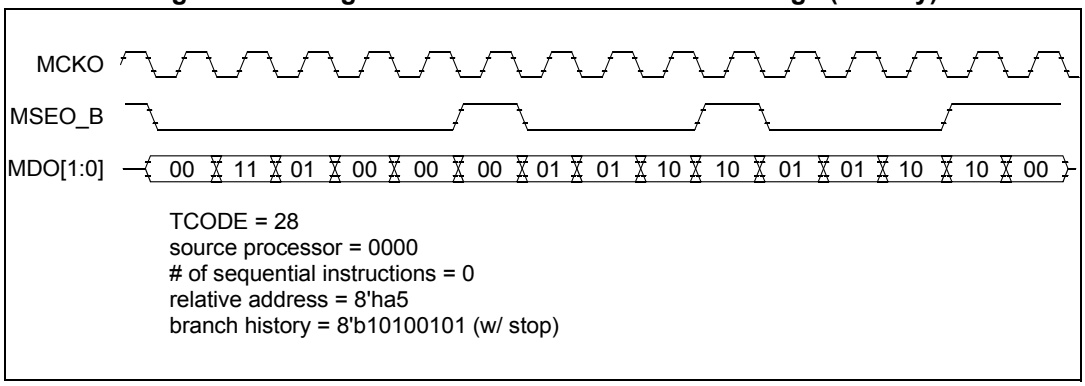


Figure 789. Program Trace - Direct Branch (Traditional) & Error Messages

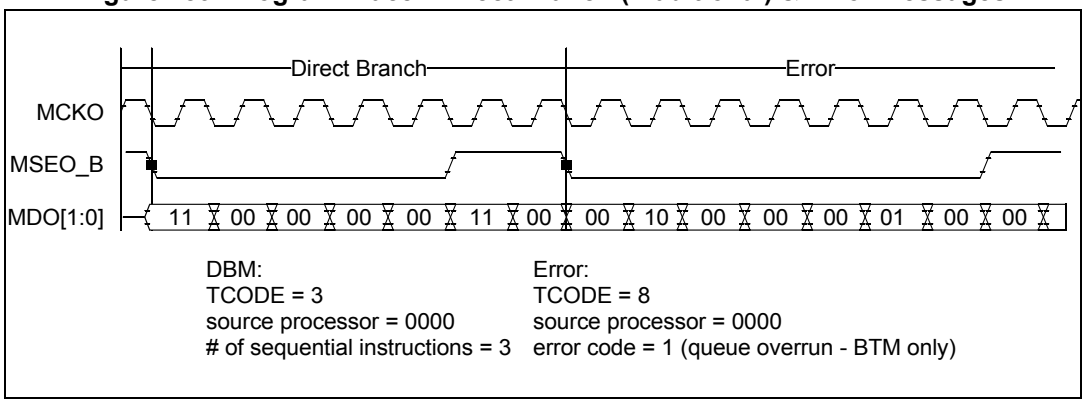
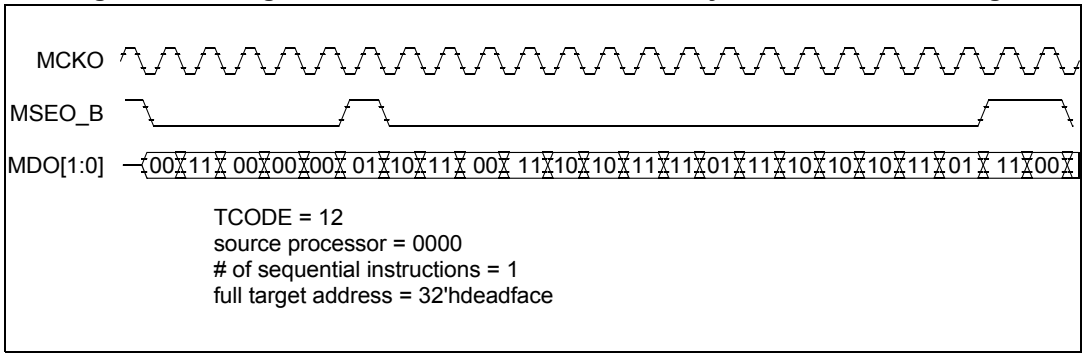


Figure 790. Program Trace - Indirect Branch with Synchronization Message



56.12 Data Acquisition Messaging

This section details the Data Acquisition mechanisms supported by the Nexus 2+ module. Data Acquisition Trace is implemented using Data Acquisition Trace Messages in accordance with IEEE-ISTO 5001-2010 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.

56.12.1 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.

56.12.2 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.

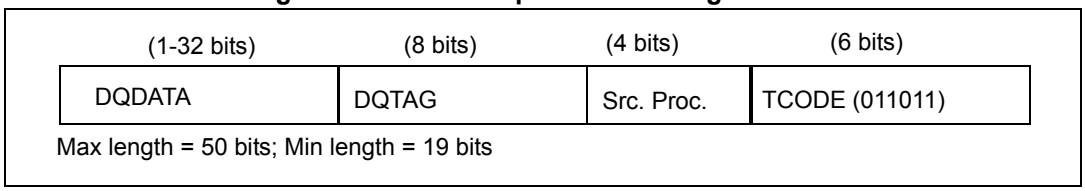
56.12.3 Data Acquisition Trace event

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 (e200z0h) 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 791. .Data Acquisition Message Format



56.13 Watchpoint Trace Messaging

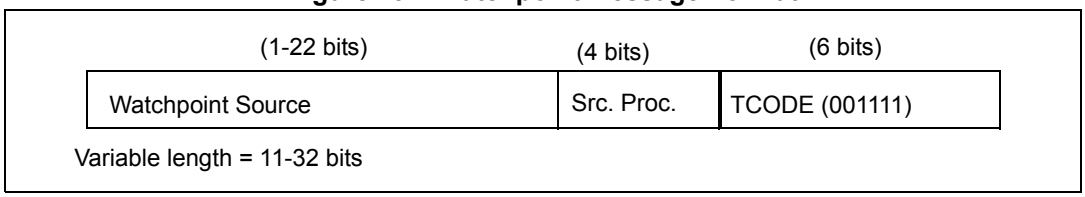
Enabling Watchpoint Messaging is done by setting the Watchpoint Trace Enable bit in the DC1 Register. Setting the individual Watchpoint sources is supported through the Nexus1 module and the Performance Monitor unit. The Nexus1 module is capable of setting multiple types of watchpoints. Please refer to the Core 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 Zen 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 801* for details on **nex_evto_b**.

Watchpoint information is messaged out in the following format:

Figure 792. Watchpoint Message Format



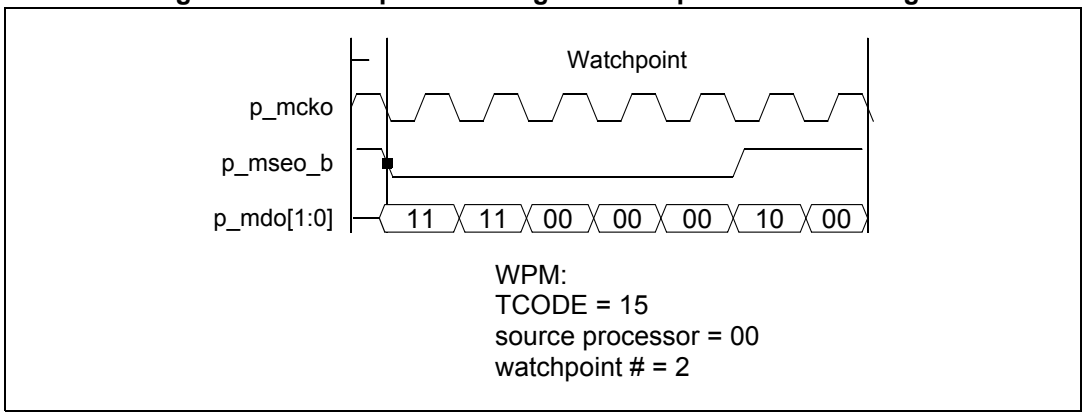
The Watchpoint Source message field will contain a '1' for each asserted watchpoint. Leading zeros are truncated.

Table 799. Watchpoint Source Encoding

Watchpoint Source (1:22 bits)	Watchpoint Description
0000000000000000000000	No Watchpoints enabled for Watchpoint Trace Messaging
xxxxxxxxxxxxxxxxxxxxx1	Watchpoint #0 enabled for WTM
xxxxxxxxxxxxxxxxxxxx1x	Watchpoint #1 enabled for WTM
xxxxxxxxxxxxxxxxxxx1xx	Watchpoint #2 enabled for WTM
xxxxxxxxxxxxxxxxxx1xxx	Watchpoint #3 enabled for WTM
xxxxxxxxxxxxxxxxx1xxxx	Watchpoint #4 enabled for WTM
xxxxxxxxxxxxxxxx1xxxxx	Watchpoint #5 enabled for WTM
xxxxxxxxxxxxxxx1xxxxxx	Watchpoint #6 enabled for WTM
xxxxxxxxxxxxxx1xxxxxxx	Watchpoint #7 enabled for WTM
xxxxxxxxxxxxx1xxxxxxxx	Watchpoint #8 enabled for WTM
xxxxxxxxxxxx1xxxxxxxxx	Watchpoint #9 enabled for WTM
xxxxxxxxxxx1xxxxxxxxxx	Watchpoint #10 enabled for WTM
xxxxxxxxxx1xxxxxxxxxxx	Watchpoint #11 enabled for WTM
xxxxxxxxx1xxxxxxxxxxxx	Watchpoint #12 enabled for WTM
xxxxxxxx1xxxxxxxxxxxxx	Watchpoint #13 enabled for WTM
xxxxxxx1xxxxxxxxxxxxxx	Watchpoint #14 enabled for WTM
xxxxxx1xxxxxxxxxxxxxxx	Watchpoint #15 enabled for WTM
xxxxx1xxxxxxxxxxxxxxxx	Watchpoint #16 enabled for WTM
xxxx1xxxxxxxxxxxxxxxxx	Watchpoint #17 enabled for WTM
xxx1xxxxxxxxxxxxxxxxxx	Watchpoint #18 enabled for WTM
xx1xxxxxxxxxxxxxxxxxxx	Watchpoint #19 enabled for WTM
x1xxxxxxxxxxxxxxxxxxxx	Watchpoint #20 enabled for WTM
1xxxxxxxxxxxxxxxxxxxxx	Watchpoint #21 enabled for WTM

56.13.1 Watchpoint Timing Diagram (2 MDO / 1 MSEO configuration)

Figure 793. Watchpoint Message & Watchpoint Error Message



56.14 Nexus 2+ 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 2+ 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).

Using the Read/Write Access Registers (RWCS/RWA/RWD), memory mapped AHB resources can be accessed through Nexus 2+. 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 2+ module.

56.14.1 Single Write Access

1. Initialize the Read/Write Access Address Register (RWA) through the access method outlined in *Section 56.5*. 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 56.5*. 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 56.5*. 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 801* for details) and clears the ERR bit in the RWCS Register. Otherwise, if the access completes with an error, 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** pin as well as the AC, DV, and ERR bits within the RWCS provide Read/Write Access status to the external development tool.*

56.14.2 Block Write Access

1. For a block write access, follow Steps 1, 2, and 3 outlined in *Section 56.14.1* 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 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 56.14.1* 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.

56.14.3 Single Read Access

1. Initialize the Read/Write Access Address Register (RWA) through the access method outlined in *Section 56.5*. 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 56.5*. 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 801* for detail on **nex_rdy_b**). Otherwise, if the access has completed with an error, 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 56.5*.

*Note: Only the **nex_rdy_b** pin as well as the AC, DV and ERR bits within the RWCS provide Read/Write Access status to the external development tool.*

56.14.4 Block Read Access

1. For a block read access, follow Steps 1 and 2 outlined in *Section 56.14.3* 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 801* for detail on **nex_rdy_b**). Otherwise, if the access has completed with an error, 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 801* for detail on **nex_rdy_b**).

Otherwise, if the access has completed with an error 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 56.5*.
5. If AC has not been cleared due to an error, repeat Steps 3 and 4 in *Section 56.14.3* 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 (i.e. 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.

56.14.5 Error Handling

The Nexus 2+ module handles various error conditions as follows:

56.14.5.1 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

56.14.5.2 Access Termination

The following cases are defined for sequences of the Read/Write protocol that differ from those described in the above sections.

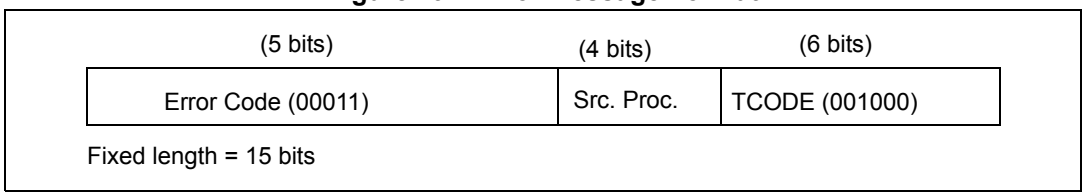
1. If the AC bit in the RWCS Register is set to start Read/Write accesses and invalid values are loaded into the RWD and/or RWA, then an AHB access error may occur. This is handled as described above.
2. If a block access is in progress (all cycles not completed), and the RWCS Register is written, then 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.

56.14.6 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 794. Error Message Format



56.15 Nexus 2+ Pin Interface

This section details information regarding the Nexus 2+ pins and pin protocol.

The Nexus 2+ 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.

56.15.1 Pins Implemented

The Nexus 2+ 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_evto_b** pin, and one (1) clock output pin (**nex_mcko**), as well as additional configuration pins described in *Table 801*. The output pins are synchronized to the Nexus 2+ output clock (**nex_mcko**).

All Nexus 2+ input functionality may be controlled through the JTAG/OnCE port in compliance with IEEE 1149.1 (see *Section 56.5* 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 Debug Support chapter.

Table 800. JTAG Pins for Nexus 2+

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 801*.

Table 801. Nexus 2+ 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 Nexus2+ 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 Nexus2+ 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_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 synchronization message (provided Program Trace is enabled & EIC = 2’b00). 2) Debug request to Nexus1 module (provided EIC = 2’b01 and this feature is implemented).
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

The Nexus auxiliary port arbitration pins are used when the Nexus 2+ module is implemented in a multi-Nexus SoC which shares a single auxiliary output port. The arbitration is controlled by an SoC level Nexus Port Controller (NPC). Refer to *Section 56.17* for detail on Nexus port arbitration.

Table 802. 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 804*.)
nex_aux_busy	O	Nexus Auxiliary Busy (**nex_aux_busy**) is an output signal to an SoC level Nexus arbiter indicating that the Nexus 2+ 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 2+ 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 2+ module is implemented within a multi-Nexus environment. If set, port control and arbitration is controlled by the SoC level arbitration module (NAR).

56.15.2 Pin Protocol

The protocol for the processor transmitting messages via the auxiliary pins is accomplished with the MSEO pin function outlined in *Table 803*. Both single and dual pin cases are shown.

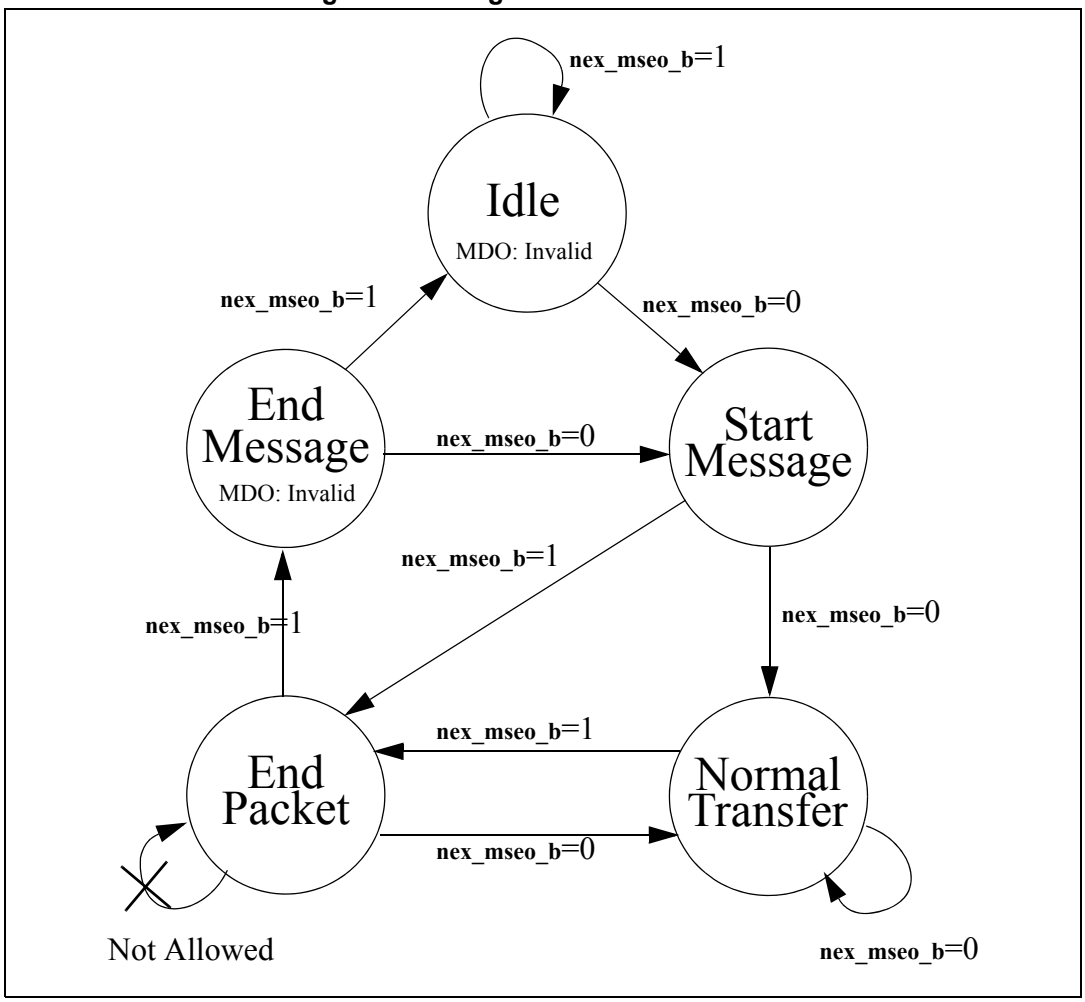
nex_mseo_b[1:0] is used to signal the end of variable-length packets, and not fixed length packets. **nex_mseo_b[1:0]** is sampled on the rising edge of the Nexus 2+ clock (**nex_mcko**).

Table 803. 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 795 illustrates the state diagram for single pin MSEO transfers.

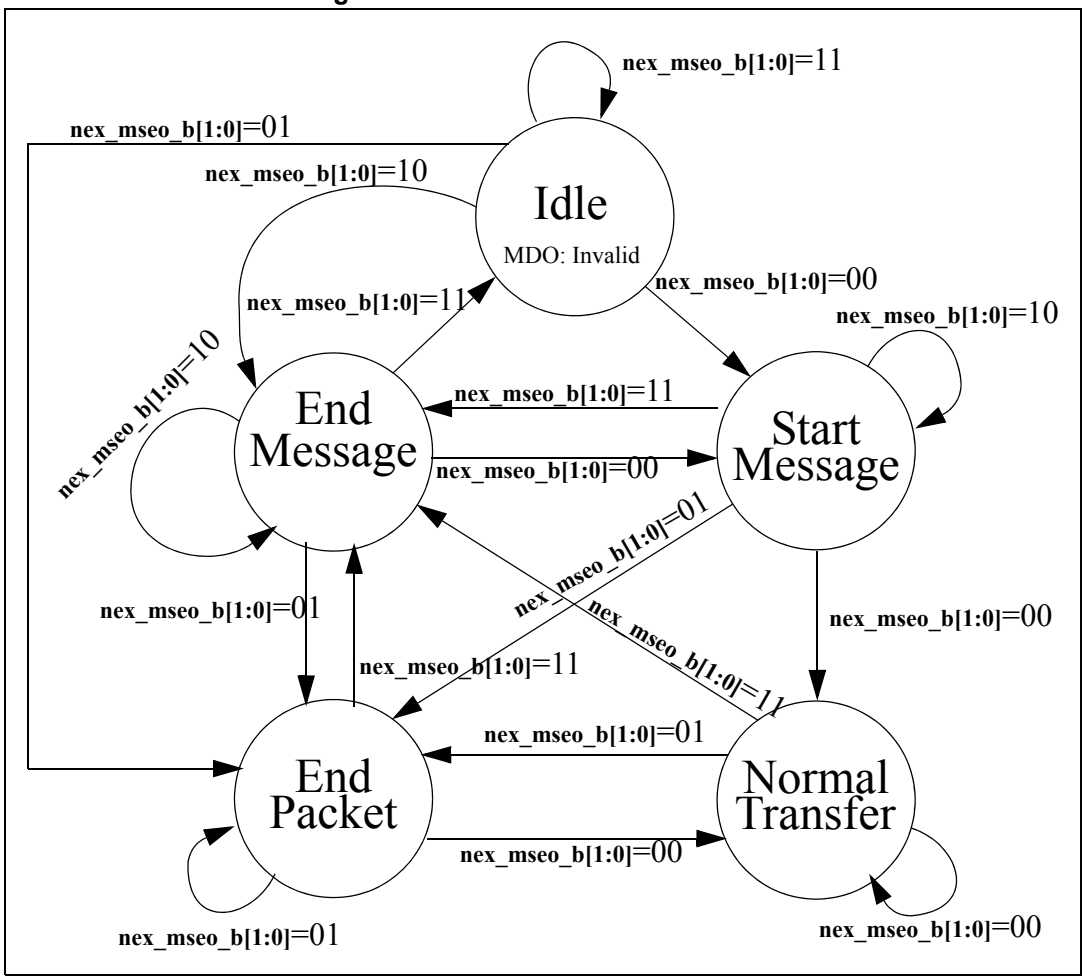
Figure 795. Single Pin MSEO Transfers



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 796 illustrates the state diagram for dual pin MSEO transfers.

Figure 796. 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.

56.16 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.

56.17 Auxiliary Port Arbitration

In a multi-Nexus environment, the Nexus 2+ 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 804* below. The Nexus 2+ module will receive a 1-bit grant signal (**nar_aux_grant**) from the SoC level arbiter. When a grant is received, the Nexus 2+ module will begin transmitting its message following the protocol outlined in *Section 56.15.2*. The Nexus 2+ 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 804. 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

56.18 Examples

The following are examples of Program Trace Messages.

Table 805 illustrates an example Indirect Branch Message with 2 MDO / 1MSEO configuration. *Table 806* 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 12, the **nex_mdo[n:0]** pins are ignored in the single MSEO case.

Table 805. 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 806. 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 807 and *Table 808* 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 807. 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 808. 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

56.19 Electrical Characteristics

For all electrical characteristics related to core processor and Nexus 2+ operation, please refer to the SPC570Sx datasheet.

56.20 IEEE 1149.1 (JTAG) RD/WR Sequences

This section contains example JTAG/OnCE sequences used to access resources.

56.20.1 JTAG Sequence for Accessing Internal Nexus Registers

Table 809. Accessing Internal Nexus 2+ 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)

Table 809. Accessing Internal Nexus 2+ Registers via JTAG/OnCE(Continued)

Step #	TMS Pin	Description
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)

56.20.2 JTAG Sequence for Read Access of Memory-Mapped Resources

Table 810. 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** 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

56.20.3 JTAG Sequence for Write Access of Memory-Mapped Resources

Table 811. 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)

Table 811. Accessing Memory-Mapped Resources (Writes)(Continued)

Step #	TCLK clocks	Description
7	—	Wait for falling edge of **nex_rdy_b** pin
8	—	If CNT > 0, go back to Step #5

57 Nexus Crossbar Multi-Master Client (NXMC)

57.1 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.

57.1.1 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 and/or writes 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

57.2 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 813*.

All NXMC input functionality is controlled through the JTAG port in compliance with IEEE 1149.1 (see *Section 57.5.1: IEEE 1149.1 (JTAG) Test Access Port* for details). A separate JTAG TAP controller is instantiated within the NXMC. *Table 812* describes the JTAG pins.

Table 812. 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 813. 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 57.5.2: Enabling NXMC operation*), assertion initiates data trace synchronization messages (provided data trace is enabled).

57.3 Supported messages and TCODEs

The NXMC 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 5001-2012 standard defines a set of public messages. The NXMC block supports the public TCODEs listed in *Table 814*. The SRC and MASTER field values are described in the device-specific chapter that describes how the modules are configured.

Table 814. 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 815*)
	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 815*)
	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 816*)
	14	14	ECODE	fixed	Error code (See *Table 817*)
	1	26	TSTAMP	variable	Timestamp (Optional)
Data Trace - Data Write Message w/ 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 815*)
	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 814. 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 w/ 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 815*)
	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 829*)
	1	26	TSTAMP	variable	Timestamp (Optional)

Table 815. Data trace size (DSZ) encodings (TCODE = 5,6,13,14)

DSZ	Transfer size
001	8-bit
010	16-bit
011	32-bit
000, 101-111	Reserved

Table 816. 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 817. 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.

57.4 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 57.5.1: IEEE 1149.1 (JTAG) Test Access Port* for details. The complete list of registers is shown in *Table 818*.

Table 818. 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 818. 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

57.4.1 Register definitions

Detailed register definitions for the current NXMC implementation are described in the following sections.

57.4.1.1 Development control register 1 (DC1)

The Development Control Registers control the basic development features of the NXMC module. *Figure 797* shows the format of the DC1.

Address: refer to *Table 818* 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	C	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

Figure 797. Development control register 1 (DC1)

1. This bit is not implemented and it's value has no impact on the system

The DC1 is described in *Table 819*.

Table 819. 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 819. 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

57.4.1.2 Development control register 2 (DC2)

The Development Control Registers control the basic development features of the NXMC module. *Figure 798* shows the format of the DC2.

Address: refer to *Table 818* 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
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 798. Development control register 2 (DC2)

The DC2 is described in *Table 820*.

Table 820. 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.

57.4.1.3 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 799* shows the format of the WT register.

Address: refer to *Table 818* 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 799. Watchpoint trigger register (WT)

The WT register is described in *Table 821*.

Table 821. 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. 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

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.

57.4.1.4 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 800* shows the format of the DTC

register.

Address: refer to *Table 818* 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 800. Data trace control (DTC) register

The DTC register is described in *Table 822*.

Table 822. 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.

57.4.1.5 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 801* shows the format of the DTSA1 and DTSA2 registers.

Address: refer to *Table 818* 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[31:16]															
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	DATA TRACE START ADDRESS[15:0]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 801. Data trace start address registers 1 and 2 (DTSA1 and DTSA2)

57.4.1.6 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 802* shows the format of the DTEA1 and DTEA2 registers.

Address: refer to *Table 818* 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[31:16]															
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	DATA TRACE END ADDRESS[15:0]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 802. Data trace end address registers 1 and 2 (DTEA1 and DTEA2)

Table 823 illustrates the range that are selected for data trace for various cases of DTSA being less than, greater than, or equal to DTEA.

Table 823. 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.

57.4.1.7 Breakpoint/watchpoint control register 1 (BWC1)

Breakpoint/watchpoint control register 1 controls the attributes for generation of NXMC watchpoint#1. *Figure 803* shows the format of the BWC1 register.

Address: refer to *Table 818* 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 803. Breakpoint/watchpoint control register 1(BWC1)

The BWC1 register is described in *Table 824*.

Table 824. 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

57.4.1.8 Breakpoint/watchpoint control register 2 (BWC2)

Breakpoint/Watchpoint Control Register2 controls attributes for generation of NXMC watchpoint#2. *Figure 804* shows the format of the BWC2 register.

Address: refer to *Table 818* 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 804. Breakpoint/watchpoint control register 2(BWC2)

The BWC2 register is described in *Table 825*.

Table 825. BWC2 field description

Field	Description
0:1 BWE2	Breakpoint/Watchpoint #2 Enable. 00Internal 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

57.4.1.9 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 805* shows the format of the BWA1 and BWA2 registers.

Address: refer to *Table 818* Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	BREAKPOINT / WATCHPOINT ADDRESS[31:16]															
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	BREAKPOINT / WATCHPOINT ADDRESS[15:0]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 805. Breakpoint /watchpoint address registers 1 and 2 (BWA1 and BWA2)

57.4.1.10 Unimplemented registers

Unimplemented registers are those with client select and index value combinations other than those listed in *Table 818*. 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).

57.5 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.

57.5.1 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 57.5.4: NXMC register access via JTAG*.

57.5.2 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.

57.5.3 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, etc.

57.5.4 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 818*.

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 818*), 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 806* and *Table 826*.



Figure 806. JTAG DR for Nexus register access

Table 826. JTAG DR field description for Nexus register access

Field	Description
Nexus Register Index	Selected from values in *Table 818*
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.

57.6 Functional description

57.6.1 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.

57.6.1.1 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.

57.6.1.2 DTM message formats

The NXMC block supports five types of DTM messages:

- Data write
- Data read
- Data write synchronization
- Data read synchronization
- Error messages

57.6.1.2.1 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. Data write message and data read message information is messaged out in the format shown in *Figure 807*. The SRC field values are described in the device-specific chapter that describes how the modules are configured.

8/16/32 bits	1–32 bits	3 bits	4 bits	4 bits	6 bits
DATA	U-ADDR	DSZ	MASTER	SRC	TCODE (111010 or 111011)

Figure 807. DTM message format for DATA width = 32 bits

57.6.1.2.2 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.

57.6.1.2.3 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 808*. The SRC field values are described in the device-specific chapter that describes how the modules are configured.

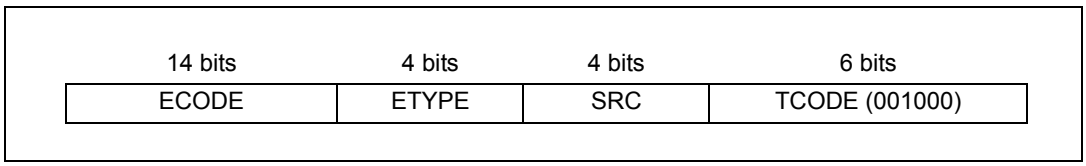


Figure 808. DTM error message format (fixed length = 48 bits)

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 827*):

- 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 809*. The SRC and MASTER field values are described in the device-specific chapter that describes how the modules are configured.

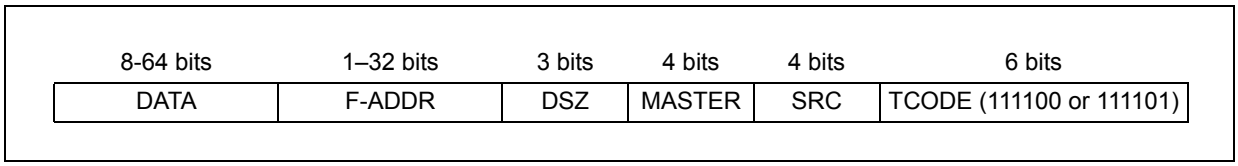


Figure 809. Data write/read with synchronization message format

Exception conditions that result in data trace synchronization are summarized in *Table 827*.

Table 827. 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.

57.6.1.3 DTM operation

57.6.1.3.1 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 (i.e., DTC, DTSA/E1/2 and WT register fields) are correctly configured to ensure correct DTM functionality.

57.6.1.3.2 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.

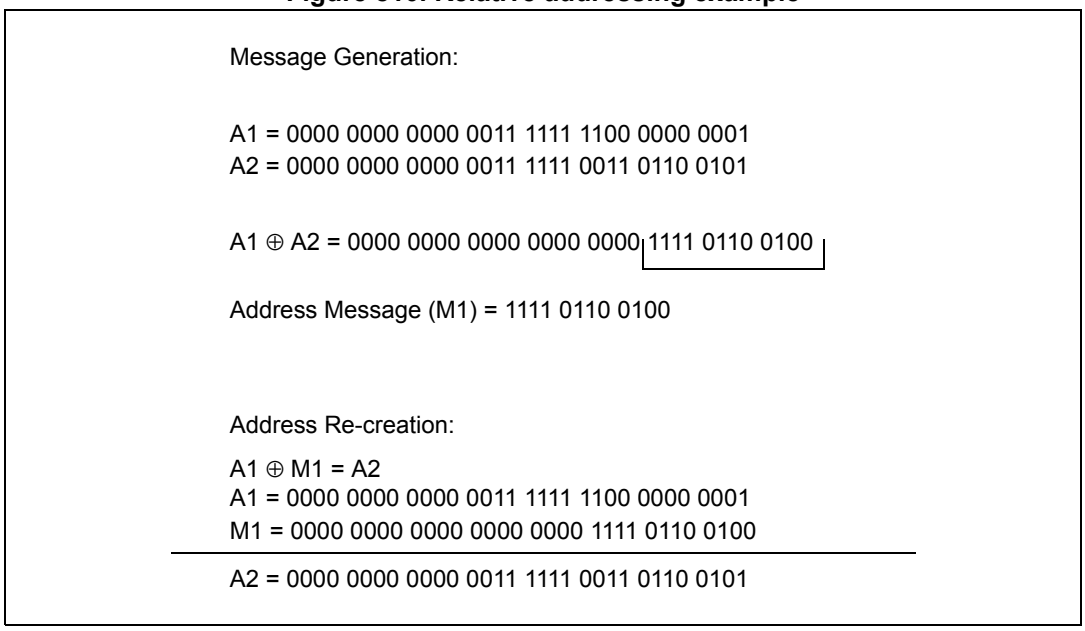
Note: If multiple trace messages need to be queued at the same time, watchpoint messages have the highest priority (WPM > ICTM > DTM).

57.6.1.3.3 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 810*.

Figure 810. Relative addressing example



57.6.1.3.4 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.

57.6.1.3.5 System bus cycle special cases

The system bus cycle special cases are shown in *Table 828*.

Table 828. 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

57.6.1.3.6 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 57.6.1.2.1: 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.

57.6.2 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.

57.6.2.1 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 57.6.1.3.6: 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 (i.e., 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 811* and *Table 829*.

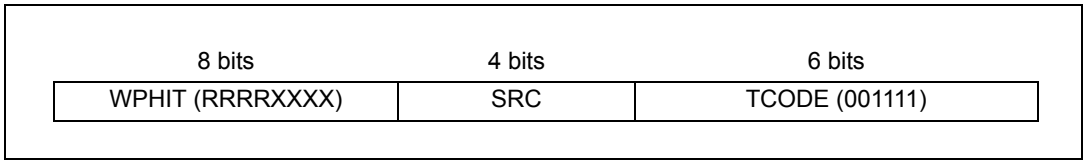


Figure 811. Watchpoint message format (fixed length = 18 bits)

Table 829. 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).

57.6.2.2 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 812*.

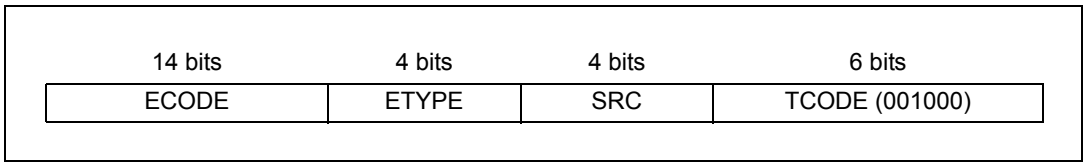


Figure 812. Error message format (fixed length = 28 bits)

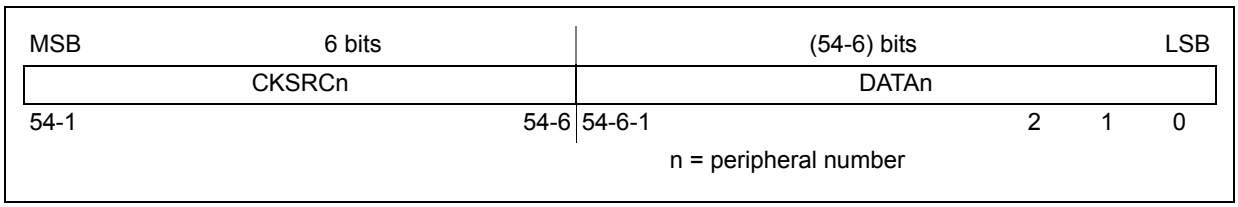


Figure 813. Peripheral DATA bus fields

58 Cyclic Redundancy Check (CRC) Unit

58.1 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.

58.2 Main features

- 1 contexts for the concurrent CRC computation
- Separate CRC engine for each context
- Zero-wait states during the CRC computation (pipeline scheme)
- 4 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

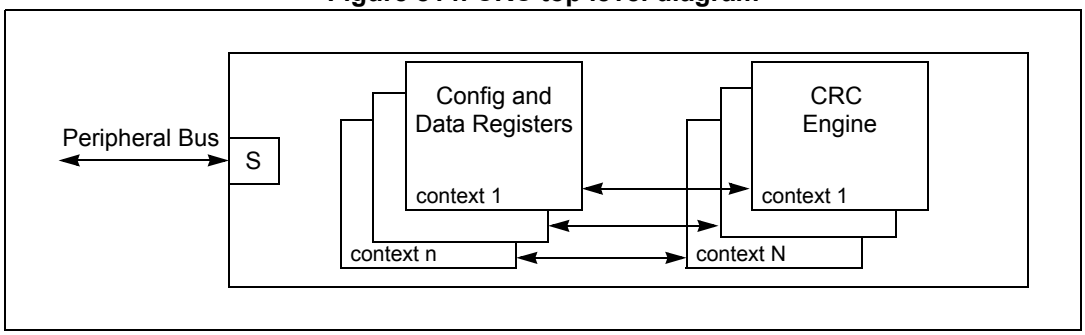
58.2.1 Standard features

- Peripheral bus interface
- CRC-8 VDA CAN
- CRC-16-CCITT
- CRC-32 Ethernet
- CRC-8-H2F Autosar polynomial

58.3 Block diagram

The top level diagram of the CRC module is given in *Figure 814*.

Figure 814. CRC top level diagram



1. Where context number, n = 1.

58.4 Signal description

The CRC module does not generate any external signals.

58.4.1 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

58.5 Memory map and registers

This section describes, in address-order, all the CRC registers.

58.5.1 Register description

Table 830 shows the memory map for the CRC registers.

Table 830. CRC unit memory map

Offset from CRCU_BASE	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 58.5.1.1*
0x004 + *n**0x10	CRC Input Register *n* (CRC_INP*n*)	Saf-Relv	*Section 58.5.1.2*

Table 830. CRC unit memory map(Continued)

Offset from CRCU_BASE	Register	Register classification(1)	Location
0x008 + *n**0x10	CRC Current Status Register *n* (CRC_CSTAT*n*)	Saf-Relv	*Section 58.5.1.3*
0x00C + *n**0x10	CRC Output Register *n* (CRC_OUTP*n*)	Saf-Relv	*Section 58.5.1.4*

1. No-Saf = Not safety relevant, Saf-Lat = Safety Latent, Saf-Relv = Safety Relevant, Saf-Crit = Safety Critical.

58.5.1.1 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 815. 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 831. 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.
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.

Table 831. CRC_CFG*n* field descriptions(Continued)

Field	Description
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.

58.5.1.2 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 816. CRC Input Register *n* (CRC_INP*n*)

Table 832. 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.

58.5.1.3 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 817. CRC Current Status Register *n* (CRC_CSTAT*n*)

Table 833. 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.

58.5.1.4 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 818. CRC Output Register *n* (CRC_OUTP*n*)

Table 834. 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.

58.6 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 42*, *Equation 43* and *Equation 44* 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.

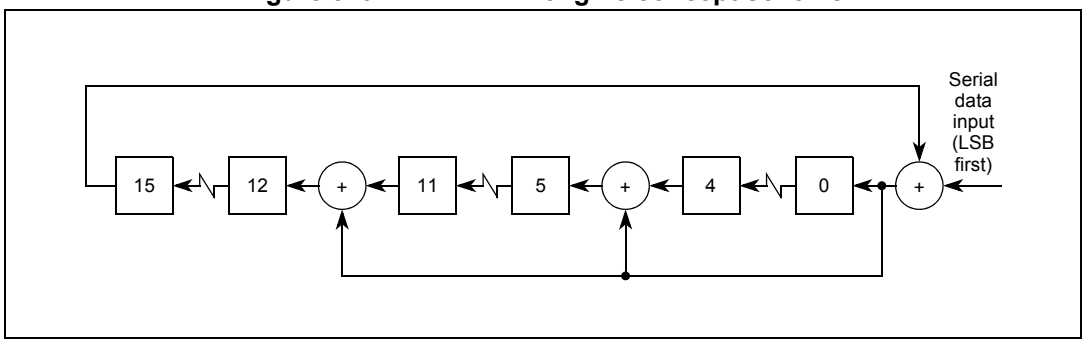
Equation 42 CRC8 VDA CAN:
$$x^8 + x^4 + x^3 + x^2 + 1$$

Equation 43 CRC-CCITT (x.25 protocol):
$$x^{16} + x^{12} + x^5 + 1$$

Equation 44 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 45 CRC-8-H2F (Autosar 4.0 protocol):
$$x^5 + x^3 + x^2 + x + 1$$

Figure 819. CRC-CCITT engine concept scheme

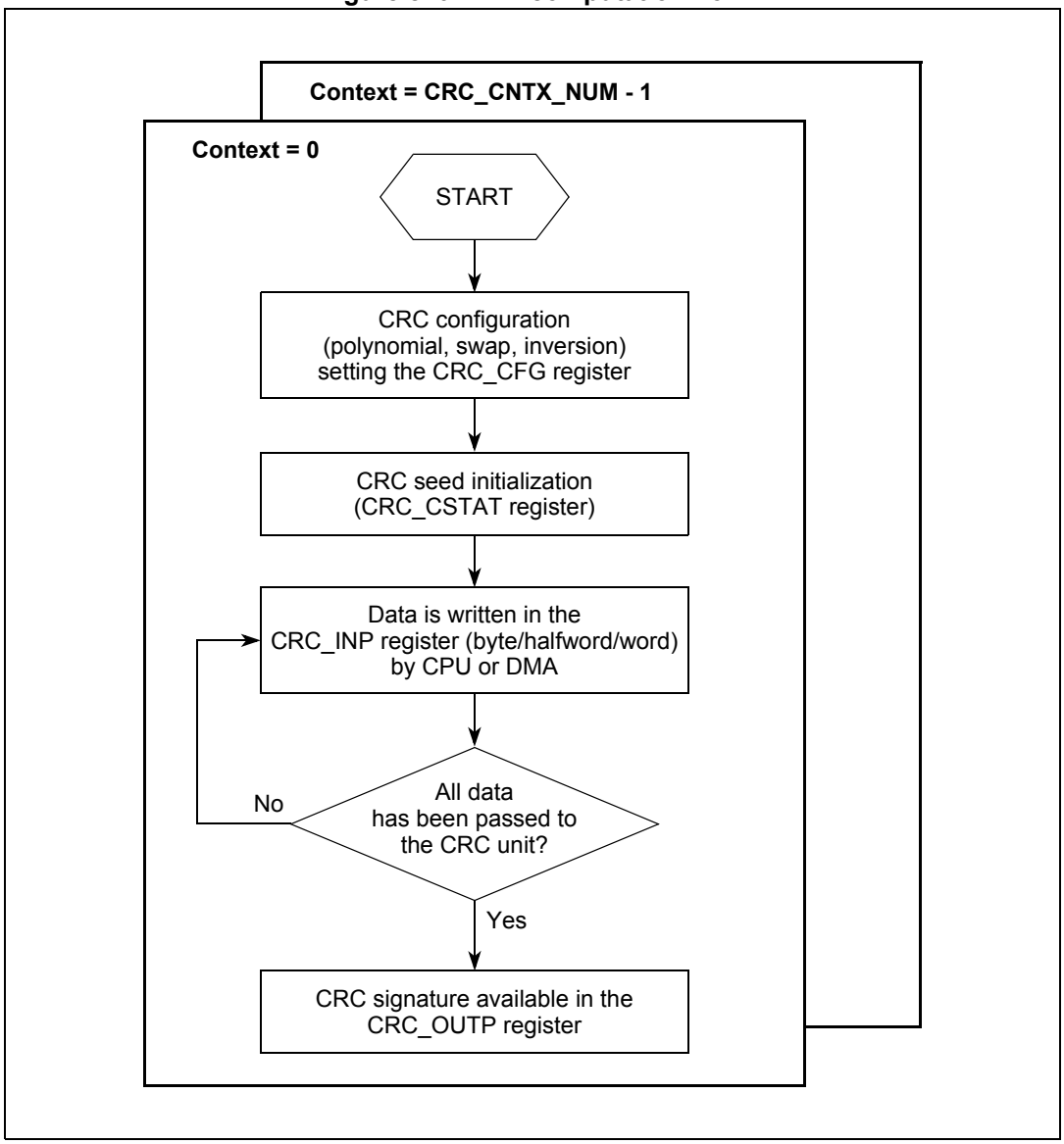


The initial seed value of the CRC can be programmed by initializing the CRC_CSTAT register. The concept scheme (serial data loading) of the CRC engine is given in *Figure 819* for the CRC-CCITT. The design implementation executes the CRC computation in a single clock cycle (parallel data loading). A 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 (for example, SPI, etc.), a bit swap (MSB → LSB, LSB → MSB) and/or bit inversion of the final CRC signature can be applied (CRC_OUTP register) before transmitting the CRC.

Use of the CRC module is summarized in the flow chart given in *Figure 820*.

Figure 820. CRC computation flow



58.7 Use cases

58.7.1 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:

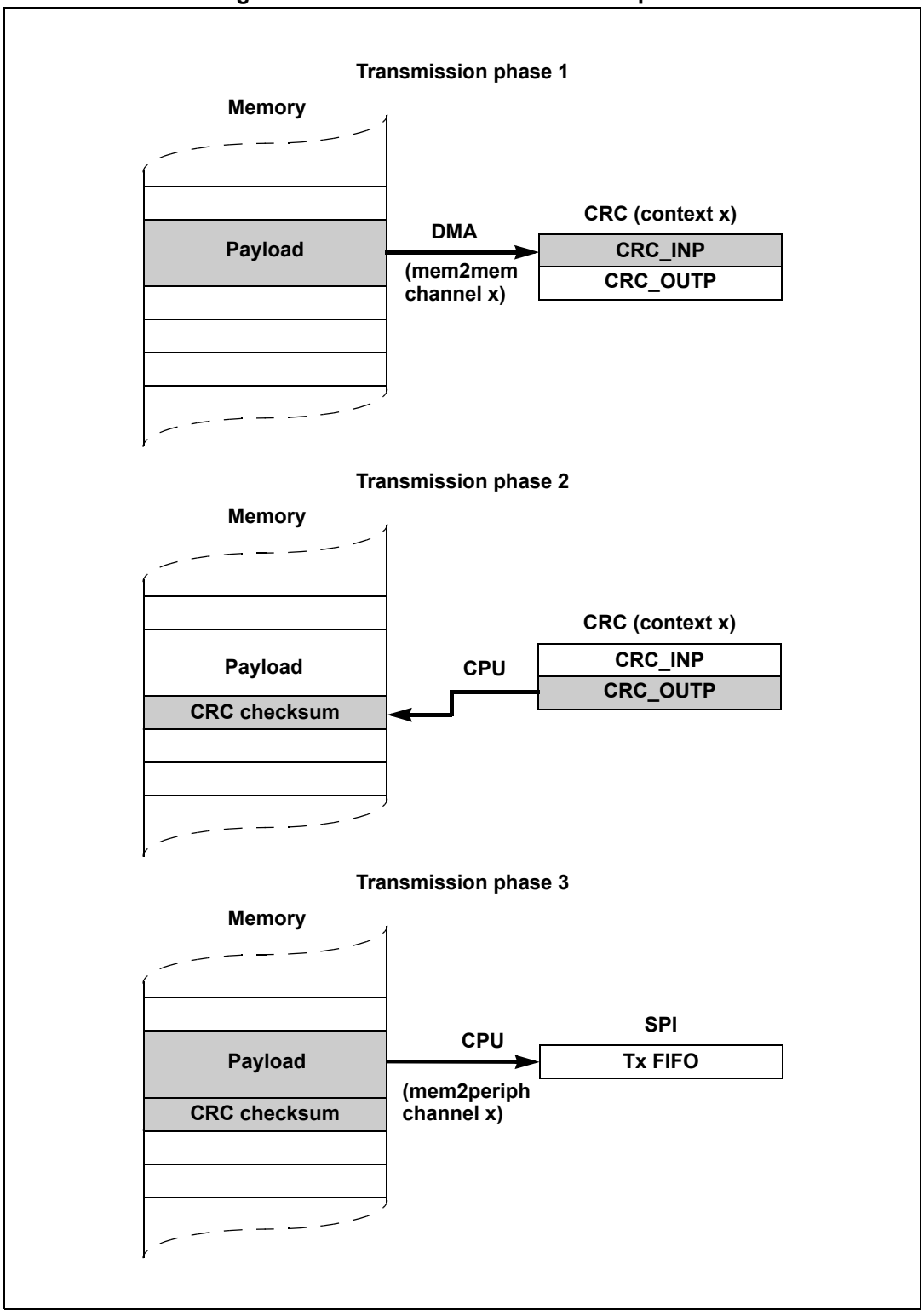
- Calculation of the CRC of the configuration registers during the process safety time
- Calculation of the CRC on the incoming/outcoming frames for the communication protocols (not protected with CRC by definition of the protocol itself) used as a safety-relevant peripheral

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 821*, 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 821. DMA-CRC transmission sequence



58.7.2 Register programming

- INV and/or SWAP (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

59 Memory Error Management Unit (MEMU)

59.1 Introduction

The MEMU is responsible for collection and reporting of error events associated with ECC (Error Correction Code) logic used on:

- SRAM
- Peripheral System 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 Bit error in the data part that is detected via ECC for a System RAM or Peripheral RAM or flash memory.
  - Single Bit error in the data part that is detected via MBIST on any RAM.
- Uncorrectable Error: It comprises the following:
  - Multiple bit error that is detected via ECC for a System RAM or Peripheral RAM or flash memory.
  - 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 59.4: Design overview* for the reporting block details.

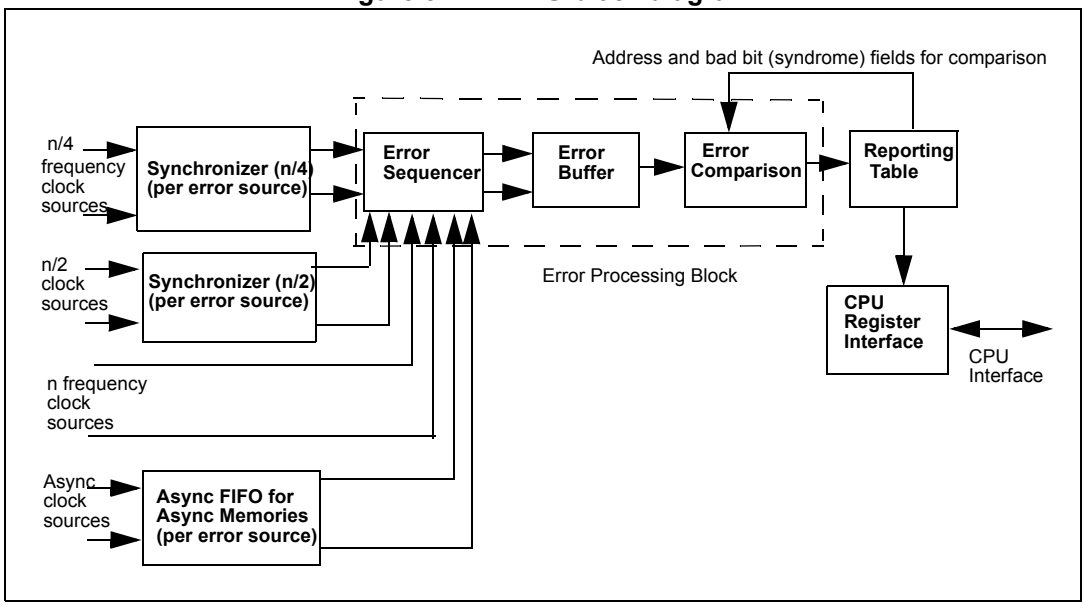
59.2 Features

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.

59.3 Block diagram

Figure 822. MEMU block diagram



59.4 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 59.7.3: 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 (*System RAM correctable error reporting table status register (SYS_RAM_CERR_STSn)* or *System RAM uncorrectable error reporting table status register (SYS_RAM_UNCERR_STS)*). 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 823. MEMU error buffer uniqueness check flow chart

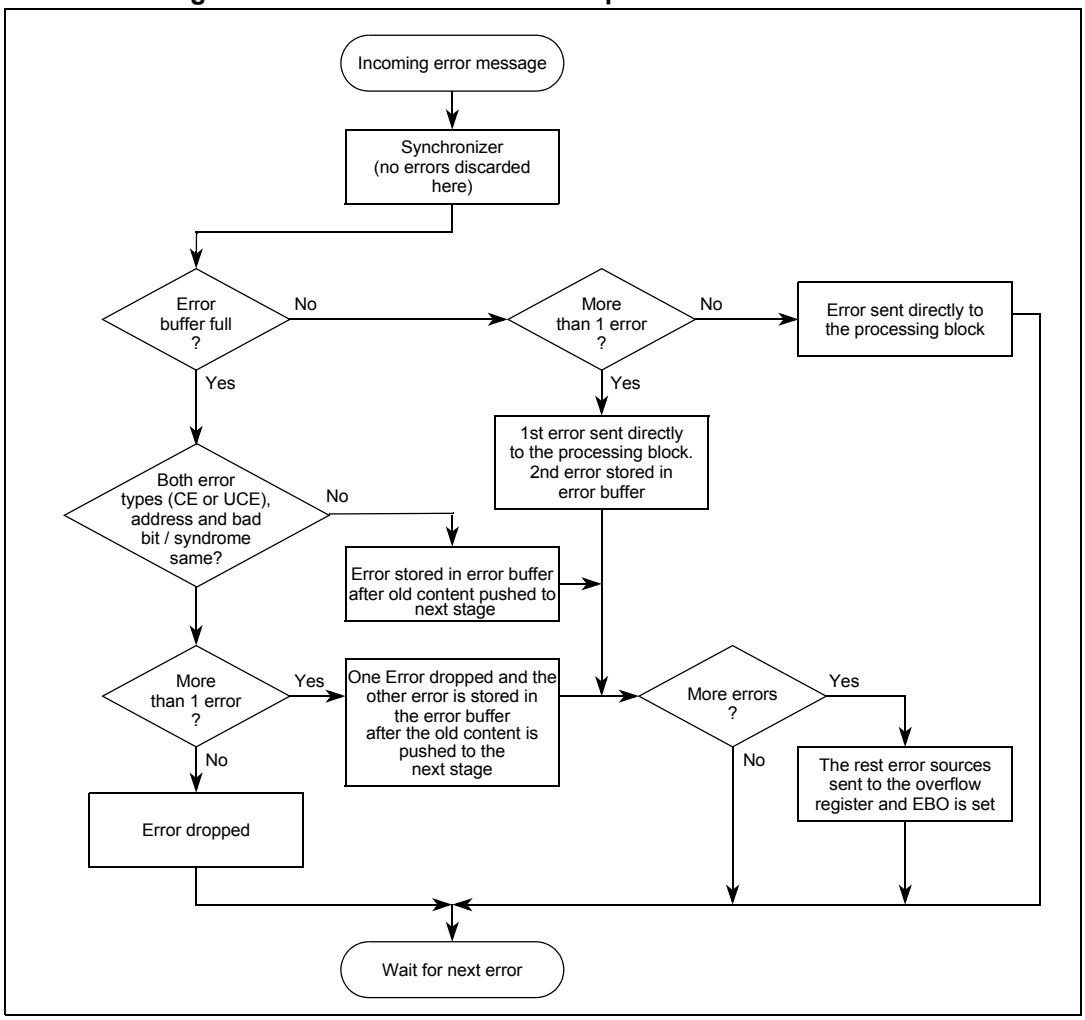
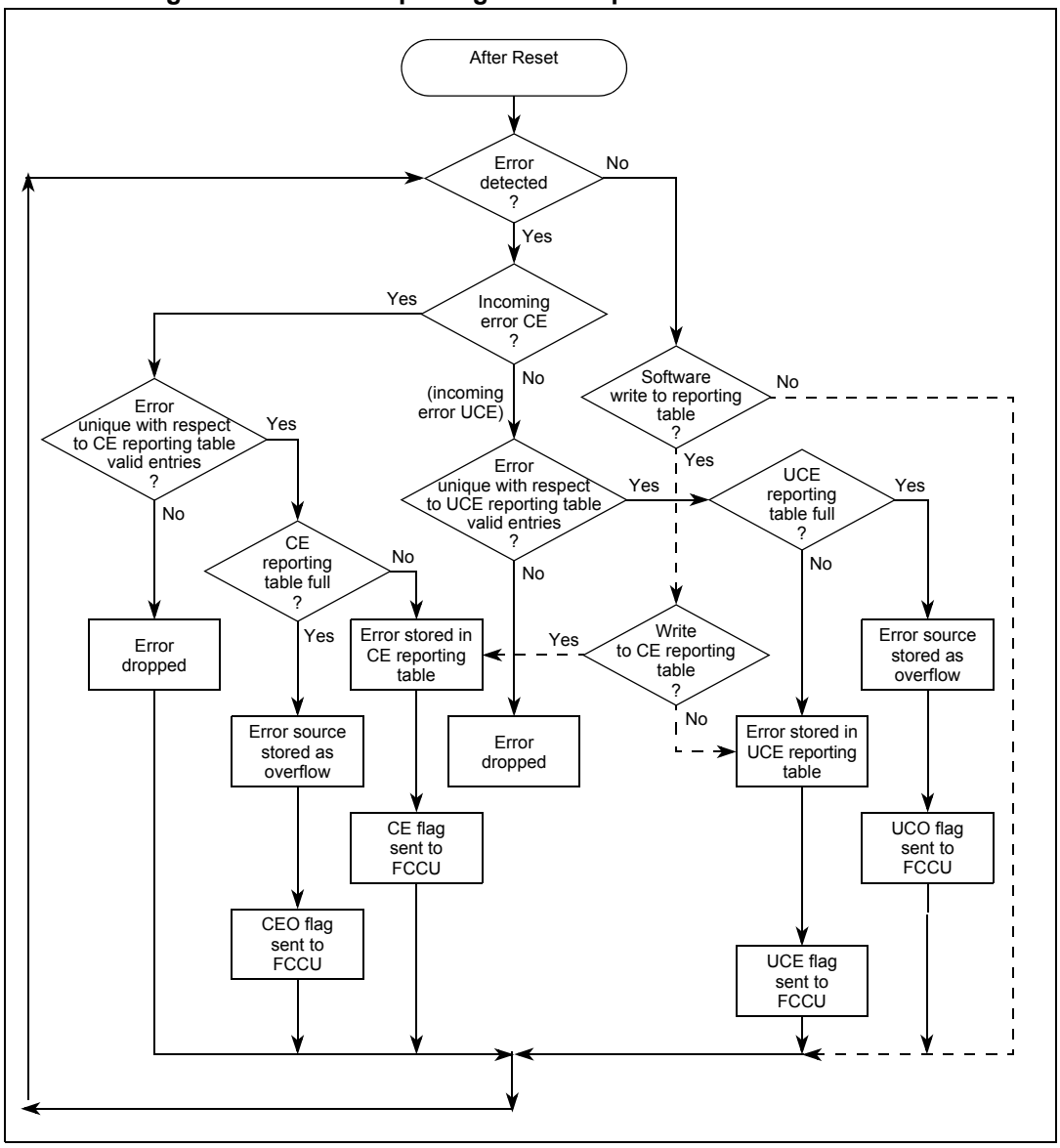


Figure 824. MEMU reporting table uniqueness check flow chart



59.5 External signal description

MEMU has no external signals.

59.6 Memory map and register definition

59.6.1 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 835. MEMU memory map

Offset or address	Register	Access	Reset value	Page
0x0000	Control register (CTRL)	R/W	0000_0000	59.6.2.1 on page 1452
0x0004	Error flag register (ERR_FLAG)	R/W	0000_0000	59.6.2.2 on page 1453
0x000C	Debug register (DEBUG)	R/W	0000_0000	59.6.2.3 on page 1455
0x0020 + n*0x8 (n = 0 to 10–1)	System RAM correctable error reporting table status registers (SYS_RAM_CERR_STSn)	R/W	0000_0000	59.6.2.4 on page 1457
0x0024 + n*0x8 (n = 0 to 10–1)	System RAM correctable error reporting table address registers (SYS_RAM_CERR_ADDRn)	R/W	0000_0000	59.6.2.5 on page 1458
0x0028 + (10 –1)*0x8	System RAM uncorrectable error reporting table status register (SYS_RAM_UNCERR_STS)	R/W	0000_0000	59.6.2.6 on page 1458
0x002C + (10–1)*0x8	System RAM uncorrectable error reporting table address register (SYS_RAM_UNCERR_ADDR)	R/W	0000_0000	59.6.2.7 on page 1459
0x0030 + n*0x4 + (10–1)*0x8 (n = 0 to 9÷32 – 1)(1)	System RAM concurrent overflow registers(2) (SYS_RAM_OFLWn)	w1c	0000_0000	59.6.2.8 on page 1460
0x0620 + n*0x8 (n = 0 to 2–1)	Peripheral RAM correctable error reporting table status registers (PERIPH_RAM_CERR_STSn)	R/W	0000_0000	59.6.2.9 on page 1460
0x0624 + n*0x8 (n = 0 to 2–1)	Peripheral RAM correctable error reporting table address registers (PERIPH_RAM_CERR_ADDRn)	R/W	0000_0000	59.6.2.10 on page 1461
0x0628 + (2–1)*8	Peripheral RAM uncorrectable error reporting table status register (PERIPH_RAM_UNCERR_STS)	R/W	0000_0000	59.6.2.11 on page 1461
0x062C + (2–1)*8	Peripheral RAM uncorrectable error reporting table address register (PERIPH_RAM_UNCERR_ADDR)	R/W	0000_0000	59.6.2.12 on page 1462
0x0630 + n*0x4 + (2–1)*0x8 (n = 0 to 1÷32 – 1)(1)	Peripheral RAM concurrent overflow registers(1) (PERIPH_RAM_OFLWn)	w1c	0000_0000	59.6.2.13 on page 1463

Table 835. 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_STSn)	R/W	0000_0000	*59.6.2.14 on page 1463*
0x0C24 + n*0x8 (n = 0 to 20–1)	Flash memory correctable error reporting table address registers (FLASH_CERR_ADDRn)	R/W	0000_0000	*59.6.2.15 on page 1464*
0x0C28 + (20–1)*8	Flash memory uncorrectable error reporting table status register (FLASH_UNCERR_STS)	R/W	0000_0000	*59.6.2.16 on page 1464*
0x0C2C + (20–1)*8	Flash memory uncorrectable error reporting table address register (FLASH_UNCERR_ADDR)	R/W	0000_0000	*59.6.2.17 on page 1465*
0x0C30 + n*0x4 + (20–1)*0x8 (n = 0 to 2÷32 – 1)(1)	Flash memory concurrent overflow registers(1)(FLASH_OFLWn)	w1c	0000_0000	*59.6.2.18 on page 1466*

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.

59.6.2 Register descriptions

59.6.2.1 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 825. Control register (CTRL)

Table 836. 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.

59.6.2.2 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 826. 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 837. 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 837. 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 837. 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.

59.6.2.3 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 827. Debug register (DEBUG)

Table 838. 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 838. 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.

59.6.2.4 System RAM correctable error reporting table status register (SYS_RAM_CERR_STSn)

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 828. System RAM correctable error reporting table status register (SYS_RAM_CERR_STSn)

Table 839. SYS_RAM_CERR_STSn 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.

59.6.2.5 System RAM correctable error reporting table address register (SYS_RAM_CERR_ADDRn)

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[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	ERR_ADD[16:31]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 829. System RAM correctable error reporting table address register (SYS_RAM_CERR_ADDRn)

Table 840. SYS_RAM_CERR_ADDRn field descriptions

Field	Description
0:31 ERR_ADD	Error address field Indicates the address on which the error was detected.

59.6.2.6 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 830. System RAM uncorrectable error reporting table status register (SYS_RAM_UNCERR_STS)

Table 841. 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.

59.6.2.7 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[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	ERR_ADD[16:31]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 831. System RAM uncorrectable error reporting table address register (SYS_RAM_UNCERR_ADDR)

Table 842. 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.

59.6.2.8 System RAM concurrent overflow registers (SYS_RAM_OFLWn)

Note: The reporting table is not cleared on software reset.

Offset: 0x0030 + n*0x4 + (10–1)*0x8
(n = 0 to 9÷32 – 1)(1) Access: R/W

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	OFLW[0:15]															
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[16:31]															
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 832. System RAM concurrent overflow registers (SYS_RAM_OFLWn)

Table 843. SYS_RAM_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 59.7.3: Handling overflows (Multiple error reporting)* occurs.

59.6.2.9 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 833. Peripheral RAM correctable error reporting table status register (PERIPH_RAM_CERR_STSn)

Table 844. PERIPH_RAM_CERR_STSn 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.

59.6.2.10 Peripheral RAM correctable error reporting table address register (PERIPH_RAM_CERR_ADDRn)

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[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	ERR_ADD[16:31]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 834. Peripheral RAM correctable error reporting table address register (PERIPH_RAM_CERR_ADDRn)

Table 845. PERIPH_RAM_CERR_ADDRn field descriptions

Field	Description
0:31 ERR_ADD	Error address field Indicates the address on which the error was detected.

59.6.2.11 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)*8 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 835. Peripheral RAM uncorrectable error reporting table status register (PERIPH_RAM_UNCERR_STS)

Table 846. 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.

59.6.2.12 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)*8 Access: R/W

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	ERR_ADD[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	ERR_ADD[16:31]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 836. Peripheral RAM uncorrectable error reporting table address register (PERIPH_RAM_UNCERR_ADDR)

Table 847. 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.

59.6.2.13 Peripheral RAM concurrent overflow registers (PERIPH_RAM_OFLWn)

Note: The reporting table is not cleared on software reset.

Offset: 0x0630 + n*0x4 + (2–1)*0x8
(n = 0 to 1÷32 – 1)(1) Access: R/W

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	OFLW[0:15]															
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[16:31]															
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 837. Peripheral RAM concurrent overflow registers (PERIPH_RAM_OFLWn)

Table 848. PERIPH_RAM_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 59.7.3: Handling overflows (Multiple error reporting)* occurs.

59.6.2.14 Flash memory correctable error reporting table status register (FLASH_CERR_STSn)

Note: The reporting table is not cleared on software reset.

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 838. Flash memory correctable error reporting table status register (FLASH_CERR_STSn)

Table 849. FLASH_CERR_STSn 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.

59.6.2.15 Flash memory correctable error reporting table address register (FLASH_CERR_ADDRn)

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[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	ERR_ADD[16:31]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 839. Flash memory correctable error reporting table address register (FLASH_CERR_ADDRn)

Table 850. FLASH_CERR_ADDRn register field descriptions

Field	Description
0:31 ERR_ADD	Error address field Indicates the address on which the error was detected.

59.6.2.16 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)*8 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 840. Flash memory uncorrectable error reporting table status register (FLASH_UNCERR_STS)

Table 851. 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.

59.6.2.17 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[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	ERR_ADD[16:31]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 841. Flash memory uncorrectable error reporting table address register (FLASH_UNCERR_ADDR)

Table 852. 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.

59.6.2.18 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[0:15]															
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[16:31]															
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 842. Flash memory concurrent overflow registers (FLASH_OFLWn)

Table 853. 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 59.7.3: Handling overflows (Multiple error reporting)* occurs.

59.7 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.

59.7.1 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:

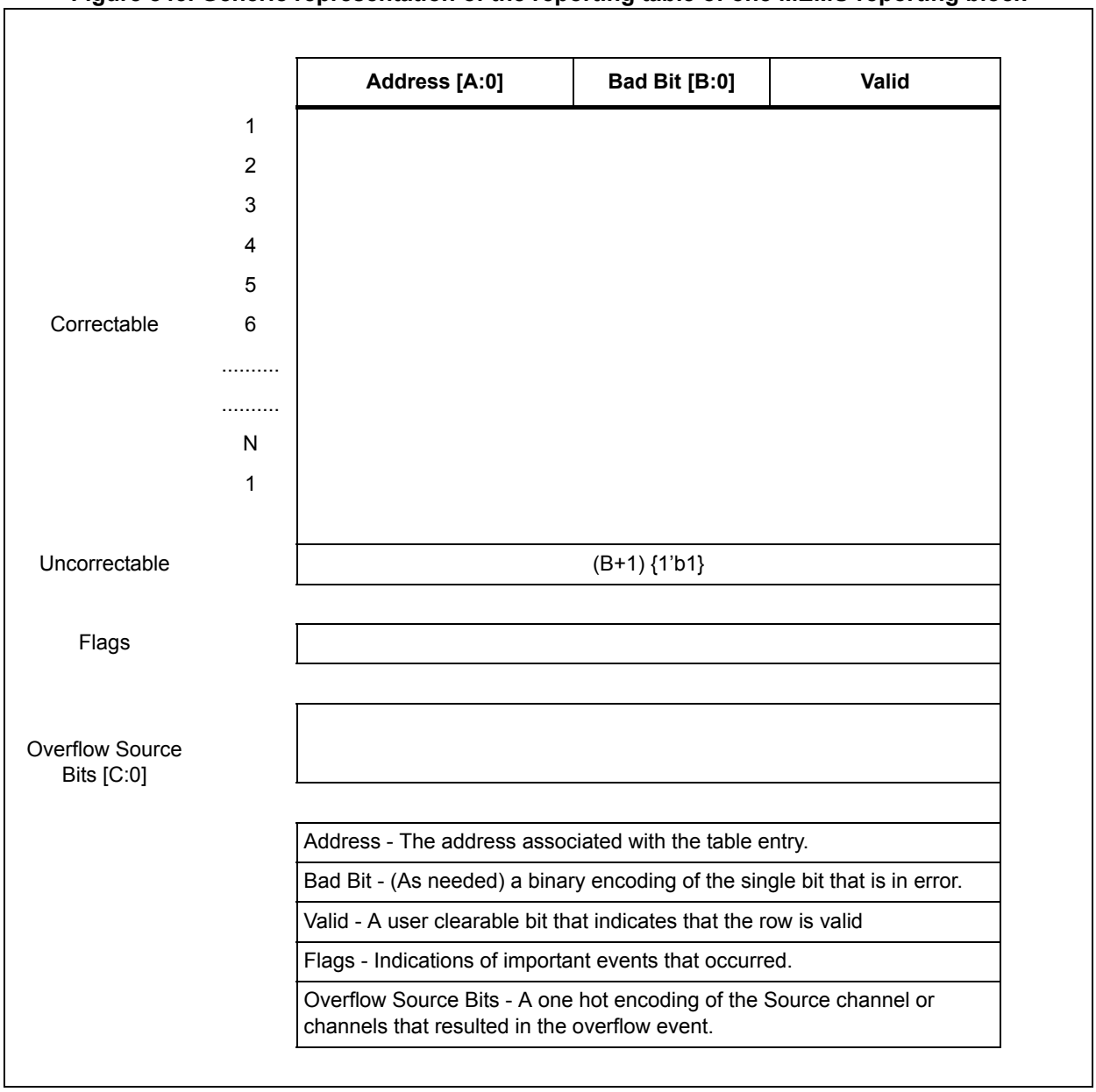
1. Check that the VLD bit is not set,
2. Write an invalid address/bad bit information,
3. Set VLD bit,
4. Check that the address still has an invalid value,
5. Write address/bad bit information to desired address.

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).

59.7.2 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 843. Generic representation of the reporting table of one MEMU reporting block



59.7.3 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’.

60 Fault Collection and Control Unit (FCCU)

60.1 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.

60.2 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[w]. 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.

60.2.1 Standard features

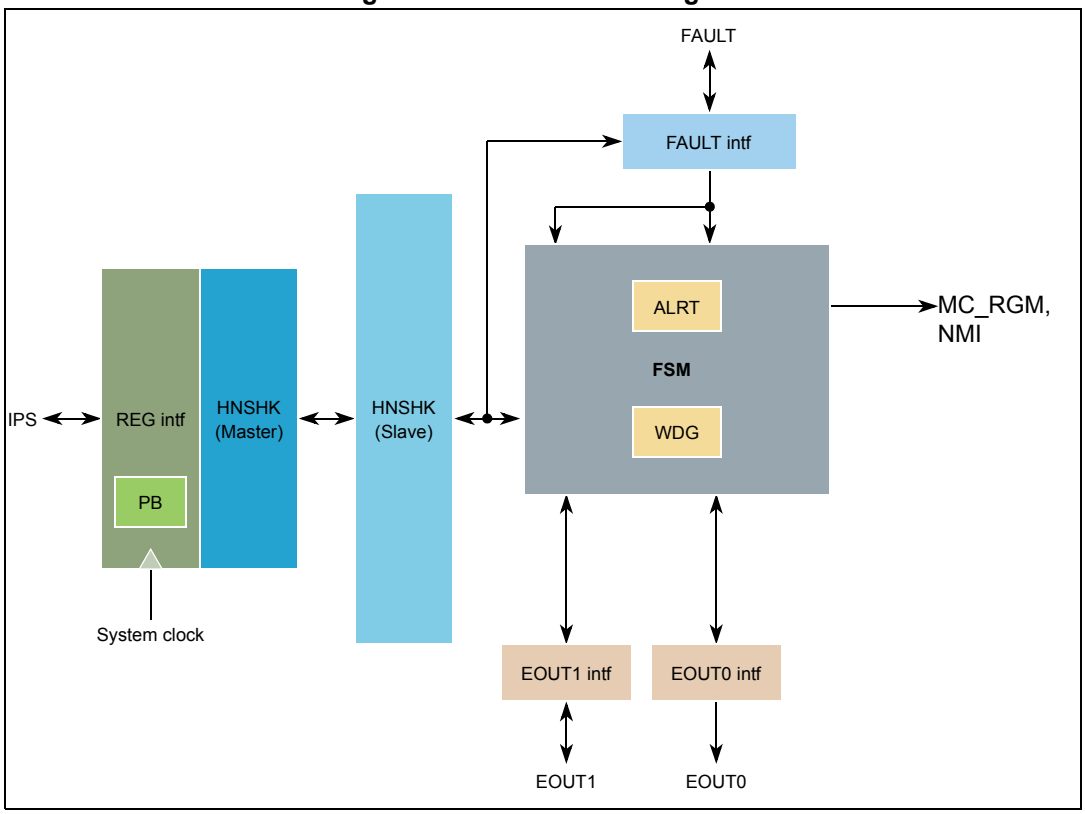
- IPS (slave bus signals) bus interface

60.3 Block diagram

The top-level diagram of the FCCU module is given in *Figure 844*.

w. Actual value depends on the SoC settings at pad level

Figure 844. FCCU block diagram



The FCCU module includes the submodules listed in *Table 854*.

Table 854. 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

60.4 Signal description

60.4.1 Pinout

The FCCU generates two external signals, EOUT[0] and EOUT[1]. These are described in *Section 60.5.9*.

For the availability of these signals on a chip, see the device configuration chapter.

60.4.2 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 863*
- 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.

60.5 Functional description

60.5.1 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 a 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

60.5.2 FSM description

The FCCU module functionality is depicted by the FSM diagram given in *Figure 845*.

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 sub-set 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 writeable 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

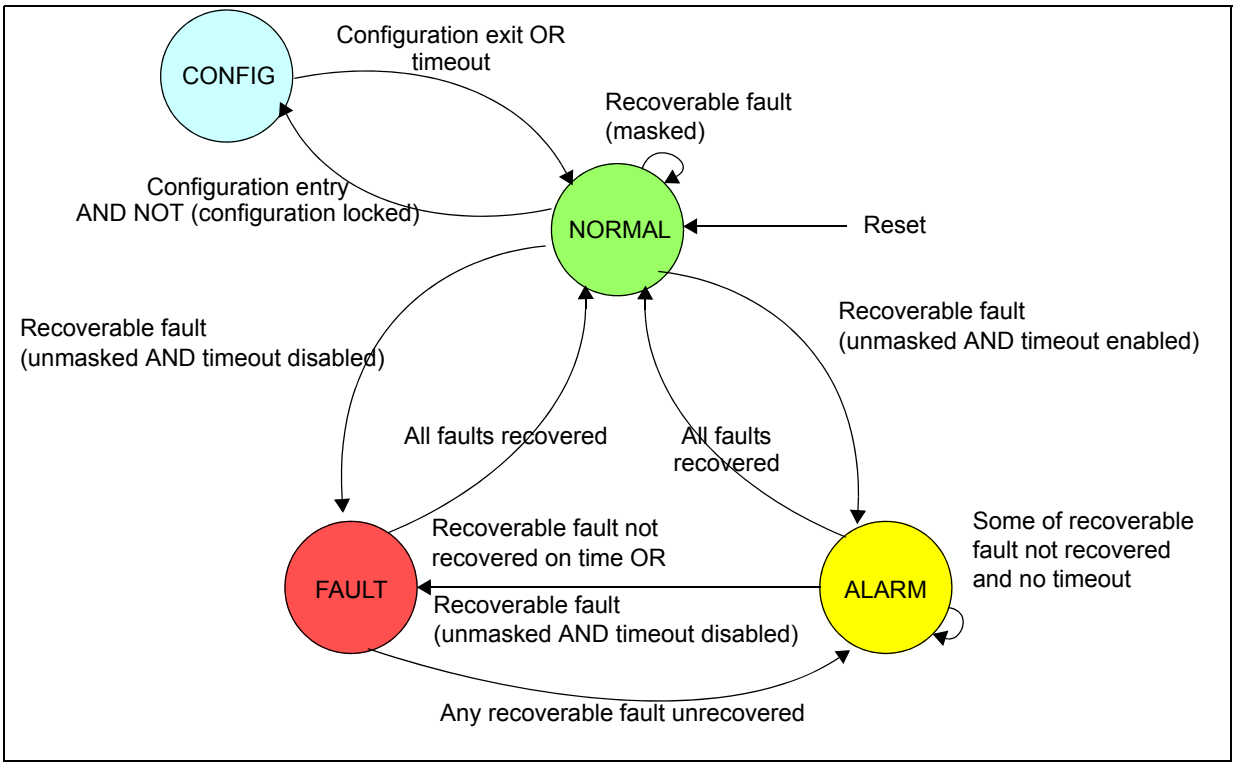
- FAULT: the FCCU moves into the FAULT state when one of the following condition occurs:
  - timeout related to a recoverable fault when the FCCU is in the ALARM state
  - unmasked recoverable faults with the timeout disabled

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 845. FCCU state diagram



60.5.3 Reset interface

The FCCU has two input resets, as described in *Table 855* 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 856* 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 857* 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 855*.

Table 855. 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 856 and *Table 857* describe a typical reset strategy at system level.

Table 856. Reset reactions

FAULT reaction	Active reset	System reset signal
Hard	Long functional reset	disabled
Soft	Short functional reset	disabled

Table 857 shows the Reset sources.

Table 857. Reset sources

Reset source	System reset signal
External reset	enabled
POR	enabled
STCU	enabled
FCCU failure	application dependent
Functional reset	disabled

Table 858 summarizes the MC actions corresponding to each reset generated by FCCU.

Table 858. 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.

60.5.4 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.

60.5.5 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 846* and *Figure 847*, 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 846. RF (ALARM state) recovery (a)

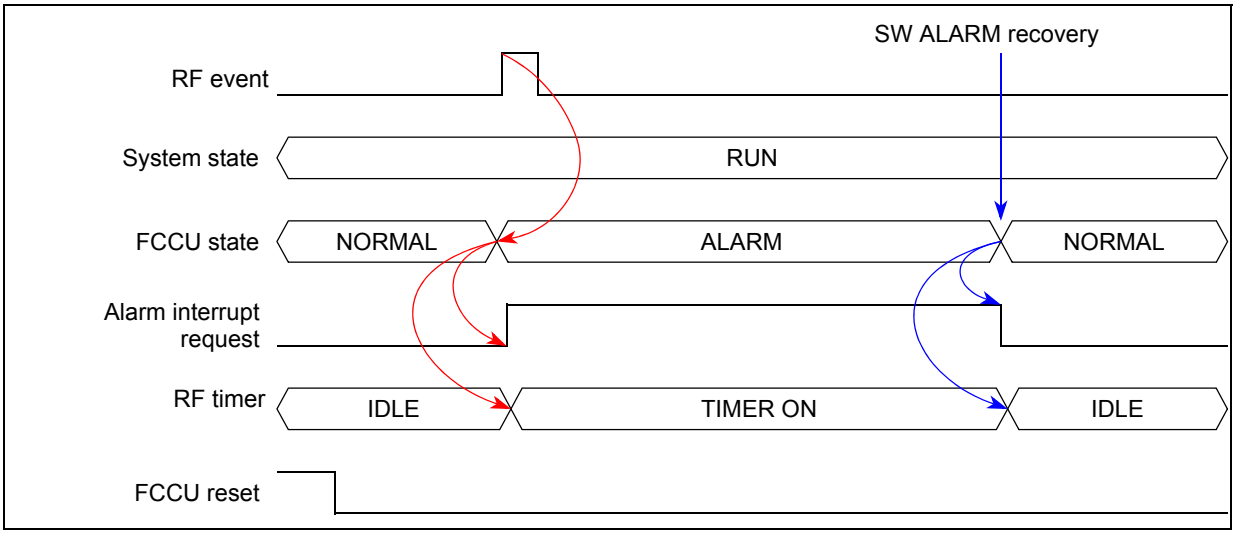
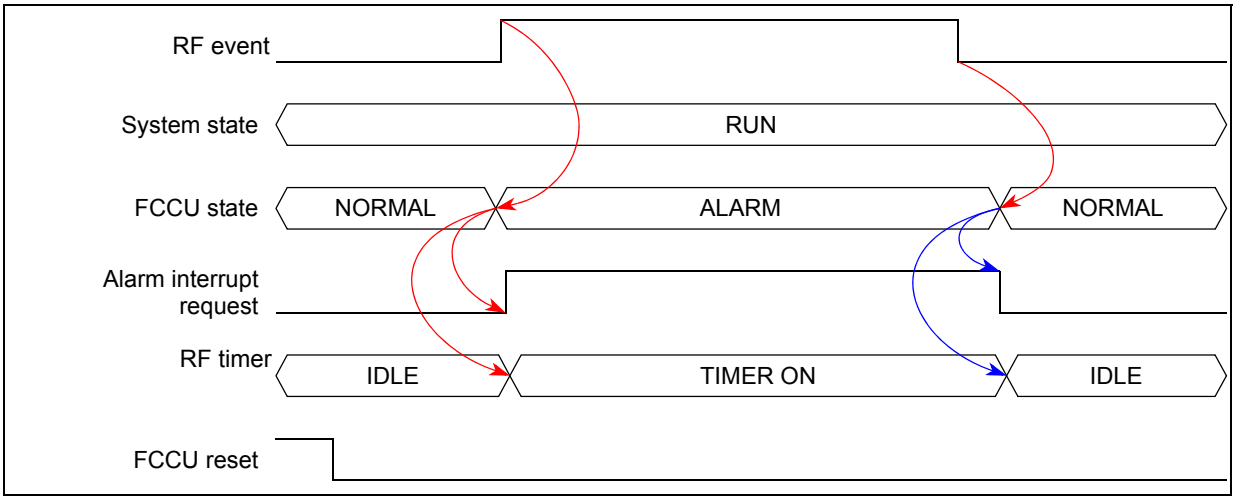


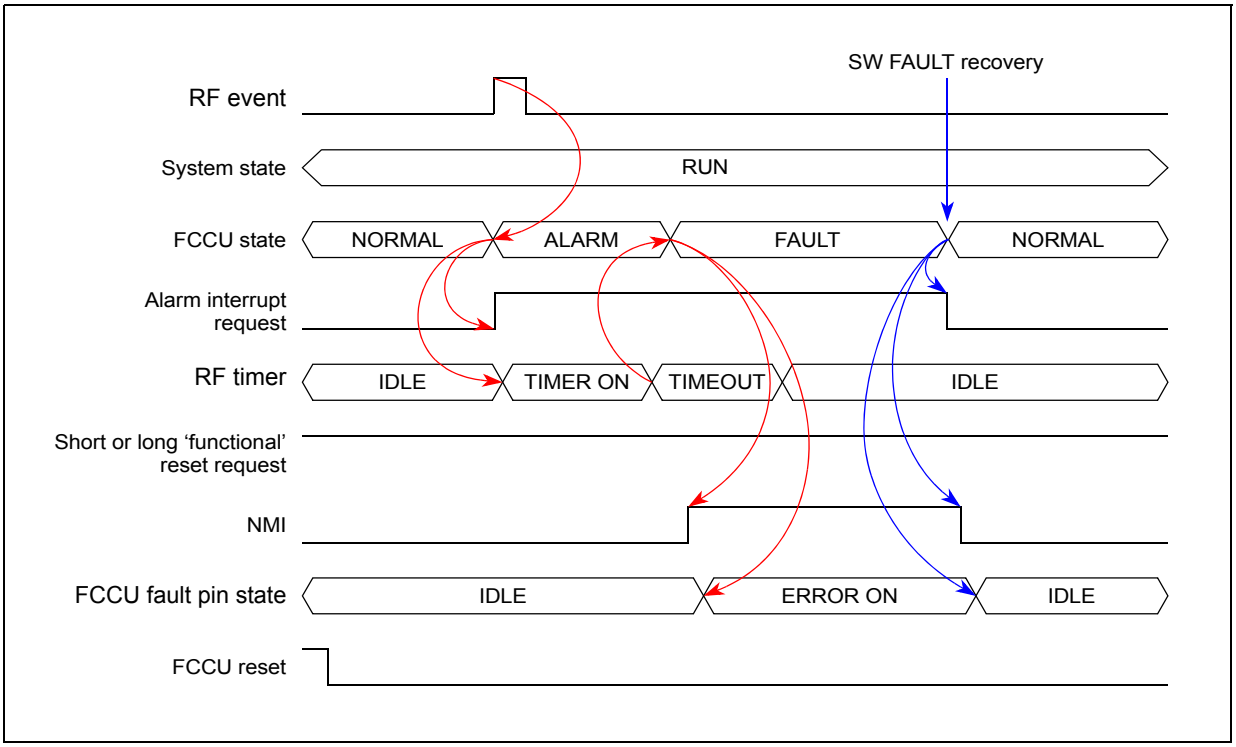
Figure 847. RF (ALARM state) recovery (b)



A typical sequence related to a RF management (ALARM → FAULT state), see *Figure 848*, 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 848. RF(ALARM -> FAULT state) recovery



60.5.6 NMI/WKPU interface

The wake-up interface includes the following signals:

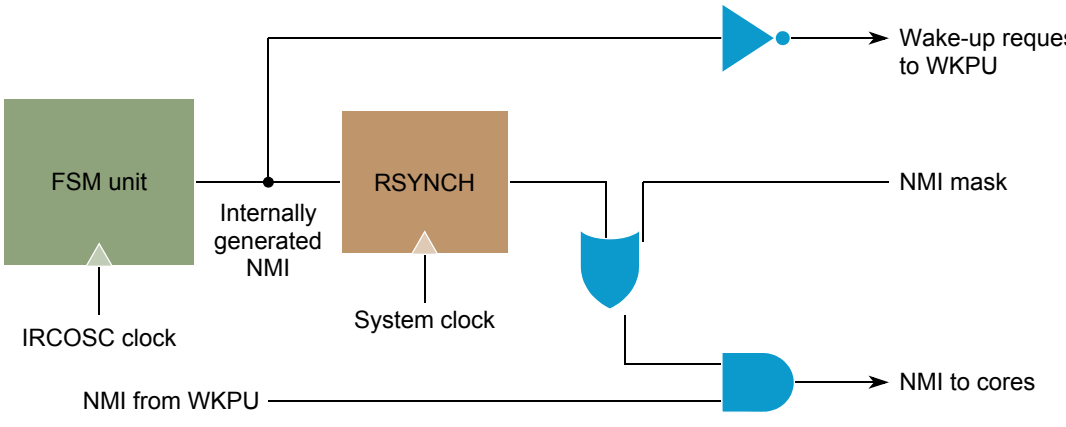
- Nonmaskable interrupt: NMI interrupt source generated by the WKPU. It is synchronous with the system clock and active low. A logical AND interconnects the NMI sources (internally generated by FCCU and by WKPU) in order to provide a single NMI output.
- Wake-up indication: wake-up indication provided to the WKPU to bring the system out of dormant state in case of critical conditions. It is synchronous to the IRCOSC clock and active high. It is asserted when the FCCU is in FAULT state concurrently with NMI reaction enabled for the corresponding error source, responsible for the fault state.

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 849. NMI/WKUP scheme



Note: All NMI signals are shown active-low.

60.5.7 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, in order to generate fake faults directly in the FAULT root, 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 850*. 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 *Table 17: FCCU failure inputs* 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 851*. 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 850. FAULT injection

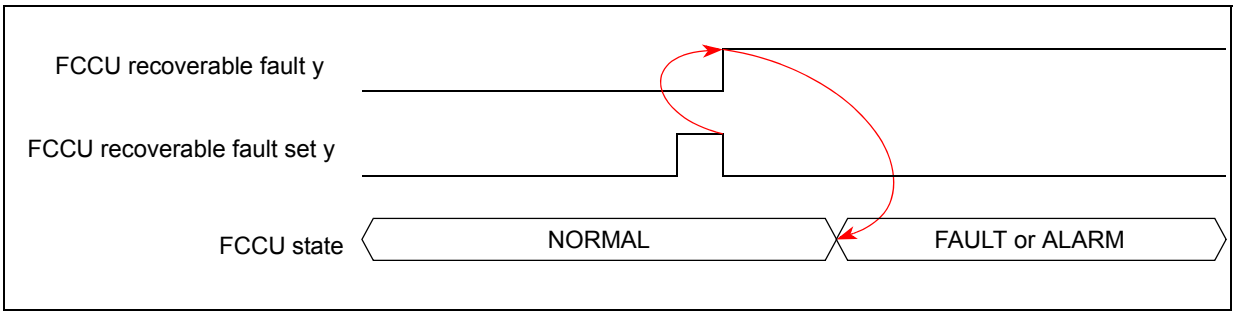
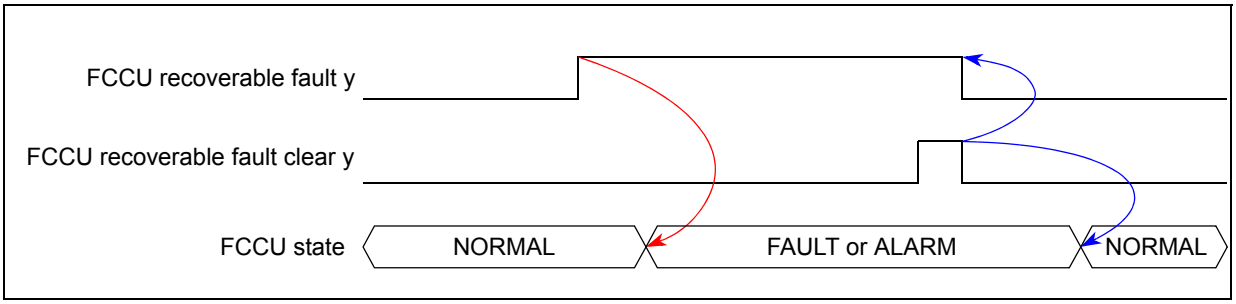


Figure 851. FAULT clearing



60.5.8 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 852. STCU-FCCU (case a)

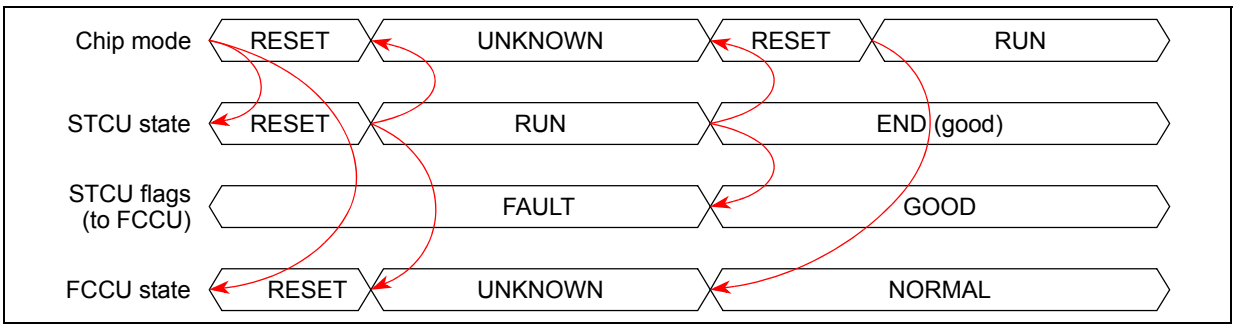


Figure 853. STCU-FCCU (case b)

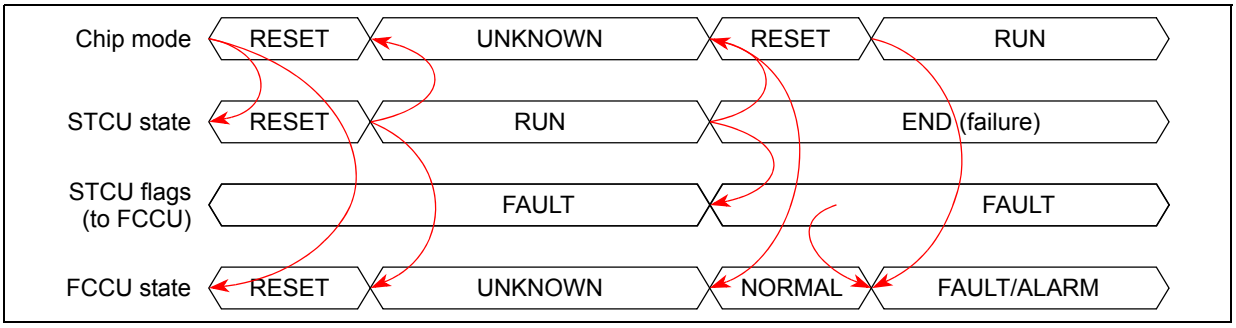
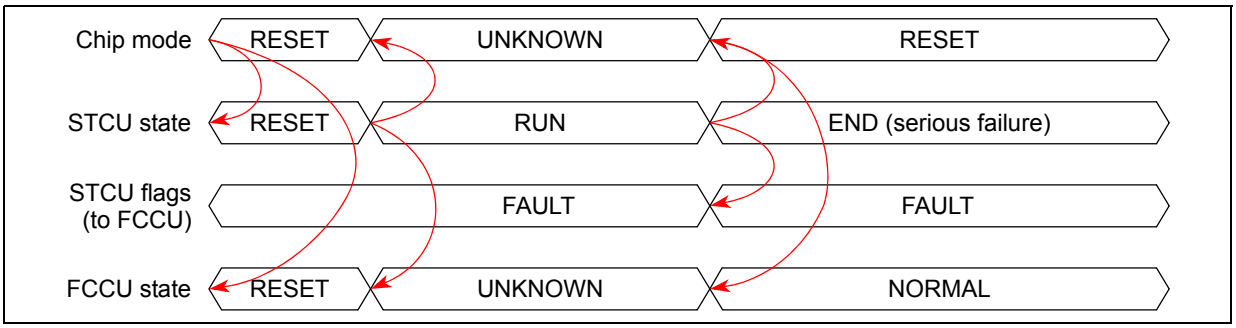


Figure 854. STCU-FCCU (case c)



60.5.9 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, destructive 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 859. FCCU error pin behavior on STATE transitions(1)

Fault signaling state	Transition from NORMAL to FAULT state with Fault disabled | enabled			NORMAL+CONFIG states			Transition from NORMAL to ALARM state 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.

60.5.9.1 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 860* and the related timing diagram is given in *Figure 855* and *Figure 856*.

Table 860. 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

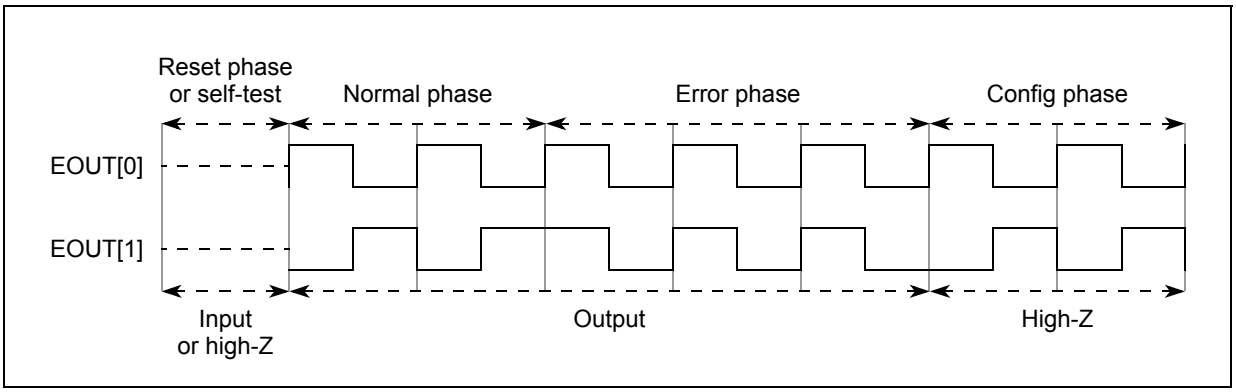
1. Final value depends on the SoC settings at pad level.
2. Till FCCU_SET_AFTER_RESET is written, the Error indicating pads are driven to High impedance from FCCU's side. That bit is write-able only after the FCCU enters CONFIG state. Hence on entry to CONFIG state, user may still see high impedance.

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”(x).

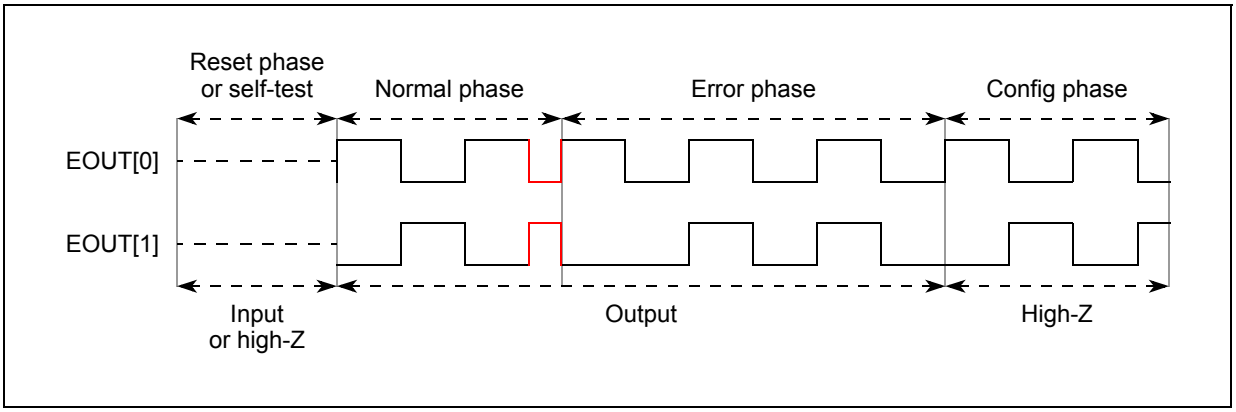
Note: Figure 855 and Figure 856 are formatted to display the behavior in all four phases (reset, normal, error, and config), not to imply transitions between one phase to another. In particular, transition from error phase to config phase is not possible.

Figure 855. Dual rail protocol (slow switching mode)



x. Actual value depends on the SOC settings at pad level

Figure 856. Dual rail protocol (fast switching mode)



60.5.9.2 Time switching protocol

The encoding scheme is given in *Table 861* and the related timing diagram is given in *Figure 857*.

Table 861. 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.

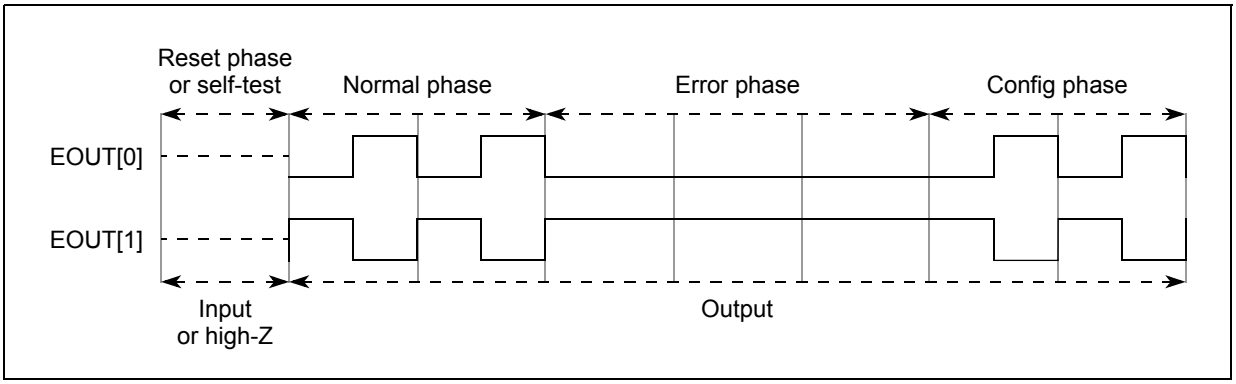
In Time Switching mode the second output (FCCU Error output pin[1]) is the inverted signal of first output (FCCU Error output pin[0]).

In the RESET phase the output pins are set as “high impedance”(y).

Note: Figure 857 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.

y. Actual value depends on the SOC settings at pad level

Figure 857. Time switching protocol



60.5.9.3 Bi-stable protocol

The encoding scheme is given in *Table 862* and the related timing diagram is given in *Figure 858*.

Table 862. 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.

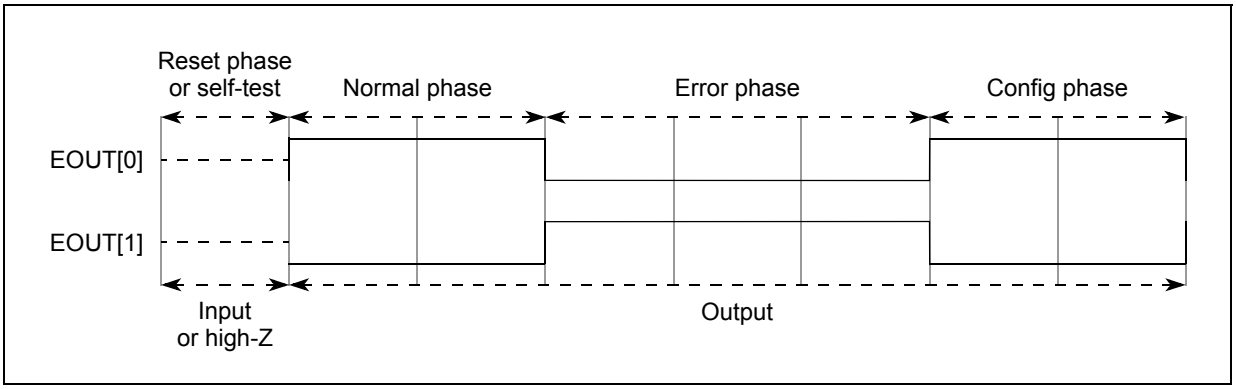
In the FAULT state, the faulty logical state is indicated. In NORMAL or ALARM state, “no-faulty” state is indicated. In Bi-stable mode the second output (FCCU Error output[1]) is the inverted signal of first output (FCCU Error output[0]).

In the RESET phase the output pins are set as “high impedance”(z).

Note: Figure 858 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.

z. Actual value depends on the SOC settings at pad level

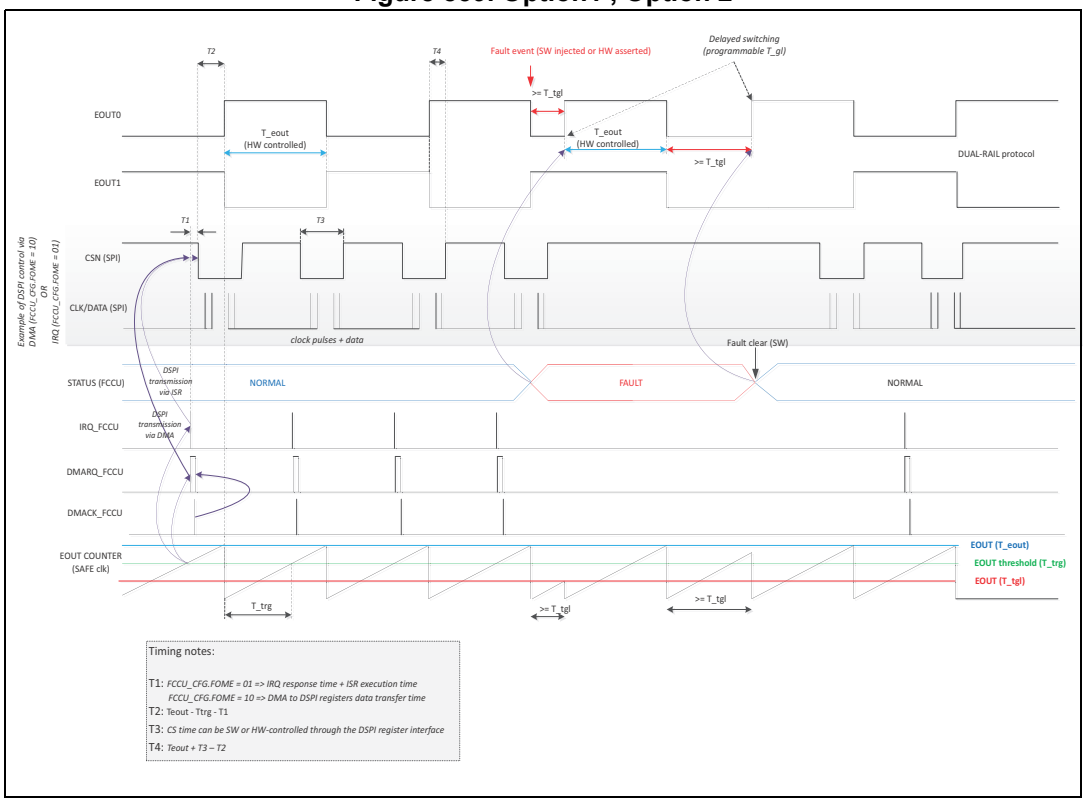
Figure 858. Bi-stable protocol



60.5.9.4 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 859*).

Figure 859. Option1 , Option 2

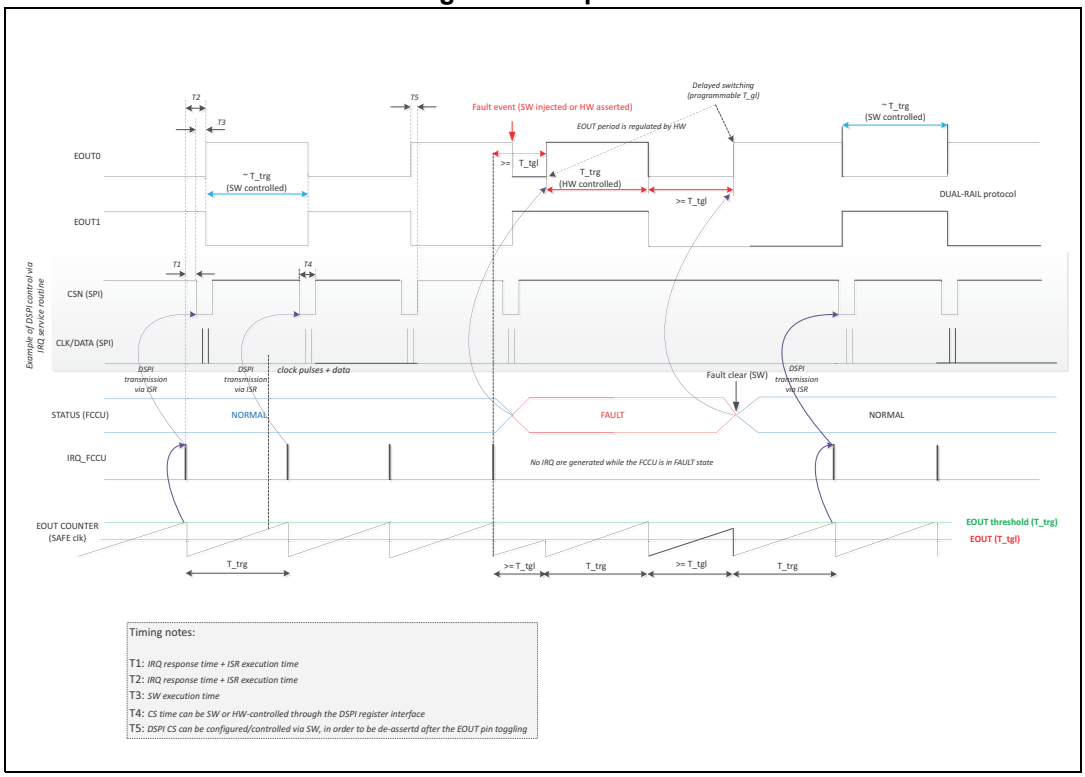


60.5.9.5 Option 3

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 860* 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.

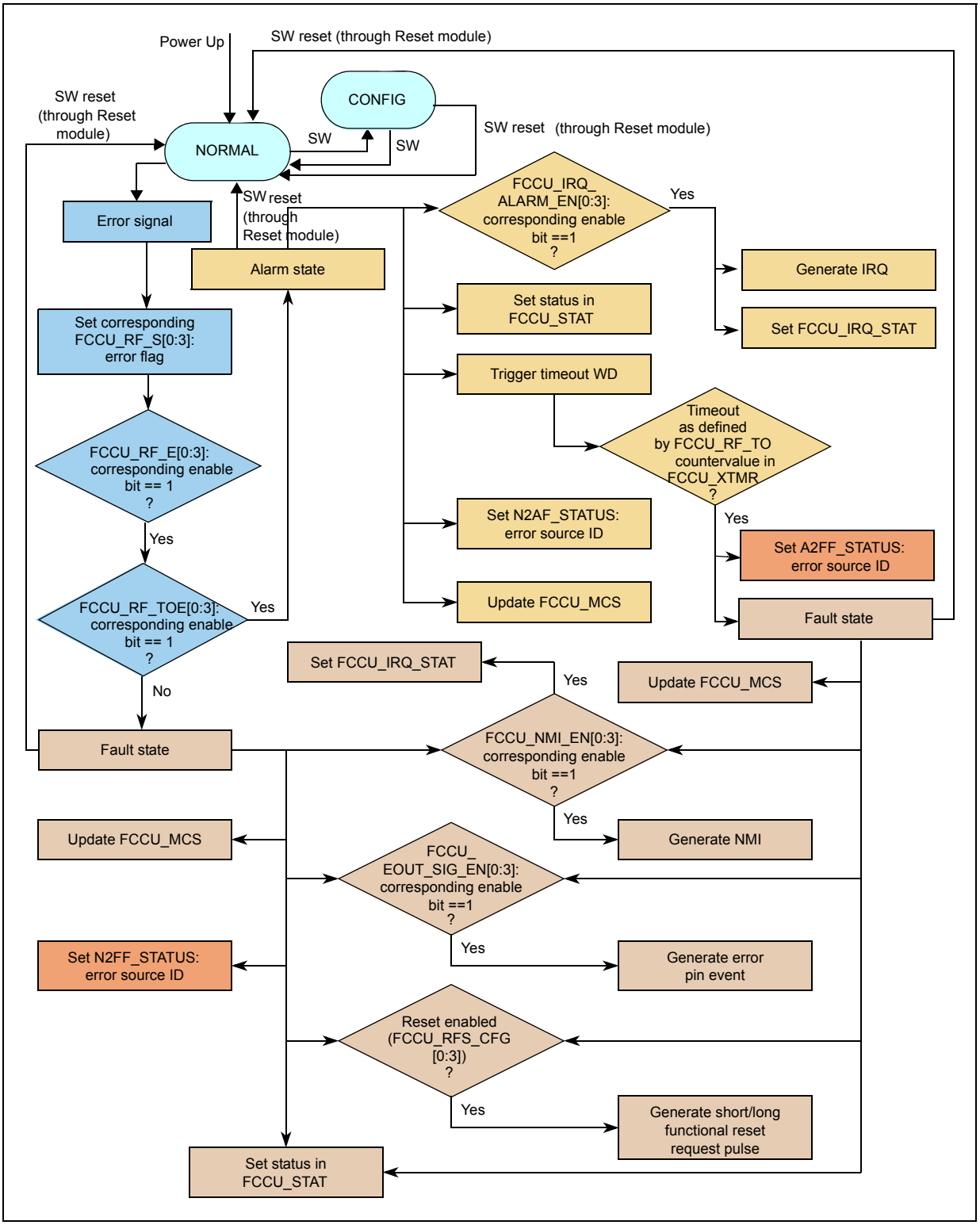
Figure 860. Option 3



60.5.10 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 and/or FCCU_SIG_EN being set, it generates a NMI and/or signals error out to the external world. FCCU FSM status is also captured in FCCU_STAT.

Figure 861. Error signal flow



60.6 Register description

The FCCU registers are listed in *Table 863*. Any address offset not explicitly mentioned in *Table 863* is reserved.

All the registers accessible in write mode only in CONFIG state (last column of the *Table 863*) 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 863. FCCU memory map

Address offset	Register	Access(1)	Register classification(2)	Location
0x000	FCCU Control Register (FCCU_CTRL)	R/W always	Saf-Lat	*Section 60.6.1*
0x004	FCCU CTRL Key Register (FCCU_CTRLK)	W always	Saf-Relv	*Section 60.6.2*
0x008	FCCU Configuration Register (FCCU_CFG)	R always; W in CONFIG state only	Saf-Lat	*Section 60.6.3*
0x01C	FCCU RF Configuration Register 0 (FCCU_RF_CFG0)	R always; W in CONFIG state only	Saf-Lat	*Section 60.6.4*
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 863. FCCU memory map

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 60.6.5*
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 60.6.6*
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 60.6.7*
0x094	FCCU RF Enable Register 0 (FCCU_RF_E0)	R always; W in CONFIG state only	Saf-Lat	*Section 60.6.8*
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 60.6.9*
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 60.6.10*

Table 863. FCCU memory map

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 60.6.11*
0x0BC	FCCU IO Control Register (FCCU_EINOUT)	R/W always	Saf-Lat	*Section 60.6.12*
0x0C0	FCCU Status Register (FCCU_STAT)	R only	Saf-Lat	*Section 60.6.13*
0x0C4	FCCU NA Freeze Status Register (N2AF_STATUS)	R/W always	No-Saf	*Section 60.6.14*
0x0C8	FCCU AF Freeze Status Register (A2FF_STATUS)	R/W always	No-Saf	*Section 60.6.15*
0x0CC	FCCU NF Freeze Status Register (N2FF_STATUS)	R/W always	No-Saf	*Section 60.6.16*
0x0D0	FCCU FA Freeze Status Register (F2A_STATUS)	R/W always	No-Saf	*Section 60.6.17*
0x0DC	FCCU RF Fake Register (FCCU_RFF)	R/W always	No-Saf	*Section 60.6.18*
0x0E0	FCCU IRQ Status Register (FCCU_IRQ_STAT)	R/W always	Saf-Relv	*Section 60.6.19*
0x0E4	FCCU IRQ Enable Register (FCCU_IRQ_EN)	R/W always	Saf-Lat	*Section 60.6.20*
0x0E8	FCCU XTMR Register (FCCU_XTMR)	R/W always	No-Saf	*Section 60.6.21*
0x0EC	FCCU MCS Register (FCCU_MCS)	R only	Saf-Lat	*Section 60.6.22*
0x0F0	FCCU Transient Lock Register (FCCU_TRANS_LOCK)	W always	Saf-Lat	*Section 60.6.23*
0x0F4	FCCU Permanent Lock Register (FCCU_PERMNT_LOCK)	W always	Saf-Lat	*Section 60.6.24*
0x0F8	FCCU Delta T Register(FCCU_DELTA_T)	R always; W in CONFIG state only	Saf-Lat	*Section 60.6.25*
0x0FC	FCCU IRQ Alarm Enable Register 0 (FCCU_IRQ_ALARM_EN0)	R always; W in CONFIG state only	Saf-Lat	*Section 60.6.26*
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 863. FCCU memory map

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 60.6.27*
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 60.6.28*
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

60.6.1 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 862. FCCU Control Register (FCCU_CTRL)

Table 864. 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 FCCU's FSM remains in the same state.
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.
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 60.6.21*) [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

60.6.2 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.

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.

Offset: 0x004 Access: User write-only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R																
W	CTRLK															
Reset	0	0	0	0	0	0	0	0	0	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	CTRLK															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 863. FCCU CTRL Key Register (FCCU_CTRLK)

Table 865. FCCU_CTRLK field descriptions

Field	Description
0:31 CTRLK	Control register key 0x913756AF Key for the operation OP1 0x825A132B Key for the operation OP2

60.6.3 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.

Offset: 0x008 Access: User read/write(1)

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0							FCCU_SET_AFTER_RESET	FCCU_SET_CLEAR		0	0	Reserved			
W	TGL															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	FOPE	FOME		OD	0	SM	PS	FOM			FOP					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0(2)	0	0	0

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 864. FCCU Configuration Register (FCCU_CFG)

Table 866. 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.

Table 866. FCCU_CFG field descriptions(Continued)

Field	Description
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.
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.

Table 866. FCCU_CFG field descriptions(Continued)

Field	Description
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. FOM bit fields do not update to default state on software reset through RGM_PRST5 register.
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 859*.
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 859*.
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 eTMR 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 860*

60.6.4 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 868* for register offset to channel number relationship

Figure 865. FCCU RF Configuration Register *n* (FCCU_RF_CFG*n*)

Table 867. 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 868 shows FCCU RF configuration register channels.

Table 868. 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]

Table 868. FCCU RF configuration register channels(Continued)

Address offset	Register name	Channel range (x) (bit location [0:31])
0x024	FCCU_RF_CFG2	RFC[95:64]
0x028	FCCU_RF_CFG3	RFC[127:96]

60.6.5 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 870* for register offset to channel number relationship.

Figure 866. FCCU RFS Configuration Register *n* (FCCU_RFS_CFG*n*)

Table 869. 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 870 shows FCCU **RFS** configuration register channels.

Table 870. 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]

60.6.6 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 60.5.7*).

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 872* for register offset to channel number relationship. Refer to for the errors description.

Figure 867. FCCU RF Status register *n* (FCCU_RF_S*n*)

Table 871. 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 872 shows FCCU **RF** status register channels.

Table 872. 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]

60.6.7 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															
Reset	0	0	0	0	0	0	0	0	0	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															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 868. FCCU RF Key register (FCCU_RFK)

Table 873. FCCU_RFK field descriptions

Field	Description
0:31 RFK	recoverable fault key = 0xAB34_98FE

60.6.8 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 875* for register offset to channel number relationship.

Figure 869. FCCU RF Enable Register *n* (FCCU_RF_E*n*)

Table 874. 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 875 shows FCCU **RF_E*n*** enable register channels.

Table 875. 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]

60.6.9 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 877* for register offset to channel number relationship.

Figure 870. FCCU RF Timeout Enable Register *n* (FCCU_RF_TOE*n*)

Table 876. 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 877 shows FCCU **RF** timeout enable register channels.

Table 877. 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]

60.6.10 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 871. FCCU RF Timeout Register (FCCU_RF_TO)

Table 878. FCCU_RF_TO field descriptions

Field	Description
0:31 TO	Recoverable fault timeout Timeout = (TO) x $T_{RC16MHz}$

60.6.11 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 863*. 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 872. FCCU CFG Timeout Register (FCCU_CFG_TO)

Table 879 shows FCCU_CFG_TO field descriptions.

Table 879. 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

60.6.12 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 880 shows Bi-stable encoding.

Table 880. 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 eTMR 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 860*.

Figure 873. FCCU IO Control Register (FCCU_EINOUT)

Table 881. FCCU_EINOUT field descriptions

<table>
<tr><th>Field</th><th>Description</th></tr>
<tr><td>26
EIN1</td><td>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.</td></tr>
<tr><td>27
EIN0</td><td>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.</td></tr>
<tr><td>30:31
EOUT</td><td>
<table>
<tr><th colspan="2">Register</th><th colspan="2">Signals</th><th rowspan="2">Description</th></tr>
<tr><th>EINOUT.
EOUT0</th><th>EINOUT.
EOUT0</th><th>FCCU_EOUT1</th><th>FCCU_EOUT0</th></tr>
<tr><td>0</td><td>0</td><td>1 xor PS</td><td>0 xor PS</td><td>Opposite phase</td></tr>
<tr><td>0</td><td>1</td><td>0 xor PS</td><td>1 xor PS</td><td>Opposite phase</td></tr>
<tr><td>1</td><td>0</td><td>0 xor PS</td><td>0 xor PS</td><td>In phase</td></tr>
<tr><td>1</td><td>1</td><td>1 xor PS</td><td>1 xor PS</td><td>In phase</td></tr>
</table>
</td></tr>
</table>

60.6.13 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 874. FCCU Status Register (FCCU_STAT)

Table 882. FCCU_STAT field descriptions

Field	Description
26:27 PhysicErrorPin	Physical Error Pin's state PhysicErrorPin[0] corresponds to error pin 0 and PhysicErrorPin[1] 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 882. 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

60.6.14 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 875. FCCU NA Freeze Status Register (N2AF_STATUS)

Table 883. 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

60.6.15 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 876. FCCU AF Freeze Status Register (A2FF_STATUS)

Table 884. A2FF_STATUS field descriptions

Field	Description
22:23 AF_SRC	Fault source 00 No fault 01 Reserved 10 Recoverable fault 11 Multiple and/or recoverable 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

60.6.16 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 877. FCCU NF Freeze Status Register (N2FF_STATUS)

Table 885. 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

60.6.17 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 878. FCCU FA Freeze Status Register (F2A_STATUS)

Table 886. 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

60.6.18 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 60.5.7*. 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 879. FCCU RF Fake Register (FCCU_RFF)

Table 887. 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.

60.6.19 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 859*.
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 eTMR 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 860*.

Figure 880. FCCU IRQ Status Register (FCCU_IRQ_STAT)

Table 888. 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 888. 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 859*.

60.6.20 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 881. FCCU IRQ Enable Register (FCCU_IRQ_EN)

Table 889. 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

60.6.21 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.

ETMR 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.

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 891 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															
W																
Reset	0	0	0	0	0	0	0	0	0	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															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 882. FCCU XTMR Register (FCCU_XTMR)

Table 890. FCCU_XTMR field descriptions

Field	Description
0:31 XTMR	Alarm/Watchdog/Safe request timer The current timer value is measured in IRCOSC clock cycles. These bits can be read by the software.

Table 891. 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
ETMR	0x0000_0000	0x0000_0000	0x0000_0000	Running

60.6.22 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 883. FCCU MCS Register (FCCU_MCS)

Table 892. 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 892. 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.

60.6.23 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 884. FCCU Transient Register (FCCU_TRANS_LOCK)

Table 893. 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

60.6.24 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 885. FCCU Permanent Lock Register (FCCU_PERMNT_LOCK)

Table 894. FCCU_PERMNT_LOCK field descriptions

Field	Description
24:31 PERMNTKEY	Transistion Locking Key value 0xFF Configuration Locked Any other value: No Change

60.6.25 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 when the FCCU is 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													
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 eTMR 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 860*

Figure 886. FCCU Delta T Register (FCCU_DELTA_T)

Table 895. 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	Value of Delta_T (in microseconds) Max. value can be 10,000 µs (10 ms)

60.6.26 FCCU IRQ Alarm Enable register *n* (FCCU_IRQ_ALARM_EN*n*)

These registers enable the corresponding IRQ alarm.

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 W	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)
Reset	0	0	0	0	0	0	0	0	0	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	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)
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. See *Table 897* for register offset to channel number relationship.

Figure 887. FCCU IRQ Alarm Enable register *n* (FCCU_IRQ_ALARM_EN*n*)

Table 896. 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 897 shows FCCU IRQ alarm enable register channels.

Table 897. 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]

60.6.27 FCCU NMI Enable registers *n* (FCCU_NMI_EN*n*)

These registers enable the NMI.

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 899* for register offset to channel number relationship.

Figure 888. FCCU NMI Enable register *n* (FCCU_NMI_EN*n*)

Table 898. 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 899 shows FCCU NMI enable register channels.

Table 899. 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]

60.6.28 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.

Offsets: 0x11C + 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	EOUTEN31(1)	EOUTEN30(1)	EOUTEN29(1)	EOUTEN28(1)	EOUTEN27(1)	EOUTEN26(1)	EOUTEN25(1)	EOUTEN24(1)	EOUTEN23(1)	EOUTEN22(1)	EOUTEN21(1)	EOUTEN20(1)	EOUTEN19(1)	EOUTEN18(1)	EOUTEN17(1)	EOUTEN16(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	EOUTEN15(1)	EOUTEN14(1)	EOUTEN13(1)	EOUTEN12(1)	EOUTEN11(1)	EOUTEN10(1)	EOUTEN9(1)	EOUTEN8(1)	EOUTEN7(1)	EOUTEN6(1)	EOUTEN5(1)	EOUTEN4(1)	EOUTEN3(1)	EOUTEN2(1)	EOUTEN1(1)	EOUTEN0(1)
W																
Reset	The reset value is chip-specific; see the chapter that describes how modules are configured and connected.															

1. See *Table 901* for register offset to channel number relationship.

Figure 889. FCCU EOUT Signaling Enable register *n* (FCCU_EOUT_SIG_EN*n*)

Table 900. 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 for bistable protocol.

Table 901 shows FCCU EOUT signaling enable register channels.

Table 901. 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]

60.7 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.

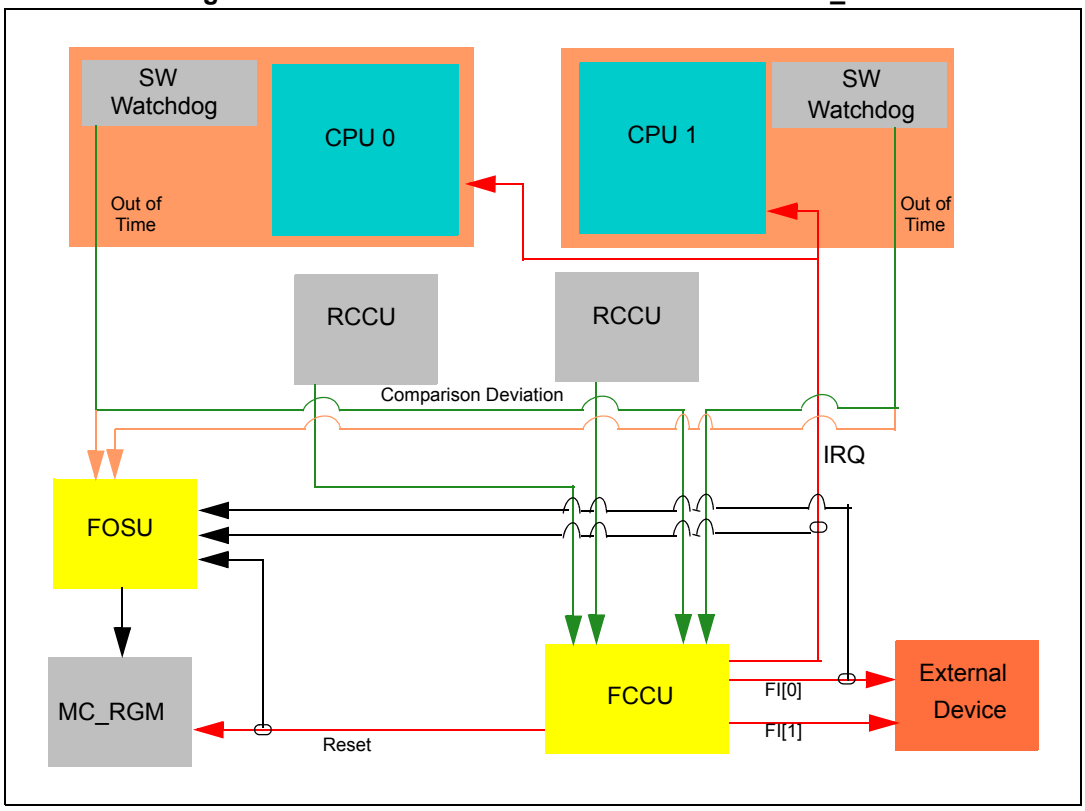
The FOSU contains a timer with a duration of FOSU_COUNT,driven by the IRCOSC.The timer is initialized and started on any captured, enabled fault. While the timer is running, any subsequent captured fault will neither restart nor reinitialize the timer. The timer is stopped when the FCCU shows any of 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 890. 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.

60.8 Use cases and limitations

60.8.1 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

60.8.2 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.

61 Self-Test Control Unit (STCU2)

61.1 Introduction

Self-Test Control Unit (STCU2) is a comprehensive programmable hardware module that controls the Self-Test sequence applied both during the Off-line and/or On-line 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 Off-line 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 Off-line Self-Test and to load or check the execution of the On-line 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 time-out.
- Direct PLL control during the Off-line Self-Test sequence.
- Two sets of registers to separately collect the LBIST or MBIST results during Off-line and On-line Self-Test.

To increase the security of this module, a different Security Key sequence is also required during Off-line and On-line 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 time-out 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).

61.2 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)
- On-line hardware Self-Test sequence abort capability.
- Programmable Internal clock prescaler to reduce Internal and MBIST or LBIST TCK clocks.
- Programmable PLL direct control during Off-line Self-Test sequence.
- PLL Lock signal monitoring during Off-line and On-line Self-Test.
- Programmable WDG Timer for Self-Check MBIST or LBIST Time Execution.
- Hard-coded WDG time-out to flag Off-line initialization errors.
- Hard-coded WDG time-out to Auto-lock STCU2 write access after Double-Key sequence.
- LBIST and SRAM or ROM MBIST Off-line status flags.
- STCU2 internal errors Off-line status flag.
- LBIST and SRAM or ROM MBIST On-line status flags
- STCU2 internal errors On-line Status flag.
- On-line hardware abort status flag
- LBIST and SRAM or ROM MBIST Off-line or On-line programmable failure severity (UF/RF).
- STCU2 internal errors Off-line or On-line 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 time-out 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 time-out is detected after software application writes Double Security Key sequence (available when STCU_CFG[WRP] bit is active).

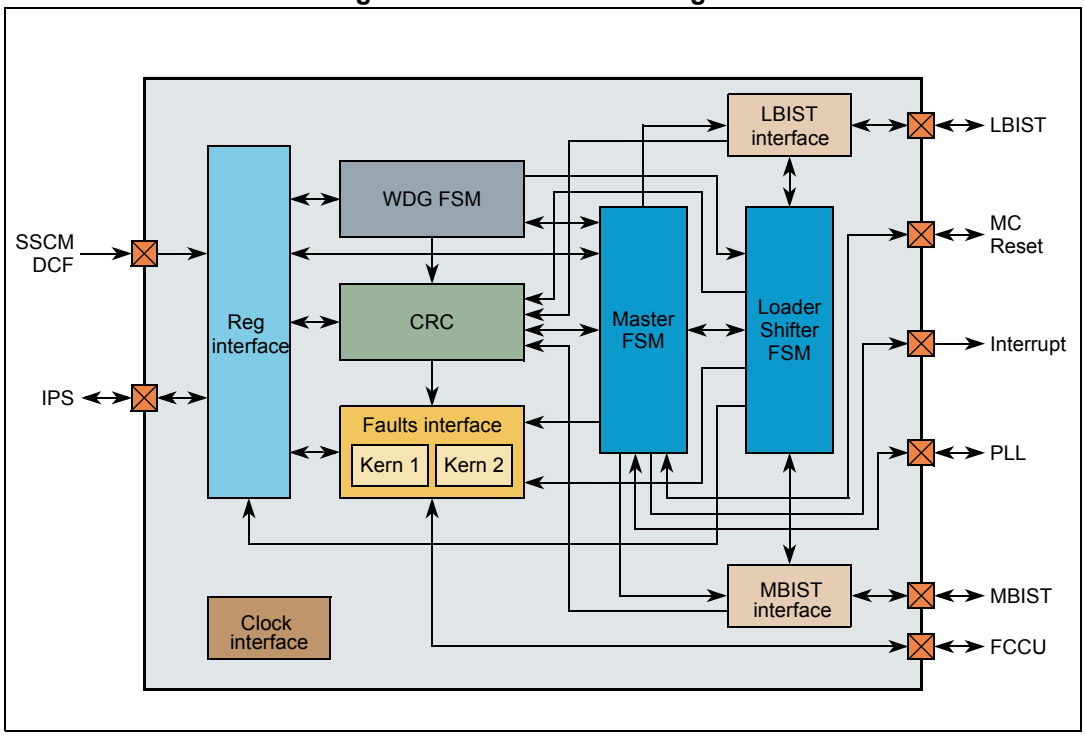
After completion of On-line Self-Test (MBIST or LBIST) a functional reset is generated.

So no partition level functional behavior is guaranteed once On-line Self-Test is triggered.

61.3 Block diagram

A top level diagram of the STCU2 module is given in *Figure 891*.

Figure 891. STCU2 block diagram



The STCU2 module includes the following sub-modules:

- **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 time-out.
  - 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.

61.3.1 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 Off-line Self-Test is executed.
- The IPS access on some specific registers (see *Section 61.5: Register description*) 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 rw 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/hword/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 Off-line on registers where writes are only allowed during the On-line 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 61.5: Register description*.

61.4 Functional description

61.4.1 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 time-outs in case of incorrect STCU2 programming.

61.4.2 Reset management

The asynchronous reset line initializes the STCU2 module status, forcing the Off-line 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.

61.4.2.1 Off-line reset generation

At the end of the Off-line Self-Test execution, the STCU2 raises the global functional reset.

61.4.2.2 On-line reset generation (only MBIST)

The On-line execution of the MBIST does not generate any reset from STCU2 at the end of the execution. This is becasue the software application has to manage all the possible consequences respective to the write operations peformed by the MBIST execution other than prevent any read/writ operation during the Self-Test execution. If necessary, the softare must also restore the memory content (only for RAMs) at the end of the Self-Test execution.

61.4.2.3 On-line Reset generation (LBIST enabled)

At the end of the execution, the On-line execution of the LBIST generates a global functional reset.

61.4.3 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 61.7.4: Design implementation information*.

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 00h, the second on 01h, and so on. The first MBIST is mapped at the address 10h, the second on 11h 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 61.5.22: STCU2 LBIST Control Register n (STCU_LB_CTRLn)* and *Section 61.5.30: STCU2 MBIST Control Register n (STCU_MB_CTRLn)*.

61.4.4 ABORT management

The STCU2 module provides On-line Self-Test execution abort. When the abort is detected, the On-line 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 61.4.2: Reset management*, regarding reset specific management when abort is detected.

61.4.4.1 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 On-line Self-Test execution run.

61.4.5 PLL interface

The STCU2 module is able to directly control the PLL during the Off-line Self-Test operations depending on the status of the STCU_RUN[MBPLLEN] and

STCU_RUN[LBPLLEN] as described in *Section 61.5.1: STCU2 Run Register (STCU_RUN)*.

When STCU_RUN[RUN] is set to start the Off-line 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.

61.4.6 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.

61.4.7 Watchdogs

The STCU2 implements three different Watchdogs to ensure that operations are finished on time.

61.4.7.1 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 time-out flags the incorrect behavior. See *Section 61.5.5: STCU2 Watchdog Register Granularity (STCU_WDG)* for more details.

61.4.7.2 LBIST and MBIST scheduling

The LBIST and MBIST execution time is configured as described in *Section 61.5.5: STCU2 Watchdog Register Granularity (STCU_WDG)* to account for the overall execution time of the Self-Test sequence. If the selected LBISTs or MBISTs are not yet completed:

- During an Off-line 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 On-line 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 On-line or Off-line session and the time-out 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 61.4.2: Reset management*.

61.4.7.3 Write access time-out

The access on the STCU2 registers during the Self-Test configuration phase (both On-line or Off-line) 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.

61.4.8 CRC

The CRC block is used to increase the intrinsic coverage of the STCU2 module and implements a 32-bit ethernet protocol polynomy 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.

61.4.9 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.

61.5 Register description

The STCU2 registers are listed in *Table 902*.

Table 902. STCU2 register set

Address offset	Register Name	Location
0x0000	STCU2 Run Register (STCU_RUN)	*Section 61.5.1*
0x0004–0x0007	Reserved	
0x0008	STCU2 SK Code Register (STCU_SKC)	*Section 61.5.2*
0x000C	STCU2 Configuration Register (STCU_CFG)	*Section 61.5.3*
0x0010	STCU2 PLL Configuration Register (STCU_PLL_CFG)	*Section 61.5.4*
0x0014	STCU2 Watchdog Register Granularity (STCU_WDG)	*Section 61.5.5*
0x0018–0x001B	Reserved	
0x001C	STCU2 CRC Expected Status Register (STCU_CRCE)	*Section 61.5.6*
0x0020	STCU2 CRC Read Status Register (STCU_CRCR)	*Section 61.5.7*

Table 902. STCU2 register set(Continued)

Address offset	Register Name	Location
0x0024	STCU2 Error Register (STCU_ERR_STAT)	*Section 61.5.8*
0x0028	STCU2 Error FM Register (STCU_ERR_FM)	*Section 61.5.9*
0x002C	STCU2 Off-line LBIST Status Register (STCU_LBS)	*Section 61.5.10*
0x0030	STCU2 Off-line LBIST End Flag Register (STCU_LBE)	*Section 61.5.11*
0x0034–0x003B	Reserved	
0x003C	Reserved	
0x0040	STCU2 LBIST Unrecoverable FM Register (STCU_LBUFM)	*Section 61.5.12*
0x0044	STCU2 Off-line MBIST Status Low Register (STCU_MBSL)	*Section 61.5.13*
0x0048	STCU2 Off-line MBIST Status Medium Register (STCU_MBSM)	*Section 61.5.14*
0x004C	STCU2 Off-line MBIST Status High Register (STCU_MBSH)	*Section 61.5.15*
0x0050	STCU2 Off-line MBIST End Flag Low Register (STCU_MBEL)	*Section 61.5.16*
0x0054	STCU2 Off-line MBIST End Flag Medium Register (STCU_MBEM)	*Section 61.5.17*
0x0058	STCU2 Off-line MBIST End Flag High Register (STCU_MBEH)	*Section 61.5.18*
0x005C–0x0073	Reserved	
0x0074	STCU2 MBIST Unrecoverable FM Low Register (STCU_MBUFML)	*Section 61.5.19*
0x0078	STCU2 MBIST Unrecoverable FM Medium Register (STCU_MBUFMM)	*Section 61.5.20*
0x007C	STCU2 MBIST Unrecoverable FM High Register (STCU_MBUFMH)	*Section 61.5.21*
0x0100+n*0x40 n=0 to (NLBIST-1)(1)	STCU2 LBIST Control Register (STCU_LB_CTRL*n*)	*Section 61.5.22*
0x0104+n*0x40 n=0 to (NLBIST-1)(1)	STCU2 LBIST PC Stop Register (STCU_LB_PCS*n*)	*Section 61.5.23*
0x0108+n*0x40 n=0 to (NLBIST-1)(1)	STCU2 LBIST PRPG Low Register (STCU_LB_PRPGL*n*)	*Section 61.5.24*
0x010C+n*0x40 n=0 to (NLBIST-1)(1)	STCU2 LBIST PRPG High Register (STCU_LB_PRPGH*n*)	*Section 61.5.25*
0x0110+n*0x40 n=0 to (NLBIST-1)(1)	STCU2 Off-line LBIST MISR Expected Low Register (STCU_LB_MISREL*n*)	*Section 61.5.26*
0x0114+n*0x40 n=0 to (NLBIST-1)(1)	STCU2 Off-line LBIST MISR Expected High Register (STCU_LB_MISREH*n*)	*Section 61.5.27*
0x0118+n*0x40 n=0 to (NLBIST-1)(1)	STCU2 Off-line LBIST MISR Read Low Register (STCU_LB_MISRRL*n*)	*Section 61.5.28*
0x011C+n*0x40 n=0 to (NLBIST-1)(1)	STCU2 Off-line LBIST MISR Read High Register (STCU_LB_MISRRH*n*)	*Section 61.5.29*
0x0120–0x05FF	Reserved	
0x0600+n*0x4 n=0 to (NMCUT-1)(1)	STCU2 MBIST Control Register (STCU_MB_CTRL*n*)	*Section 61.5.30*

1. See LBIST and NMCUT values in *Table 98: DCF client list* of *Chapter 9: Device Configuration Format (DCF) Records*.

61.5.1 STCU2 Run Register (STCU_RUN)

The STCU_RUN register defines the RUN bit to start the Off-line 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 Off-line 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 892. STCU2 Run Register (STCU_RUN)

Table 903. STCU_RUN field descriptions

Field	Description
21 BYP	Bypass mode 0 Off-line Self-Test is active. If the STCU_RUN[RUN] bit is not set before the hard-coded WDG Time-out, the WDTO error is generated in the STCU_ERR_STAT register. 1 Off-line 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	Off-line MBIST with PLL Enabled 0 Off-line MBIST is executed without using the on-chip PLL. 1 Off-line MBIST is executed enabling the on-chip PLL control interface selecting the parameters defined in STCU_PLL_CFG register.
23 LBPLLEN	Off-line LBIST with PLL Enabled 0 Off-line LBIST is executed without using the on-chip PLL. 1 Off-line 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 Off-line Self-Test procedure has been completed. 0 Idle 1 Off-line Self-Test procedure is running

61.5.2 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 Off-line or On-line 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 time-out, 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 time-out expires, only Key2 needs to be applied. The effect of this write operation is to re-initialize the WDG time-out 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 00000000h.

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 893. STCU2 SK Code Register (STCU_SKC)

Table 904. STCU_SKC field descriptions

Field	Description
0:31 SKC	STCU2 security key code for Off-line Test 0xD3FEA98B = Key1 to unlock the write access the STCU2 0x2C015674 = Key2 to unlock the write access the STCU2 STCU2 security key code for On-line Test 0x753F924E = Key1 to unlock the write access the STCU2 0x8AC06DB1 = Key2 to unlock the write access the STCU2

61.5.3 STCU2 Configuration Register (STCU_CFG)

The STCU_CFG register includes the global configuration of the STCU2 and can be updated both in the Off-line and On-line Test steps.

Access to this register is described in *Figure 894*. It also depends on the state of the WRP bit as follows:

- When WRP = 0, the register can be written during the On-line 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 Off-line Self-Test phase, you can always write to this field. In the On-line Self-Test phase, you can write to this field only when WRP = 0.

Figure 894. STCU2 Configuration Register (STCU_CFG)

Table 905. 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. 0h to (LBIST -1) – pointer to LBIST 10h to (10h + MBIST -1) – pointer to MBIST 7Fh – 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. 00h No delay 01h 001 * 16 STCU2 Core clock cycles 02h 002 * 16 STCU2 Core clock cycles 03h 003 * 16 STCU2 Core clock cycles ... FDh 253 * 16 STCU2 Core clock cycles FEh 254 * 16 STCU2 Core clock cycles FFh 255 * 16 STCU2 Core clock cycles
23 WRP	Write Protection. 0 Specific STCU2 registers can be written through IPS bus interface after Off-line Self-Test sequence has completed. 1 Specific STCU2 registers cannot be written though IPS even after Off-line 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 905. STCU_CFG field descriptions(Continued)

Field	Description
28 MBU	MBIST MBU Test Enabled. 0 MBIST Full algo or reduced one without PMOS test (PMOSEN set to 0) is applied. 1 MBIST simplified Multi Bit Upset algo is used to check the RAM. If a ROM is selected, the applied algo will be 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

61.5.4 STCU2 PLL Configuration Register (STCU_PLL_CFG)

The STCU_PLL_CFG register defines the parameters used to program the PLL only when the Off-line LBISTs or MBISTs are performed and STCU_RUN[MBPLLEN] = 1 or STCU_RUN[LBPLLEN] = 1.

In the On-line state, these bits are not effective. It is also possible to set the PLL during active On-line 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 Off-line 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 895. STCU2 PLL Configuration Register (STCU_PLL_CFG)

Table 906. 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 24: 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 24: 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 24: Dual PLL Digital Interface (PLLDIG)* to define the exact mapping between the PLLLDF value and the respective PLL's Loop Division factor.

61.5.5 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 Off-line Self-Test sequence is not run and the STCU_RUN[BYP] bit is not set, the bits 15...0 define the time-out before the STCU_ERR[WDTO] bit is set and the STCU2 core clock is switched off.

When the Off-line or On-line Self-Test sequence is run, it defines the time-out 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:

- Off-line Self-Test phase is active
- On-line 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 896. STCU2 Watchdog Register Granularity (STCU_WDG)

Table 907. 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 Off-line and On-line 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

61.5.6 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 46 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:

- Off-line Self-Test phase is active.
- On-line 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 W	CRCE															
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	CRCE															
Reset	The reset value is chip-specific; see the chapter that describes how modules are configured and connected.															

Figure 897. STCU2 CRC Expected Status Register (STCU_CRCE)

Table 908. STCU_CRCE field descriptions

Field	Description
0:31 CRCE	CRC expected signature

61.5.7 STCU2 CRC Read Status Register (STCU_CRCR)

The STCU_CRCR register reports the value obtained at the end of the Off-line and On-line 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 898. STCU2 CRC Read Status Register (STCU_CRCR)

Table 909. STCU_CRCR field descriptions

Field	Description
0:31 CRCR	Read CRC signature

61.5.8 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 On-line and Off-line 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 61.7.4: Design implementation information*.

Access to this register is described in *Figure 899* 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	0	0	0	0	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	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 899. STCU2 Error Register (STCU_ERR_STAT)

Table 910. STCU_ERR_STAT field descriptions

Field	Description
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	Off-line 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 Off-line Self-Test sequence execution. The On-line 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.
28 WDTO	Off-line Watchdog time-out. 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

Table 910. STCU_ERR_STAT field descriptions(Continued)

Field	Description
29 CRCS	Off-line CRC status. This flag is activated *only* when Off-line 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	Off-line 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	Off-line 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)

61.5.9 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:

- Off-line Self-Test phase is active.
- On-line 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 900. STCU2 Error FM Register (STCU_ERR_FM)

Table 911. 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 time-out 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

61.5.10 STCU2 Off-line LBIST Status Register (STCU_LBS)

The STCU_LBS register includes the results corresponding to the execution of the selected Off-line 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 901. STCU2 Off-line LBIST Status Register (STCU_LBS)

Table 912. STCU_LBS field descriptions

Field	Description
16:31 LBS[(NLBIST-1):0]	Off-line status of the selected LBIST 0 Failed LBIST execution 1 Successful LBIST execution

61.5.11 STCU2 Off-line LBIST End Flag Register (STCU_LBE)

The STCU_LBE register includes the End Flag respective to the execution of the selected Off-line 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 902. STCU2 Off-line LBIST End Flag Register (STCU_LBE)

Table 913. STCU_LBE field descriptions

Field	Description
16:31 LBE[(NLBIST-1):0]	Off-line End status of the selected LBIST 0 LBIST execution not yet completed 1 LBIST execution finished

61.5.12 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:

- Off-line Self-Test phase is active.
- On-line 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 903. STCU2 LBIST Unrecoverable FM Register (STCU_LBUFM)

Table 914. STCU_LBUFM field descriptions

Field	Description
16:31 LBUFM[(NLBIST-1):0]	LBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping

61.5.13 STCU2 Off-line MBIST Status Low Register (STCU_MBSL)

The STCU_MBSL register includes the results corresponding to the execution of the selected Off-line MBIST in the range NMCUT = 0..31. The size of the register depends on the number of RAMs/ROMs 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 904. STCU2 Off-line MBIST Status Low Register (STCU_MBSL)

Table 915. STCU_MBSL field descriptions

Field	Description
0:31 MBS[(NMCUT-1):0]	Off-line 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

61.5.14 STCU2 Off-line MBIST Status Medium Register (STCU_MBSM)

The STCU_MBSM register includes the results corresponding to the execution of the selected Off-line MBIST in the range NMCUT = 32...63. The size of the register depends on the number of RAMs or ROMs 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 905. STCU2 Off-line MBIST Status Medium Register (STCU_MBSM)

Table 916. STCU_MBSM field descriptions

Field	Description
0:31 MBS[(NMCUT-1):32]	Off-line 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

61.5.15 STCU2 Off-line MBIST Status High Register (STCU_MBSH)

The STCU_MBSH register includes the results corresponding to the execution of the selected Off-line MBIST in the range NMCUT = 64...95. The size of the register depends on the number of RAMs/ROMs 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 906. STCU2 Off-line MBIST Status High Register (STCU_MBSH)

Table 917. STCU_MBSH field descriptions

Field	Description
0:31 MBS[(NMCUT-1):64]	Off-line 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

61.5.16 STCU2 Off-line MBIST End Flag Low Register (STCU_MBEL)

The STCU_MBEL register includes the End Flag respective to the execution of the selected Off-line MBIST in the range NMCUT = 0..31. The size of the register depends on the number of RAMs/ROMs 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 907. STCU2 Off-line MBIST End Flag Low Register (STCU_MBEL)

Table 918. STCU_MBEL field descriptions

Field	Description
0:31 MBE[(NMCUT-1):0]	Off-line End status (0 to 31) of the selected MBIST. 0 MBIST execution still ongoing 1 MBIST execution finished

61.5.17 STCU2 Off-line MBIST End Flag Medium Register (STCU_MBEM)

The STCU_MBEM register includes the End Flag respective to the execution of the selected Off-line MBIST in the range NMCUT = 32..63. The size of the register depends on the number of RAMs/ROMs 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 908. STCU2 Off-line MBIST End Flag Medium Register (STCU_MBEM)

Table 919. STCU_MBEM field descriptions

Field	Description
0:31 MBE[(NMCUT-1):32]	Off-line End status (32 to 63) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished

61.5.18 STCU2 Off-line MBIST End Flag High Register (STCU_MBEH)

The STCU_MBEH register includes the End Flag respective to the execution of the selected Off-line MBIST in the range NMCUT = 64..95. The size of the register depends on the number of RAMs/ROMs 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 909. STCU2 Off-line MBIST End Flag High Register (STCU_MBEH)

Table 920. STCU_MBEH field descriptions

Field	Description
0:31 MBE[(NMCUT-1):64]	Off-line End status (64 to 95) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished

61.5.19 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 number of RAMs/ROMs 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:

- Off-line Self-Test phase is active.
- On-line 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 910. STCU2 MBIST Unrecoverable FM Low Register (STCU_MBUFML)

Table 921. STCU_MBUFML field descriptions

Field	Description
0:31 MBUFM[(NMCUT-1):0]	MBIST Unrecoverable Fault Mapping. 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping

61.5.20 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 number of RAMs/ROMs 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:

- Off-line Self-Test phase is active.
- On-line 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 911. STCU2 MBIST Unrecoverable FM Medium Register (STCU_MBUFMM)

Table 922. STCU_MBUFMM field descriptions

Field	Description
0:31 MBUFM[(NMCUT-1):32]	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping

61.5.21 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 number of RAMs/ROMs 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:

- Off-line Self-Test phase is active.
- On-line 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 912. STCU2 MBIST Unrecoverable FM High Register (STCU_MBUFMH)

Table 923. STCU_MBUFMH field descriptions

Field	Description
0:31 MBUFM[(NMCUT-1):64]	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping

61.5.22 STCU2 LBIST Control Register *n* (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:

- Off-line Self-Test phase is active.
- On-line Self-Test phase is active and STCU_CFG[WRP] = 0.

Offset: 0x0100+n*0x40 (n=0 to (NLBIST-1)) 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 913. STCU2 LBIST Control Register *n* (STCU_LB_CTRL*n*)

Table 924. 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. 0h to (LBIST -1) — pointer to LBIST 10h to (10h + MBIST -1b) — pointer to MBIST 7Fh — 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 924. 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

61.5.23 STCU2 LBIST PC Stop Register *n* (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:

- Off-line Self-Test phase is active.
- On-line Self-Test phase is active and STCU_CFG[WRP] = 0.

Offset: 0x0104+n*0x40 (n=0 to (NLBIST-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	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 914. STCU2 LBIST PC Stop Register *n* (STCU_LB_PCS*n*)

Table 925. STCU_LB_PCS*n* field descriptions

Field	Description
6:31 PCS	Pattern counter stop PCS defines the pattern counter stop value.

61.5.24 STCU2 LBIST PRPG Low Register *n* (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:

- Off-line Self-Test phase is active.
- On-line Self-Test phase is active and STCU_CFG[WRP] = 0.

Offset: 0x0108+n*0x40 (n=0 to (NLBIST-1)) 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 915. STCU2 LBIST PRPG Low Register *n* (STCU_LB_PRPGL*n*)

Table 926. 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.

61.5.25 STCU2 LBIST PRPG High Register *n* (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:

- Off-line Self-Test phase is active.
- On-line Self-Test phase is active and STCU_CFG[WRP] = 0.

Offset: 0x010C+n*0x40 (n=0 to (NLBIST-1)) Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	PRPG63	PRPG62	PRPG61	PRPG60	PRPG59	PRPG58	PRPG57	PRPG56	PRPG55	PRPG54	PRPG53	PRPG52	PRPG51	PRPG50	PRPG49	PRPG48
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	PRPG47	PRPG46	PRPG45	PRPG44	PRPG43	PRPG42	PRPG41	PRPG40	PRPG39	PRPG38	PRPG37	PRPG36	PRPG35	PRPG34	PRPG33	PRPG32
W																
Reset	The reset value is chip-specific; see the chapter that describes how modules are configured and connected.															

Figure 916. STCU2 LBIST PRPG High Register *n* (STCU_LB_PRPGH*n*)

Table 927. 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.

61.5.26 STCU2 Off-line LBIST MISR Expected Low Register *n* (STCU_LB_MISREL*n*)

The STCU_LB_MISREL register defines the LSB part (31..0) of the Expected MISR of the Off-line 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 Off-line Self-Test phase is still active.

Offset: 0x0110+n*0x40 (n=0 to (NLBIST-1)) Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	MISRE31	MISRE30	MISRE29	MISRE28	MISRE27	MISRE26	MISRE25	MISRE24	MISRE23	MISRE22	MISRE21	MISRE20	MISRE19	MISRE18	MISRE17	MISRE16
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	MISRE15	MISRE14	MISRE13	MISRE12	MISRE11	MISRE10	MISRE9	MISRE8	MISRE7	MISRE6	MISRE5	MISRE4	MISRE3	MISRE2	MISRE1	MISRE0
Reset	The reset value is chip-specific; see the chapter that describes how modules are configured and connected.															

Figure 917. STCU2 Off-line LBIST MISR Expected Low Register *n* (STCU_LB_MISREL*n*)

Table 928. STCU_LB_MISREL*n* field descriptions

Field	Description
0:31 MISRE[31:0]	Off-line MISR Expected low Bits. This field defines the low part (31..0) of the Expected MISR.

61.5.27 STCU2 Off-line LBIST MISR Expected High Register *n* (STCU_LB_MISREH*n*)

The STCU_LB_MISREH register defines the MSB part (63..32) of the Expected MISR of the Off-line 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 Off-line Self-Test phase is still active.

Offset: 0x0114+n*0x40 (n=0 to (NLBIST-1)) Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	MISRE63	MISRE62	MISRE61	MISRE60	MISRE59	MISRE58	MISRE57	MISRE56	MISRE55	MISRE54	MISRE53	MISRE52	MISRE51	MISRE50	MISRE49	MISRE48
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	MISRE47	MISRE46	MISRE45	MISRE44	MISRE43	MISRE42	MISRE41	MISRE40	MISRE39	MISRE38	MISRE37	MISRE36	MISRE35	MISRE34	MISRE33	MISRE32
Reset	The reset value is chip-specific; see the chapter that describes how modules are configured and connected.															

Figure 918. STCU2 LBIST MISR Expected High Register *n* (STCU_LB_MISREH*n*)

Table 929. STCU_LB_MISREH*n* field descriptions

Field	Description
0:31 MISRE[63:32]	Off-line MISR Expected high Bits This field defines the high part (63..32) of the Expected MISR.

61.5.28 STCU2 Off-line LBIST MISR Read Low Register *n* (STCU_LB_MISRRL*n*)

The STCU_LB_MISRRL register report the LSB part (31..0) of the MISR obtained at the end of the Off-line 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 (NLBIST-1)) 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 919. STCU2 Off-line LBIST MISR Read Low Register *n* (STCU_LB_MISRRL*n*)

Table 930. STCU_LB_MISRRL*n* field descriptions

Field	Description
0:31 MISRR[31:0]	Off-line MISR Read Low Bits. This field is equivalent to the Low Bits (31..0) of the MISR obtained at the end of the Off-line LBIST execution

61.5.29 STCU2 Off-line LBIST MISR Read High Register *n* (STCU_LB_MISRRH*n*)

The STCU_LB_MISRRH register report the MSB part (63..32) of the MISR obtained at the end of each Off-line 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 (NLBIST-1)) 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 920. STCU2 Off-line LBIST MISR Read High Register *n* (STCU_LB_MISRRH*n*)

Table 931. STCU_LB_MISRRH*n* field descriptions

Field	Description
0:31 MISRR[63:32]	Off-line MISR Read High Bits. This field is equivalent to the High Bits (63..32) of the MISR obtained at the end of the Off-line LBIST execution.

.

61.5.30 STCU2 MBIST Control Register *n* (STCU_MB_CTRL*n*)

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:

- Off-line Self-Test phase is active.
- On-line Self-Test phase is active and STCU_CFG[WRP] = 0.

Offset: 0x0600+n*0x4 (n=0 to (NMCUT-1)) 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 921. STCU2 MBIST Control Register *n* (STCU_MB_CTRL*n*)

Table 932. STCU_MB_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 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. 0h to (LBIST -1) — pointer to LBIST 10h to (10h + MBIST -1b) — pointer to MBIST 7Fh — pointer to NIL. No BIST execution. others — invalid pointer, an error is set in the STCU_ERR register.

61.6 Register map

The following STCU2 register map shows a full implementation case. If fewer LBISTs or MBISTs are implemented, note of 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 number of RAMs or ROMs subject to BIST.
- The size of STCU_MBSM, STCU_MBSMSW, STCU_MBEM, STCU_MBEMSW, STCU_MBUFMM depends on the number of RAMs or ROMs 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 number of RAMs or ROMs 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 number of RAMs or ROMs subject to BIST.

Table 933. 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
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	PLLLDV2	PLLLDV1	PLLLDV0
0x0014	STCU_WDG	WDGEOC																															
0x0018	Reserved	Reserved																															
0x001C	STCU_CRCE	CRCE																															
0x0020	STCU_CRCR	CRCR																															
0x0024	STCU_ERR_STAT	Reserved						0	Reserved	Reserved				0	0	CRCSSW	0	0	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
0x003C	Reserved	Reserved																															
0x0040	STCU_LBUFM	Reserved																LBUFM15	LBUFM14	LBUFM13	LBUFM12	LBUFM11	LBUFM10	LBUFM9	LBUFM8	LBUFM7	LBUFM6	LBUFM5	LBUFM4	LBUFM3	LBUFM2	LBUFM1	LBUFM0

Table 933. 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
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
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)																																	
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
Address space not used (available for future expansion)																																	
0x0600 + n*0x4 n=0 to (NMCUT-1)	STCU_MB_CTRL*n*	CSM	PTR							Reserved																							

61.7 Use cases and limitations

The STCU2 module is designed to give high flexibility during the On-line and Off-line Self-Test. This section shows the programming operations that have to be performed to correctly use this module in three typical use cases:

- Off-line Self-Test sequence after a reset trigger event.
- Bypass mode of the Self-Test sequence after a reset trigger event.
- On-line Self-Test sequence.

61.7.1 Off-line 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 RAMs or ROMs according to the System specifications. The sequence is the following:

- Unlock the Off-line 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 Off-line 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 Off-line 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 Off-line 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 Off-line Self-Test execution has successfully completed are an additional safety layer added to improve the already implemented Auto Self-Test feature included in STCU2.

61.7.2 On-line 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 On-line 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 On-line 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 On-line Self-Test registers.
- The LBIST, MBIST and STCU2 internal fault reaction should already be programmed during the Off-line 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 Off-line 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 On-line Self-Test sequence in the STCU_CRCE register.
- Program the STCU_RUNSW[RUNSW] bit to enable the On-line Self-Test execution and the L/MBSWPLLEN bits to enable the PLL lock signal monitor during LBIST or MBIST execution.
- Start the On-line 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.
- If at least one of the selected LBISTs enables the Global functional reset, the STCU2 requests a functional reset which restarts the system. If all the selected LBISTs enable

their own dedicated functional reset, only these local reset lines are applied and only the respective LBIST partitions are cleaned up.

- 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 RAMs/ROMs 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 RAMs/ROMs 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 On-line 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.

61.7.3 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 Off-line 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.

61.7.4 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/Off-line 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 On-line Self-Test, the STCU2 may not take control of the PLL, but only monitor the lock signal to flag incorrect unlock events.

62 Register Protection (REG_PROT)

62.1 Introduction

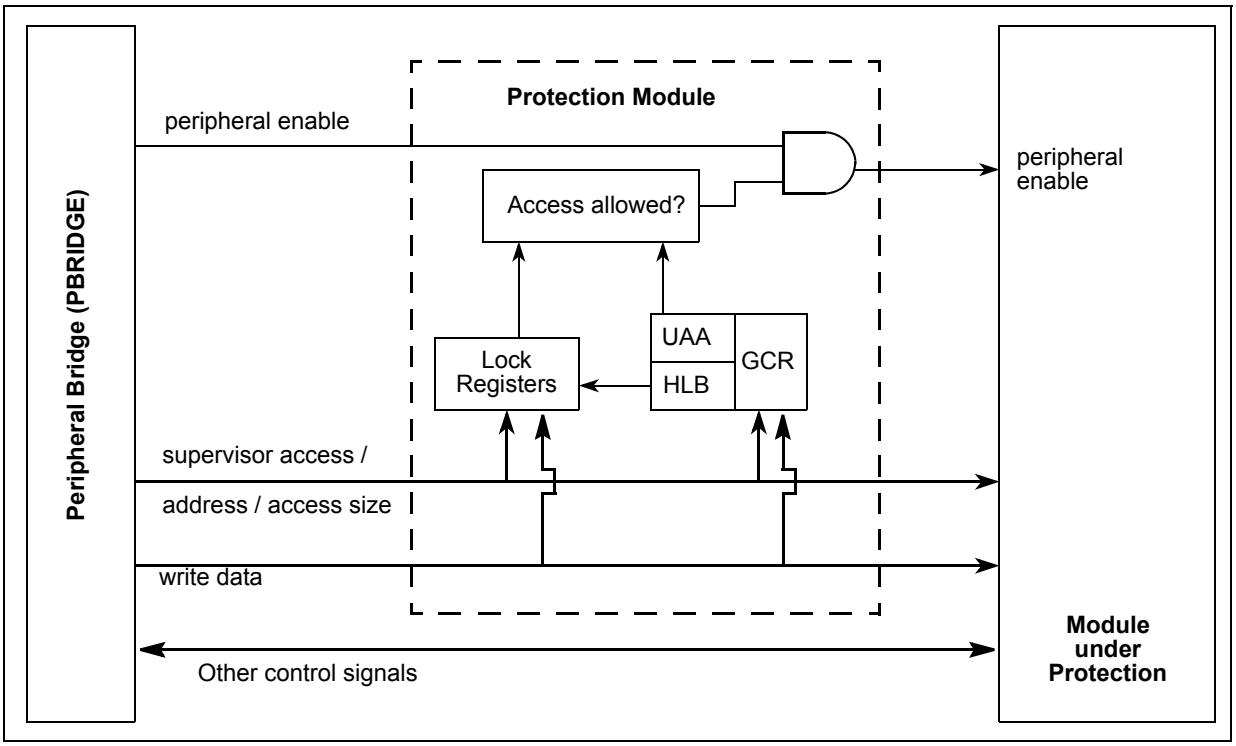
62.1.1 Overview

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 922*.

Figure 922. REG_PROT Block Diagram



62.1.2 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.

62.1.3 Modes of operation

The Register Protection module is operable when the module under protection is operable.

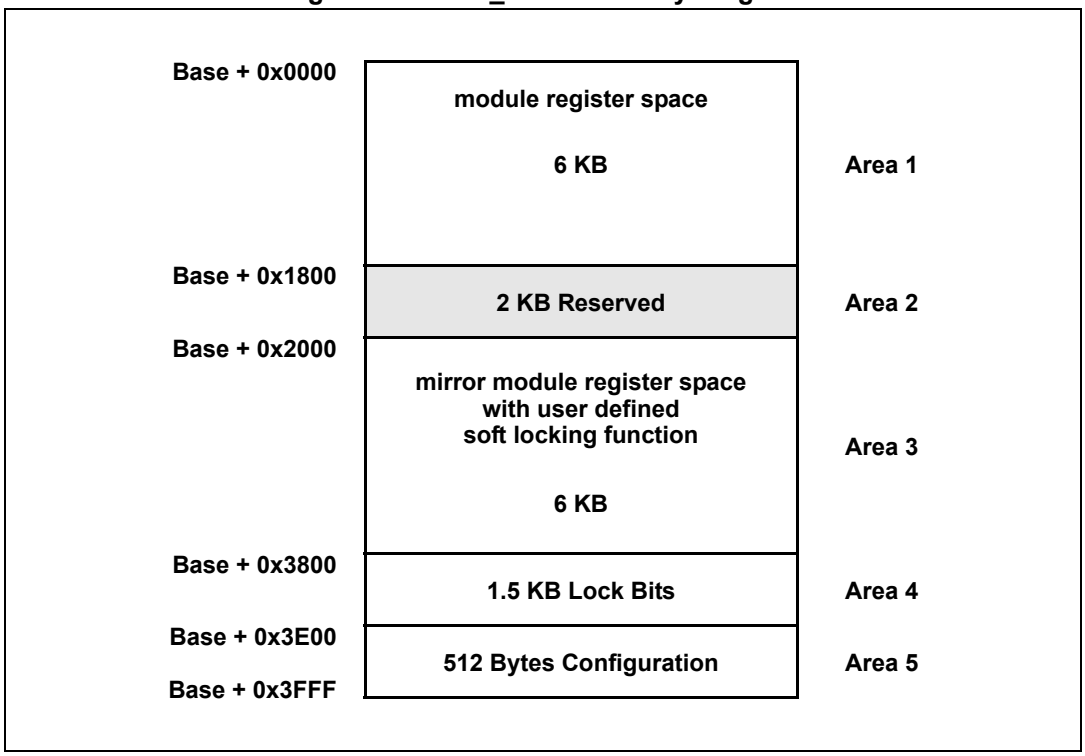
62.2 External signal description

There are no external signals.

62.3 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 923*.

Figure 923. 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.

62.3.1 Memory map

Table 934 shows the memory map for a module which is protected via a REG_PROT module.

Table 934. REG_PROT Memory Map

Offset	Use	Location
0x0000	Module Register 0 (MR0)	*Section 62.3.2.1*
0x0001	Module Register 1 (MR1)	*Section 62.3.2.1*
0x0002	Module Register 2 (MR2)	*Section 62.3.2.1*
0x0003 - 0x17FF	Module Register 3 (MR3) - Module Register 6143(MR6143)	*Section 62.3.2.1*
0x1800 - 0x1FFF	Reserved	
0x2000	Module Register 0 (MR0) + Set Soft Lock Bit 0 (LMR0)	*Section 62.3.2.2*
0x2001	Module Register 1 (MR1) + Set Soft Lock Bit 1 (LMR1)	*Section 62.3.2.2*
0x2002 - 0x37FF	Module Register 2 (MR2) + Set Soft Lock Bit 2 (LMR2) - Module Register 6143 (MR6143) + Set Soft Lock Bit 6143 (LMR6143)	*Section 62.3.2.2*
0x3800	Soft Lock Bit Register 0 (SLBR0): Soft Lock Bits 0-3	*Section 62.3.2.3*
0x3801	Soft Lock Bit Register 1 (SLBR1): Soft Lock Bits 4-7	*Section 62.3.2.3*
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 62.3.2.3*
0x3E00 - 0x3FFB	Reserved	
0x3FFC	Global Configuration Register (GCR)	*Section 62.3.2.4*

Note: Reserved registers in area #1 will be handled as specified in the protected module's documentation.

62.3.2 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.

62.3.2.1 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.

62.3.2.2 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 936*.

62.3.2.3 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 935. 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 924. Soft Lock Bit Register (SLBR*n*)

Table 936. 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 writeable
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 writeable
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 writeable
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 writeable

62.3.2.4 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 925. Global Configuration Register (GCR)

Table 937. 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.

62.4 Functional description

The following sections describe the functional characteristics of the Register Protection module.

62.4.1 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.

62.4.2 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.

62.4.2.1 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 926 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 926. Change lock settings directly via Area #4

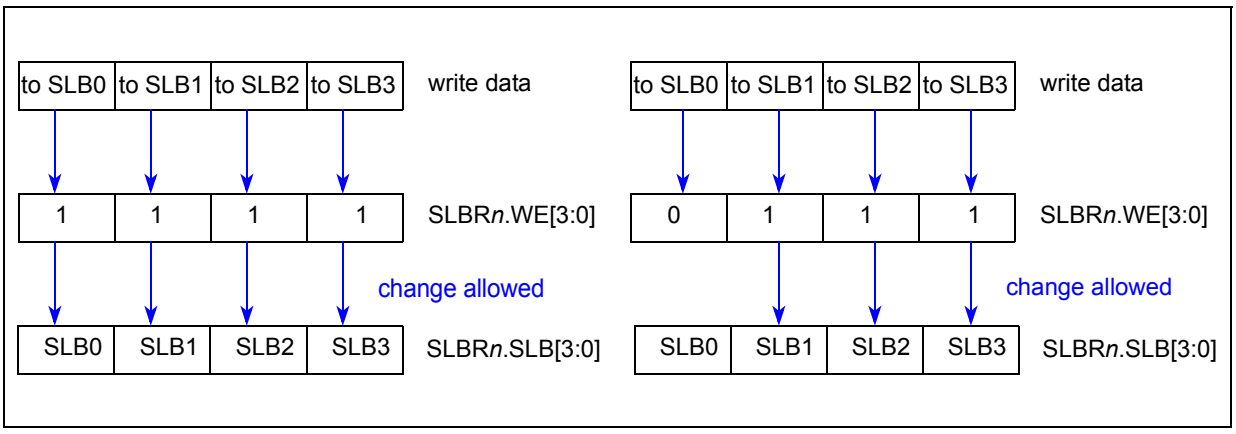
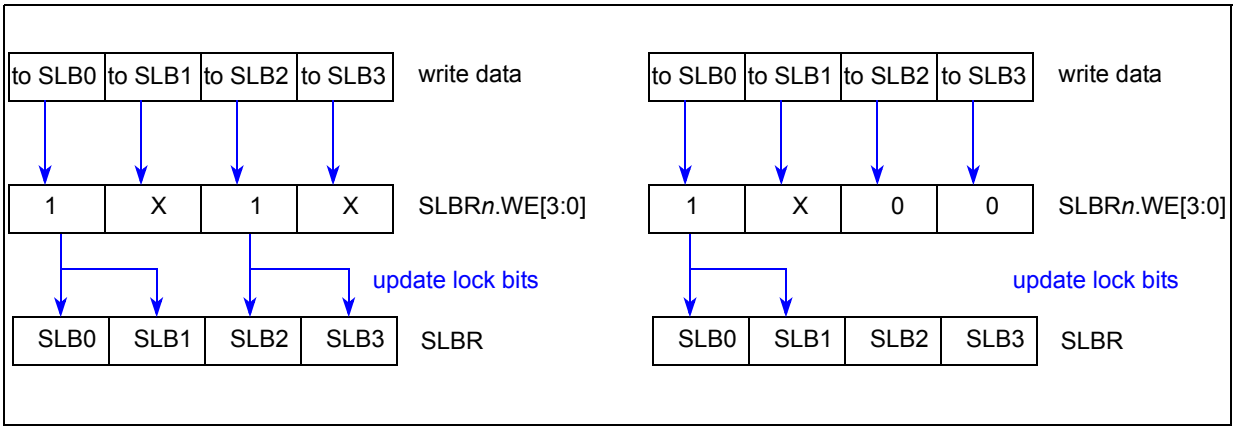


Figure 926 shows four registers that can be protected 8-bit wise. In *Figure 927* registers with 16-bit protection and in *Figure 928* registers with 32-bit protection are shown:

Figure 927. Change lock settings for 16-bit protected addresses

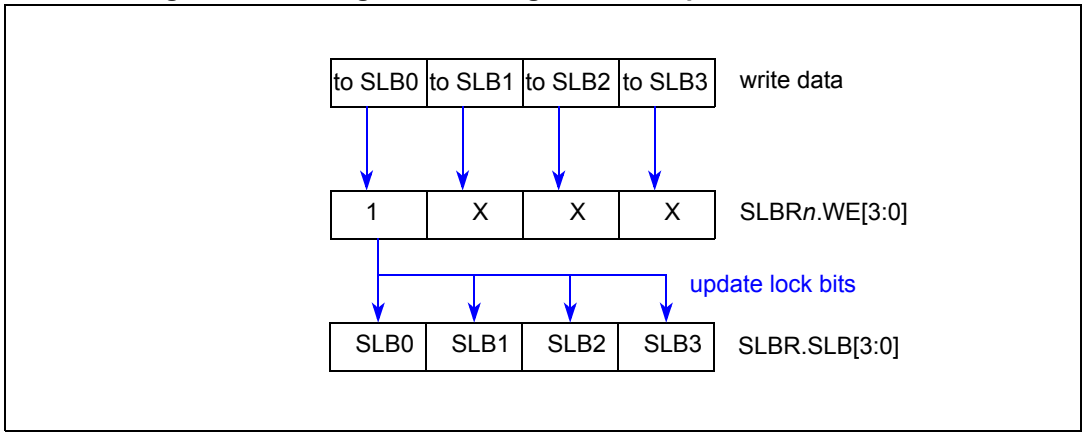


On the right side of *Figure 927* it is shown that the data written to SLBR*n*.SLB[0] is automatically written to SLBR*n*.SLB[1] also. This is done as the address reflected by SLBR*n*.SLB[0] is protected 16-bit wise. Note that in this case the write enable SLBR*n*.WE[0] must be set while SLBR*n*.WE[1] 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 927* 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.

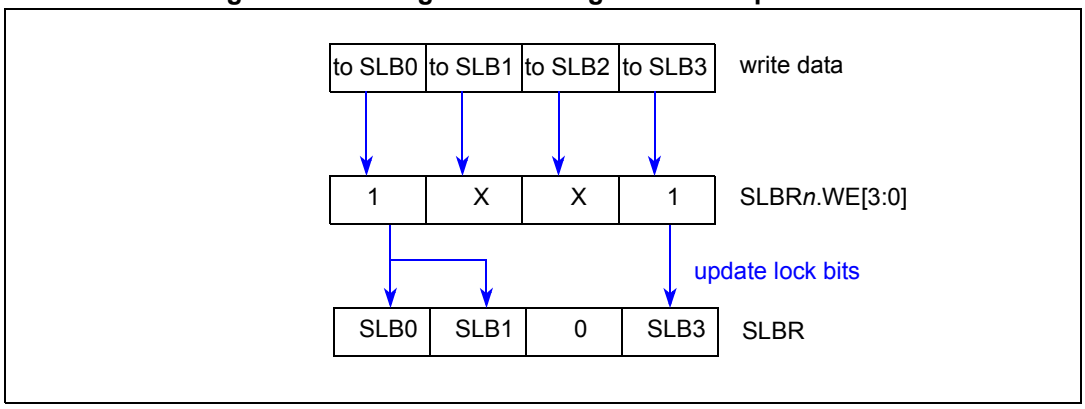
In *Figure 928* a 32-bit wise protected register is shown. When SLBR*n*.WE[0] is set the data written to SLBR*n*.SLB[0] is automatically written to SLBR*n*.SLB[3:1] also. Otherwise SLBR*n*.SLB[3:0] remains unchanged.

Figure 928. Change lock settings for 32-bit protected addresses



In *Figure 929* an example is shown which has a mixed protection size configuration:

Figure 929. Change lock settings for mixed protection

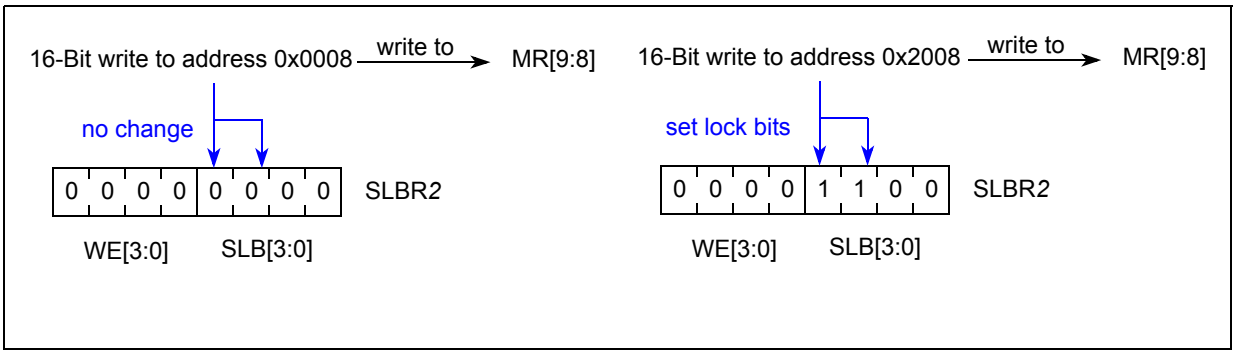


The data written to SLBR*n*.SLB[0] is mirrored to SLBR*n*.SLB[1] as the corresponding register is 16-bit protected. The data written to SLBR*n*.SLB[2] is blocked as the corresponding register is unprotected. The data written to SLBR*n*.SLB[3] is written to SLBR*n*.SLB[3].

62.4.2.2 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 930* shows one example:

Figure 930. Enable Locking Via Mirror Module Space (Area #3)

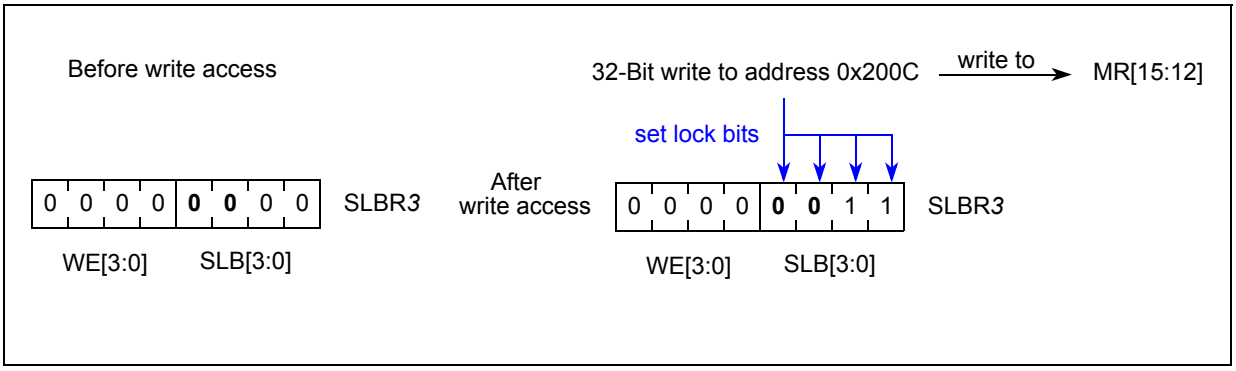


When writing to address 0x0008 the registers MR9 and MR8 in the protected module are updated. The corresponding lock bits remain unchanged (left part of *Figure 927*).

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 927*).

Figure 931 shows an example where some addresses are protected and some are not:

Figure 931. Enable locking for protected and unprotected addresses



In the example in *Figure 931* 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.

62.4.2.3 Write protection for locking bits

Changing the locking bits through any of the procedures mentioned in *Section 62.4.2.1: Change lock settings directly via Area #4* and *Section 62.4.2.2: Enable locking via mirror module space (Area #3)* 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.

62.4.3 Access errors

The protection module generates transfer errors under several circumstances. For the area definition refer to *Figure 923*

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

62.5 Initialization/application information

62.5.1 Reset

The reset state of each individual bit is shown within the Register Description section (See *Section 62.3.2*). In summary, after reset, locking for all MR*n* registers is disabled. The registers can be accessed in Supervisor Mode only.

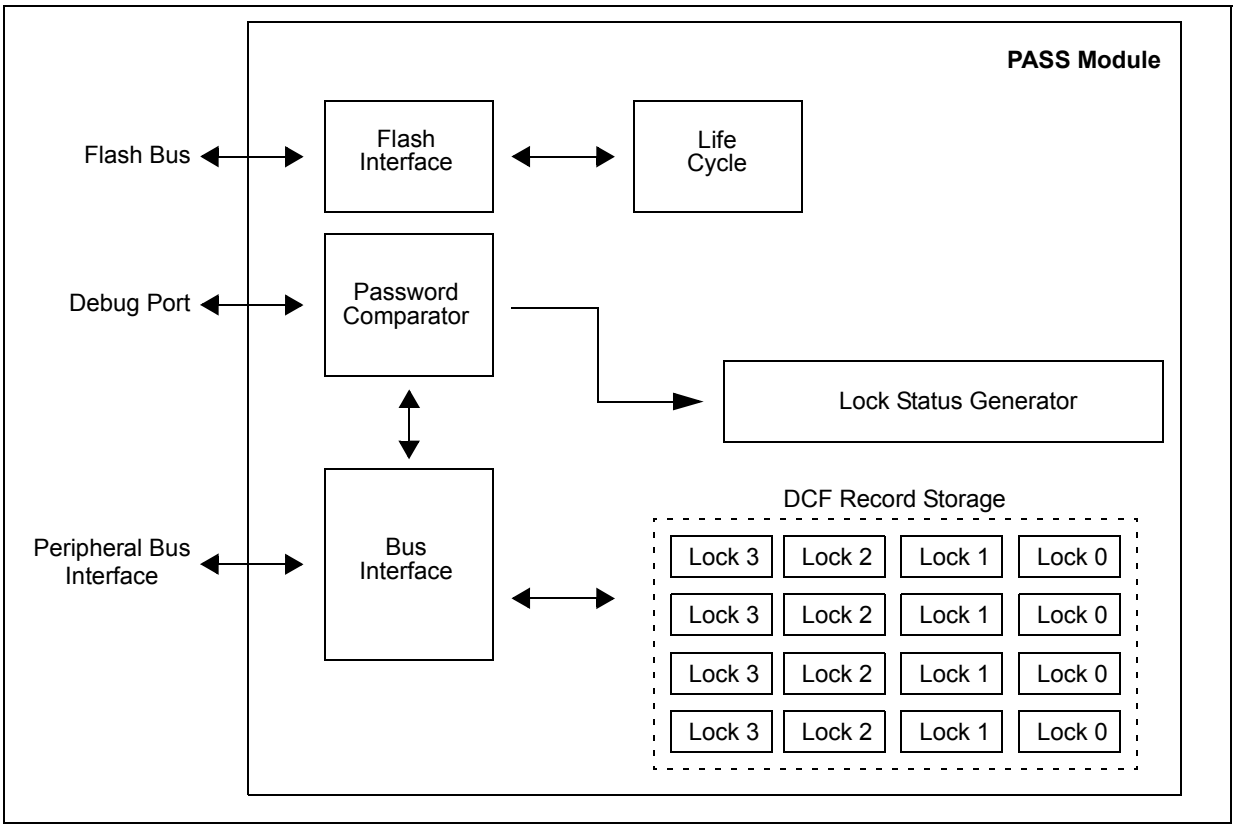
63 Password and Device Security Module (PASS)

63.1 Introduction

63.1.1 Overview

The Password and device security (PASS) module shown in *Figure 932*, receives password challenges and determines their validity. It also maintains the device security and access states.

Figure 932. PASS module block diagram



63.1.2 Features

The PASS includes these distinctive features:

- Lifecycle status
- Password comparison
- JTAG password comparison
- Retest Password Comparison

63.1.3 Modes of operation

The PASS operates identically in all system modes.

63.2 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 63.3: DCF clients*.)

Table 938 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 938. PASS memory map

Offset	Register	Access	Reset Value	DCF Client Address(1) (ADDR[14:0])	Section
0x0000	Lifecycle Status Register (LCSTAT)	R	0x000000XX(2)	None	*Section 63.2.1.1*
0x0008	Challenge Selector Register (CHSEL)	R/W	0x00000000	None	*Section 63.2.1.2*
0x0010	Challenge Status Register (CSTAT)	R	0x00000000	None	*Section 63.2.1.3*
0x0014-0x001F	Reserved				
0x0020	Challenge Input Register 0 (CIN0)	W	0x00000000	None	*Section 63.2.1.4*
0x0024	Challenge Input Register 1 (CIN1)	W	0x00000000	None	*Section 63.2.1.4*
0x0028	Challenge Input Register 2 (CIN2)	W	0x00000000	None	*Section 63.2.1.4*
0x002C	Challenge Input Register 3 (CIN3)	W	0x00000000	None	*Section 63.2.1.4*
0x0030	Challenge Input Register 4 (CIN4)	W	0x00000000	None	*Section 63.2.1.4*
0x0034	Challenge Input Register 5 (CIN5)	W	0x00000000	None	*Section 63.2.1.4*
0x0038	Challenge Input Register 6 (CIN6)	W	0x00000000	None	*Section 63.2.1.4*
0x003C	Challenge Input Register 7 (CIN7)	W	0x00000000	None	*Section 63.2.1.4*
0x0040-0x00AF	Reserved				
None	Censorship	None	0x00000000	0x00B0	*Section 63.3.1*
0x00C4-0x00DF	Reserved				
0x00E0-0x00FF	Reserved				
Password Group 0					
0x0100	Lock 0 Status Register (LOCK0_PG0)	R/W	0xXXXXXXXX(3)	0x0100	*Section 63.2.1.5*
0x0104	Lock 1 Status Register (LOCK1_PG0)	R/W	0x0000XXXX(3)	0x0104	*Section 63.2.1.6*
0x0108	Lock 2 Status Register (LOCK2_PG0)	R/W	0xXXXXXXXX(3)	0x0108	*Section 63.2.1.7*
0x010C	Lock 3 Status Register (LOCK3_PG0)	R/W	0xXXXXXXXX(3)	0x010C	*Section 63.2.1.8*

Table 938. 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[3]	0x0110	*Section 63.2.1.5*
0x0114	Lock 1 Status Register (LOCK1_PG1)	R/W	0x0000XXXX[3]	0x0114	*Section 63.2.1.6*
0x0118	Lock 2 Status Register (LOCK2_PG1)	R/W	0xXXXXXXXX[3]	0x0118	*Section 63.2.1.7*
0x011C	Lock 3 Status Register (LOCK3_PG1)	R/W	0xX0XXXXXX[3]	0x011C	*Section 63.2.1.8*
Password Group 2					
0x0120	Lock 0 Status Register (LOCK0_PG2)	R/W	0xXXXXXXXX[3]	0x0120	*Section 63.2.1.5*
0x0124	Lock 1 Status Register (LOCK1_PG2)	R/W	0x0000XXXX[3]	0x0124	*Section 63.2.1.6*
0x0128	Lock 2 Status Register (LOCK2_PG2)	R/W	0xXXXXXXXX[3]	0x0128	*Section 63.2.1.7*
0x012C	Lock 3 Status Register (LOCK3_PG2)	R/W	0xX0XXXXXX[3]	0x012C	*Section 63.2.1.8*
Password Group 3					
0x0130	Lock 0 Status Register (LOCK0_PG3)	R/W	0xXXXXXXXX[3]	0x0130	*Section 63.2.1.5*
0x0134	Lock 1 Status Register (LOCK1_PG3)	R/W	0x0000XXXX[3]	0x0134	*Section 63.2.1.6*
0x0138	Lock 2 Status Register (LOCK2_PG3)	R/W	0xXXXXXXXX[3]	0x0138	*Section 63.2.1.7*
0x013C	Lock 3 Status Register (LOCK3_PG3)	R/W	0xX0XXXXXX[3]	0x013C	*Section 63.2.1.8*

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 lifecycle status
3. Reset value depends on password group lock state

Note: Non-implemented registers generate bus aborts if enabled by the SOC.

63.2.1 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’.

63.2.1.1 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 933. Lifecycle Status register (LCSTAT)

PASS_LCSTAT register functions are as shown in *Table 939*.

Table 939. 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 63.5.3: 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 111 In Field

63.2.1.2 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 934. CHSEL register

CHSEL register functions are as shown in *Table 940*.

Table 940. 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

63.2.1.3 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 935. Challenge Status register (CSTAT)

CSTAT register functions are as shown in *Table 941*.

Table 941. 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.

63.2.1.4 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.

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[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	PW32[16:31]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 936. Challenge Input register (CIN*n*)

CIN register functions are as shown in *Table 942*.

Table 942. 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.

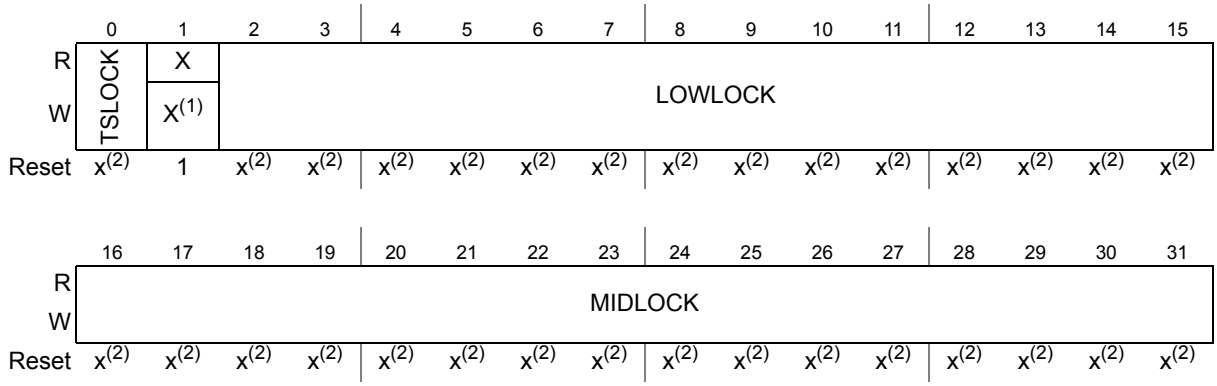
All accesses to these registers need to be made by the bus master which wrote the CHSEL register previously. If another master writes any of the CIN registers before the original master has written CIN7, the comparison fails.

63.2.1.5 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. This bit is not implemented and its value has no impact on the system.
2. If no DCF records have been written to change the lock state, the bit will reset to '1'.

Figure 937. Password Group *n* - Lock 0 Status register (LOCK0_PG*n*)

LOCK0_PG*n* register functions are as shown in *Table 943*.

Table 943. 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.

63.2.1.6 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 938. Password Group *n* - Lock 1 status register (LOCK1_PG*n*)

LOCK1_PG*n* register functions are as shown in *Table 944*.

Table 944. 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.

63.2.1.7 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

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	L_128LCK[0:15]															
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	L_128LCK[16:31]															
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 resets to ‘1’.

Figure 939. 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 945*.

Table 945. LOCK2_PG*n* register field descriptions

Field	Description
0:31 L_128LCK	Lower 128KByte Block Lock 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.

63.2.1.8 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	U_256LCK(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 940. Password Group *n* - Lock 3 status register (LOCK3_PG*n*)

LOCK3_PG*n* register functions are as shown in *Table 946*.

Table 946. 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 946. 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 U_256LCK	Upper 256 KB 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.

For operation of the region read protect feature, refer to *Section 63.4.3: 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. .

63.3 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.

63.3.1 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 938*).

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 941. Censorship register (not accessible)

Table 947. 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

63.4 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.

63.4.1 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 63.3.1: Censorship*).

63.4.2 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 *63.2.1.8: Password Group n - Lock 3 status register*

(LOCK3_PGn)). Both 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 948. 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.

63.4.2.1 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.

63.4.3 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 949: 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 7.11.1: Password and Device Security Module (PASS) configuration*).

Table 949. 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

63.4.4 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.

63.4.5 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

63.4.5.1 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.

63.4.5.2 Retest Password Storage

The Retest password is 256 bits wide and is stored in Test Flash

63.4.5.3 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.

63.4.5.4 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.

63.5 Initialization/application information

63.5.1 Reset

The reset state of each individual bit is shown within the Register Description section (see *Section 63.2.1*). However, many bits receive their values based on DCF records stored in the UTEST Flash memory block.

63.5.2 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 950. 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 950. 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 951*.

Table 951. 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.

63.5.3 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 952 lists acronyms and abbreviations used in this document.

Table 952. Acronyms and abbreviations

Term	Meaning
AHB	Advanced high-performance bus
AIC	AIPS (PBRIDGE) integrity checker
AIPS	AHB to IPS bridge (PBRIDGE)
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
BIST	Built-in self test
BIU	Bus interface unit
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
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
ECSM	Error correction status module
EDC	Error detection code
EIM	Error injection module
EMI	Electromagnetic interference
EOUT	Error out

Table 952. Acronyms and abbreviations(Continued)

Term	Meaning
ESD	Electromagnetic static discharge
ETC	eTimer Channel
FCCU	Fault collection and control unit
FMPLL	Frequency-modulated phase-locked loop
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
ME	Mode entry module (MC_ME)
MEMU	Memory error management unit
MFD	Multiplication factor divider
Mux	Multiplex

Table 952. Acronyms and abbreviations(Continued)

Term	Meaning
MVD	Minimum Voltage Detector
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
PMC	Power management controller
POR	Power On Reset
PRAM	Platform RAM controller
PWD	Password
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
SAR ADC	Successive approximation register analog-to-digital converter
SIL	Safety integrity level
SIUL	System integration unit lite
SMPU	System memory protection unit

Table 952. Acronyms and abbreviations(Continued)

Term	Meaning
SoC	System-on-chip
SOR	Sphere of redundancy
SPI	Serial peripheral interface
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
UVD	Upper Voltage Detector
VCO	Voltage controlled oscillator
VREG	Voltage regulator
WDG	Watchdog timer
WKPU	Wakeup unit
WS	Wait state
XBAR	Crossbar
XBIC	XBAR integrity checker

Revision history

Table 953. Document revision history

Date	Rev.	Changes
16-Apr-2013	1	Initial release
23-Sept-2013	2	Updated the Disclaimer
29-May-2014	3	Following are the updates in this version of the RM: **Introduction** *Figure 2: Block diagram*: Replaced AXBS by XBAR Inserted JDC block, deleted DSPI_3 block *Section 1.2: Features*: In first bullet, deleted “Single-precision floating point operations” In bullet STCU2, added “Support for Logic BIST and Memory BIST at power on” In bullet DSPI, decreased value from ‘4’ to ‘3’. *Table 2: SPC570S50Lx device feature summary (Family Superset Configuration)*: decreased value for DSPI feature from ‘4’ to ‘3’. **Embedded Memories** *Figure 1: SPC574S60Lx, SPC574S60Q6 flash memory block diagram* minicache picture doubled Minicache changed from 4-entry, 2-way to 4-entry, 4-way Updated *Section 1.3.2* concerning (OTP size, Flash segmentation and organization) and *Figure 3* **Signal Description** *Table 2: Pin startup and reset states*: Corrected startup states for Analog Input pins, PORST, TDI, TDO, TMS, and TCK. *Table 8*, *Table 9* and *Table 10*: Removed all instances of DSPI3. *Table 8: Pin multiplexing description*: Added column Strength. Inserted row ESR0 for PC[15]. For Signal column is “SOUT”, replaced in Full signal name column “MOSI (Master/slave)” and “MISO (Master/slave)” by “Serial Data”. Replaced “INP” by “NMI”. Updated clkout feature to CLOCKOUT. *Table 9: Peripheral multiplexing description*: Added eTimer Instances (eTimer_0 to eTimer_3) for FAULT_0 and FAULT_1 Functions (MSCR[712] to MSCR[719]). Replaced “INP” by “NMI”. *Table 8: eTQFP64 and eTQFP100 pinout*: Replaced “MOSI” and “MISO” instances by “Serial Data”. Updated Nexus REQ to Nexus RDY.

Table 953. Document revision history(Continued)

Date	Rev.	Changes
29-May-2014	3	**Memory Map** *Table 2: Flash memory map*: Corrected end address of System RAM (48KB size) *Table 3: Peripheral bridge 1 memory map*: Deleted DSPI 3 and changed addresses description to 'Reserved' Changed reserved start address from 0xFBC0_8000 to 0xFBC2_8000 Added "On-Platform Peripheral slot" and "Off-Platform Peripheral slot" columns Added addresses : 0xF800_4000 - 0xF807_FFFF : RESERVED 0xF808_0000 - 0xF808_3FFF : RESERVED 0xF808_4000 - 0xFBC1_FFFF : RESERVED *Table 4: Peripheral bridge 0 memory map*: Added ADC seq and DC supervisor seq with corresponding addresses Added row for JDC and set corresponding addresses Rename RCOSC dig, XOSC dig and PLL dig descriptions (was RCOSC, XOSC and PLL 0) Added row for JDC and set corresponding addresses Added PASS at start address 0xFFFE4000 (was reserved) Added BAR at start address 0xFFFF_C000 (was reserved) Change start address of latest Reserved to 0xFFFF_E000 Changed descriptions of start address 0xFFFF_C000 to Reserved (was BAR) Changed descriptions of start address 0xFFE3_8000 to Reserved (was ADC supervisor seq) Changed descriptions of start address 0xFFE3_C000 to Reserved (was ADC supervisor) Changed reserved stop address 0xFFF4_BFFF to 0xFFF3_BFFF Added "On-Platform Peripheral slot" and "Off-Platform Peripheral slot" columns Added addresses : 0xF800_4000 - 0xF807_FFFF : RESERVED 0xF808_0000 - 0xF808_3FFF : RESERVED 0xF808_4000 - 0xFBC1_FFFF : RESERVED **Functional Safety** *Section 1.5.2.2: CPU to Flash read path protection*: Rewritten the content of sentences: "The data is supposed to have ... trigger the FCCU." *Table 6: FCCU failure inputs*: Add rows for FCCU inputs 42 (SSCM,...), 44 (SMPU, mpu_mon_err,...) and 45 (RAM controller,...) Add input trigger parameter name "pram_fccu_ram_alarm" in FCCU input 43 (RAM controller) Added CMU_3 to slot 54 Updated slot 55 Added right column "Error reaction path check". Added rows for FCCU inputs: 61 (RCCU: RCCU_DMA_MUX), 62 (RCCU: RCCU_EIM) and 55 (SSCM: SSCM_ACTIVE)

Table 953. Document revision history(Continued)

Date	Rev.	Changes
29-May-2014	3	Removed figures “AHB to IPS byte read mapping” and “EDC implementation” *Section 1.3*: Added reference “in the SPC570S50Lx Safety Manual (AN4247)” Added “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.” Replaced "process safety time" by "Fault Tolerant Time Interval", and “PST” by “FTTI” Changed “MBIST is ...” by “L/MBIST are...”, footnote is removed *Section 1.5.1*: Added text “(including DMA Mux)” to DMA bullet *Section 1.5.2*: Added *Table 5: ECC Engine Summary* Added text “Whenever the checkbits ... AIPS (strategy 2)” Removed text “and pads .. Flash memory.” Removed text “It is acceptable .. to the CPU.” Removed text “Because of ... diagram shown above. *Section 1.5.3*: Removed text “HW comparison... required” and “without CCFs... measurement loop *Section 1.5.4*: Removed text “loss of clock”, “In general, no special ... between the two channels.” and “unless non-existence... can be shown *Section 1.5.5*: Changed the whole text Added cross-references to chapter Power management and chapter Power Control Unit (MC_PCU) *Section 1.5.6*: Removed “if voltage regulation ... than its supervision.” Replaced ERRORn by F0 and F1 Removed text “It is possible... reactions and indications of them” Removed text “A configuration... to a peripheral RAM.” Removed text “The configuration data... information is necessary.” Removed text “(above and beyond the ECC error signal) *Section 1.5.7*: deleted “(that is F0... for failure)”, “(except Power on)” Changed “for safe checking” by “to verify the integrity of the reaction path” *Section 1.5.8*: Changed first paragraph to “SPC570Sx supports some mechanisms to detect fault which can became latent. Two important mechanisms dedicate to this functionality are LBIST and MBIST.” Add “On cut 1.0” to text note Removed text “Random desabling... following approaches:” *Section 1.5.9*: Removed “Possibly required measures might included:” and the 3 following bullets Removed “Power supply analysis shall... EMI capacitance for PPLs).” *Section 1.6.1*: Removed paragraph text “Requirements for... module section” Removed all text with “(rationale:...” Removed paragraphs text “If required, EDC logic... unprotected signal lines.” Added cross-reference to MEMU chapter Changed text “RAMs” by ‘RAM” Added text “such as, address & control feedback” to “(and elements of the data path...)”

Table 953. Document revision history(Continued)

Date	Rev.	Changes
29-May-2014	3	*Section 1.6.2*: Added (Multiple bit upset) to MBU, (Single event latchup) to SEL and (Deep n-well) to DNW *Section 1.6.3*: Removed text “The number needs... by design.” Removed paragraph text “Signals can be compressed to... between the cores.” Removed paragraph text “If a compression scheme is... be the following” and the following 5 bullets *Section 1.6.5*: Replaced “destructive one” by “destructive escalation” *Section 1.6.7*: Removed “Otherwise the coverage... reduced accordingly.” Removed sentence “Coverage of the LBIST with respect to failure modes other than stuck-at shall be specified.” *Section 1.6.8*: Replaced “MPU” by “SMPU” *Section 1.6.9*: Removed text “To fulfill CCF... inclusive/exclusive ways.” Added Triple Voted Flip Flop in brackets to TV-FF Removed paragraphs text “The area in which no routing... the lake and the other replicated IP.” Removed text “The following is thus just an implementation hint.” Removed paragraph text “The following components... that is necessary:” and the following bullet Removed sentence “Layout and synthesis must... their effectiveness.” *Section 1.6.10*: Updated text to “1 temperature sensor...” *Section 1.7.1*: Removed sentence “It control signals are... at least two bits.” *Section 1.7.6*: Removed paragraph “SMPU supervision can be... violation is found:” and the following 4 bullets Added *Section 1.4: Module categorization* included *Table 3: PMHF module categories* and *Table 4: Module classification* Updated *Figure 1: Replication architecture* **Device Configuration** Added *Section 7.7.6.2: LINFlexD – implemented registers* Added full form of IVPR, and IVOR to *Figure 17: Interrupt scheme* Removed INTC_MPROT register from *Table 32: INTC implemented registers* *Table 33: Interrupt sources*: Removed the reset column Updated the source name column of IRQ# 477 Updated the source name column of IRQ# 674 Updated *Table 44: Register reset values* Updated *Table 47: eTimer instantiation*

Table 953. Document revision history(Continued)

Date	Rev.	Changes
29-May-2014	3	Added *Table 48: eTimer registers memory map* Updated *Table 67: Protected SIUL2 registers*. Added CMU_3 register to *Table 68: Protected CMU registers* Removed ME_PCTL226 from *Table 73: Protected MC_ME registers* Added registers : CGM_SC_DIV_RC, CGM_DIV_UPD_TYPE, CGM_DIV_UPD_TRIG, and CGM_SC_DC2 to *Table 74: Protected MC_CGM registers* Removed CGM_AC1_DC1 from *Table 74: Protected MC_CGM registers* Added registers : RGM_DBRE, and RGM_FBRE to *Table 75: Protected MC_RGM registers* Removed STCU2 configuration section Updated *Table 20: Reset settings for e200z0h resources* System reset setting for Resource “PVR” changed from “—” to “0x000” in *Table 20: Reset settings for e200z0h resources* Added *Table 24: Master ID number assignment* in *Section 7.3.2.1.1: Crossbar switch master assignments* Updated *Table 65: Reset values of selected FCCU registers* *Table 65: Reset values of selected FCCU registers*: Changed the name of registers from FCCU_RF_CFG[0:3] to FCCU_NCF_CFG[0:3], FCCU_RFS_CFG[0:7] to FCCU_NCFS_CFG[0:7], FCCU_RF_S[0:3] to FCCU_NCF_S[0:3], FCCU_RF_E[0:3] to FCCU_NCF_E[0:3], FCCU_RF_TOE[0:3] to FCCU_NCF_TOE[0:3]. Added FCCU_EOUT_SIG_EN[0:3] registers Event of all CTU peripherals changed in *Table 35: DMAMUX peripheral DMA request to input source mapping* **Reset and Boot** *Section 1.2: Hardware reset stage*: Suppress text “and functional ” in the second sentence *Figure 5: System boot sequence*: Updated picture with block text “BAR” replaced by “BAM”’ and removed text “1x0 = FlexRay boot” in C90 products only block. *Figure 6: CPU reset address logic*: Added the figure *Table 3: Destructive reset events*: Added ESR0 pin row *Table 4: Functional reset events*: Removed D_SSCM module row *Table 7: PORST electrical characteristics*: Suppressed table note 2 (LBIST not implemented) *Section 1.6*: added section included *Figure 4* *Section 1.7*: Updated paragraphs included *Figure 6* *Section 1.8*: Updated paragraphs included *Figure 7* picture quality *Section 1.9*: Updated section

Table 953. Document revision history(Continued)

Date	Rev.	Changes
29-May-2014	3	Section application execution removed *Section 1.10*: Added performances included *Table 84* *Table 3*: Added SSCM module at D[14] input Changed input D[11:22] to D[11:13] Set D[15:22] inputs to “Reserved” *Table 4*: Changed reset Module F_WAKEUP configuration to P[E],P[E],P[D];P[E] and P[D] (was 0,P[E],P[D], X and X) Updated signal description “Online Self Test completion OR ” to reset module F_SOFT_FUNC Updated table footnote (5) with “ OR when ONLINE Self TEST completes” *Table 7*: Removed CMOS type rows *Section 1.5*: updated text of list number 9 *Section 1.8*: Updated text note “RCHW_LOC_NUM” SSCM IP configuration parameter shall be set to 6” *Table 14*: deleted all rows with boot time TBD **DCF Records** *Figure 1*: removed offsets from 0x01 to 0x07 *Figure 6*: added one more bit to offset value and set it to '1' *Table 5*: Corrected Miscellaneous client PMC_REE bits DCF CS[14:0] = 000_0000_1000_0000 (was 000_1000_0000_0000) and changed its reset value from 1 to 0 with table foot note (2) “Reset value will depend on the TEST DCF programmed value.”. Add Miscellaneous client “ESR0_CFG” at DCF_address[16:2] = 000000001111110 (Reset = 0x0, ...) *Table 7*, in field descriptions: Replaced RES_VD14[LVD14_AD] by RES_HV1[RES14_AD] (bit 25) Replaced RES_VD13[HVD13_C] by RES_HV1[RES13_C] (bit 26) Replaced RES_VD11[LVD11_AD] by RES_HV0[RES11_AD] (bit 27) Replaced RES_VD11[LVD11_FL] by RES_HV0[RES11_FL] (bit 28) Replaced RES_VD11[LVD11_C] by RES_HV0[RES11_C] (bit 29) Replaced RES_VD6[HVD6_C] by RES_LV1[RES6_C] (bit 30) Replaced The RES_VD3[LVD3_C] by RES_LV0[RES3_C] (bit 31) **Power Management Controller** Reordered paragraphs in *Section 1.1* Updated *Table 4* and *Table 9* Added note to *Table 9* Rotated and updated *Table 107* Added *Section 10.4.7: BIST details* Added note to *Section 10.4.7: BIST details*

Table 953. Document revision history(Continued)

Date	Rev.	Changes
29-May-2014	3	**Security** Removed phrase about SRAM and cache initialisation from *Section 11.1* Added a cross-reference to *Chapter 1: Password and Device Security Module (PASS)* in *Section 11.3: Detailed security information* **SIUL** – offset changed from 00x0A40–0x123F to 0x0A40–0x123C in *SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR0 – SIUL2_MSCR1023)* – SIUL2 GPIO Pad Data In Register 0 – GPIO Pad Data In Register 79 (SIUL2_GPDI0 – SIUL2_GPDI79) renamed to *SIUL2 Parallel GPIO Pad Data In Register (SIUL2_PGPDI0 – SIUL2_PGPDI4)* – *SIUL2 Masked Parallel GPIO Pad Data Out Register (SIUL2_MPGPDOn)* offset changed from 0x1780–0x179C to 0x1780–0x17A0 – 0x17A0–0x17FC reserved field changed to 0x17A4–0x17FC Updated reserved end address offsets: 0x0000-0x0003 0x0024-0x0027 0x002C-0x002F 0x0034-0x0037 0x003C-0x003F Added /16/8 to supported access from addresses 0x0240 to 0x123C *Table 121*: Added text in register description “that is T(DELAY) = ... expects the worst filter delay value).” Changed n= -1 to 3 (was n=0 to 4) *Table 123*: Added figure footnote “HYS bit in MSCR registers from offset 0x240 to 0x37C is set at the time of Reset” add this footnote to HYS field (bit 13) Removed INV field (bit 16) and set it as reserved bit field *Table 126*: Updated INV field description text Added *Table 127* and *Table 115* to insert specific Reset values of some MSCRn registers *Figure 68*: changed PGPDO to PGPDI *Figure 54*: Changed register field EIF[5:0] to 6 fields from EIF[5] to EIF[0] and changed W fields to separated w1c fields *Figure 60*: Changed MAXCNT[3:0] field name to MAXCNT *Figure 61*: Changed IFCP[3:0] field name to IFCP *Figure 66*: Changed start offset to 0x1500 (was 0x1300) **XBAR** New MGPCRn registers added in *Section 14.1.2.3: Master General Purpose Control Registers (XBAR_MGPCRn)* Modified registers in *Section 14.1.2.1: XBAR Priority Registers Slave (XBAR_PRSn)* Modified registers in *Section 14.1.2.2: XBAR Control Register (XBAR_CRSn)* Removed Section “Priority assignment” in *Section 14.2.3* Removed ‘locked transfer’ feature from *Section 14.2.3*

Table 953. Document revision history(Continued)

Date	Rev.	Changes
		XBIC *Figure 80*: Added footnote “This bit is writable but has no impact” on SE5-7 and ME2-7 fields Deleted footnote at ME2 bit 10 *Figure 82*: Added footnote “This bit is writable but has no impact” on all DPME0-7 and DPSE0-7 fields Deleted footnote “Set 0 to DPME5-7 and DPSE3-7” *Section 15.1*: Added text “The attribute information contains: -Several AHB signals such as htrans, hmaster, hwrite, hsize, hprot, hunalign, hburst -decorated attributes 'hdecorated/hdecor' and a note “Decorated attribute signals are all zeros.” *Section 15.2*: Suppressed text bullet “EDC (72,64) code protects against single and double bit errors” *Table 126*: Replaced SEn and MEn by SE0-7 (bits 0 to 7) and ME0-7 (bits 8 to 15) *Table 128*: Replaced DPSEn and DPMEn by DPSE0-4 (bits 1 to 5) and DPME0-2 (bits 9 to 11) *Section : XBIC Error Status Register (XBIC_ESR)* and *Section 15.6: Functional description*: Replaced DPSE0-7 and DPME0-7 by DPSE0-4 and DPME0-2 Added sentence in the 2nd paragraph “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.”
29-May-2014	3	**PBridge** *Section 16.1*: Added note “z0 is a 32-bit microprocessor. 64-bit accesses are not possible.” *Table 145*: Removed AIPS_MPRB (Master Privilege Register B) register. Added Reserved spaces at offset addresses 0x004-0x1F, 0x030-0x3F, 0x084-0xFFand 0x130-0x133. Added Reserved spaces at offset addresses 0x020-0x2F, 0x040-0x83, 0x120-0x12F and 0x134-0x13F with a note “The following address spaces, when programmed, will not terminate in an error. Software should not program these address spaces” Added AIPS_OPACRA-AF (Off-platform Peripheral Access Control Register n) registers from 0x140 to 0x1BC offset addresses Added *Section 16.2.3: Off-Platform Peripheral Access Control Registers (PBRIDGE_OPACRx)* included text and registers as described in *Table 149: OPACR memory map*, *Figure 88: Off-Platform Peripheral Access Control Registers (AIPS_OPACRx)*, *Figure 87: OPACRn fields* and *Table 150: OPACRn field descriptions* *Table 147*: Changed offset 0x110 to Reserved space 0x110-0x113 with table note “Access to this address space will terminate in an error. **SMPU** Updated all register offset definitions *Table 139*: Re-formated table in line with register names and addresses *Section 17.6.2*: Added note “In case of ... pre-fetching in the pipeline).”

Table 953. Document revision history(Continued)

Date	Rev.	Changes
29-May-2014	3	*Figure 94*: Replaced M2PE, M2SM and M2UM reg fields (bits 14 to 19) by “Reserved” with figure note (1) Added M1UM field Deleted “r w x” in M0UM field Changed bit 10 of reset value for n=0, from ‘0’ to ‘1’ (n=0 reset value is 0x0061_8600) *Table 145*: Removed M2PE, M2SM and M2UM field and description rows *Section 17.6.4.4: SMPU Region Descriptor n – Word 3 (SMPU_RGDn_WORD3)*: Replaced “... “ by “, then SMPU_RGDn.Word2” in the second paragraph *Table 151*: Suppressed CPU TCM backdoor row *Section 17.6.4.1: SMPU Region Descriptor n – Word 0 (SMPU_RGDn_WORD0)* and *Table 155* description: Suppressed “0-modulo-32 byte” *Section 17.6.4.2: SMPU Region Descriptor n – Word 1 (SMPU_RGDn_WORD1)* and *Table 156* description: Suppressed “31-modulo-32 byte” *Figure 94*: added figure footnote (1) to Read field from bit 0 to bit 13 *Section 17.6.4.3: SMPU Region Descriptor n – Word 2 FMT0 (SMPU_RGDn_WORD2_FMT0)* and *Section 17.6.4.4: SMPU Region Descriptor n – Word 3 (SMPU_RGDn_WORD3)*: Deleted “and always works in Supervisor Mode” in text notes *Figure 94* and *Figure 96*: Added M1UM field and description (bit 23:25) Replaced M2PE, M2SM and M2UM reg fields (bits 14 to 19) by “Reserved” with figure note (1) *Section 17.7.1.1: Access evaluation: hit determination*: Added sections *Section 17.7.1.1.1: Evaluation 1a* and *Section 17.7.1.1.2: Evaluation 1b* *Figure 93*: Changed reset values to 0b00000 of bit[0:4] for (n=0) and of bit [27:31] for (n>0) (was 0b11111 for both) *Figure 94* and *Figure 96*: added figure footnote (1) to ‘0’ of Read field bit 20 **INTC** *Section 18.1: Introduction*: Updated this section *Section 18.4.1: Software vector mode*: Replaced HVEN_PRCn with HVENn *Table 164: INTC_BCR field descriptions*: Added note to HVENn field *Figure 102: INTC Interrupt Acknowledge Register for Processor 0 (INTC_IACKR0)*: Removed footnote *Section 18.4: Modes of operation*: Added note “The value of n = 0 in this chapter, since this device contains only one core” *Table 170: INTC_PSRn field descriptions*: Updated the PRC_SELn, and SWTn field descriptions. **eDMA** *Table 173: DMA memory map*: Removed columns Access and Reset value. Removed Absolute address. Added full register names when missing. Updated “Address” by “Offset” in all bitmaps and removed Absolute address. *Section 19.3.1: Control Register (DMA_CR)*: Updated register and fields description with GRP0PRI (bits 22 & 23) and EBW (bit 31).

Table 953. Document revision history(Continued)

Date	Rev.	Changes
29-May-2014	3	*Figure 137* and *Table 199*: Changed bit field LINKCH position to [3:6] (was [1:4]) and reserved bit fields to [1:2] (was [5:6]) *Table 189: DMA_DCHPRIn field descriptions*: CHPRI: removed GRPPRI in Note. *Section 19.3.22: TCD Signed Minor Loop Offset (Minor Loop Enabled and Offset Disabled) (DMA_TCDn_NBYTES_MLOFFNO)*: Added missing field descriptions table **DMACHMUX** *Table 216: DMACHMUX memory map* and *Figure 150: Channel Configuration Registers (CHCFGn)*, introduced #NCH-1 variable usage in the offset definitions **Clocking** *Section 23.1*: Removed the note about protocol clocks; Removed the reference to the high speed domain; Removed “if backup clock is enabled”. Clarified that the internal clocks are generated from both PLLs. Included CMU_3 in *Section 23.7.1.1: Clock input sources* and *Table 228*. Changed the source of the clock out divider in *Figure 158*. *Table 223*: Added CGM_SC_DC2 and removed CGM_AC1_DC1 CGM_AC1_SC. Specified the working limitations for the CORE_CLK and PBRIDGE0_CLK in the note that follows *Table 224*. Reference clock to PLL1 as well as PLL0 in *Section 23.5.2*. Updated bit numbering for MSB=0 in *Section 23.5.2.1: XOSC startup*. Added the precision that a failure is the loss of IRCOSC clock in *Section 23.8.2*. **PLLDIG** *Section 24.5.2.7: PLL1 Frequency Modulation Register (PLL1FM)*: Removed PLL2 *Table 238*: Updated DTHDIS field description with MSB and LSB definitions *Section 24.6.2*: Updated note: “Each time the input reference clock, PLL0CR[PREDIV], PLL0CR[MFD] or PLL1CR[MFD] values are modified, a power down pulse is required to re-lock the analog circuitry of the PLLs.” *Section 24.7*: Replaced text in bullets “Provide a power down pulse” by: - bullet 2: text “Power up PLL0” - bullet 3: text “Power up PLL1” *Section 24.3*: Removed bullet with text “User selectable... reset upon loss of lock” *Table 230* updated reserved space from 0x000C-001C to 0x000C-001F and from 0x0034-0038 to 0x0034-003B *Figure 165* and *Figure 168*: Replaced LOLRE by ‘0’ in bit field 29 *Table 231* and *Table 234*: Suppressed LOLRE field and description, and set bit 29 to Reserved *Section 24.5.2.3: PLL0 Divider Register (PLL0DV)*: Suppressed text “ either a reset or”

Table 953. Document revision history(Continued)

Date	Rev.	Changes
29-May-2014	3	*Section 24.5.2.7: PLL1 Frequency Modulation Register (PLL1FM)* and *Section 24.5.2.8: PLL1 Fractional Divider Register (PLL1FD)*: Suppresed text “ lock and the device being reset (if PLL1CR[LOLRE] = 1)” *Section 24.6.2*: Suppressed sentence “ To prevent an immediate reset... any of the above operations.” Changed PLL0CR to PLL0DV and PLL1CR to PLL1DV in the latest note of this paragraph. *Section 24.6.3*: Suppressed text “ reset and” in the second paragraph *Figure 173*: Shift fpll1_phi label of Down Spread graph to its peak. **CGM** *Table 247*: Added following registers: CGM_SC_DIV_RC (0x07D0) CGM_DIV_UPD_TYPE (0x07D4) CGM_DIV_UPD_TRIG (0x07D8) CGM_DIV_UPD_STAT (0x07DC) CGM_SC_DC2 (0x07EC) Added Reserved fields rows Changed address offset from 0x07E0-0x07FF to 0x07E0-0x07FF (Reserved) Suppressed row at address offset 0x082C (CGM_AC1_DC1) Changed “Reserved” address from 0x0830-0x083F to 0x082C-0x083F Changed register format from 8-bits to 32-bits (by adding bits from 9 to 31 with “always read 0” format. *Section 26.2.1.11: System Clock Divider Ratio Change Register (CGM_SC_DIV_RC)*: Added register fields and descriptions *Section 26.2.1.12: Divider Update Type Register (CGM_DIV_UPD_TYPE)*: Added register fields and descriptions *Section 26.2.1.13: Divider Update Trigger Register (CGM_DIV_UPD_TRIG)*: Added register fields and descriptions *Section 26.2.1.14: Divider Update Status Register (CGM_DIV_UPD_STAT)*: Added register fields and descriptions *Section 26.2.1.18: System Clock Divider 2 Configuration Register (CGM_SC_DC2)*: Added register fields and descriptions *Section 26.3.2.2: Divider update triggering*: Added section including *Section 26.3.2.2.1: Immediate divider update* and *Section 26.3.2.2.2: Software-triggered divider update* *Figure 203*: Change address 0x0000 by 0x0810 Changed variable SYS CLK DIV 0 value “fast XBAR clock” by “XBAR clock” *Section 26.3.1.4: System clock dividers*: Added bullet with text: “CLOCKOUT: controlled by the CGM_SC_DC2 register” *Table 282*: Added row for Divider = 2 Added text “System clock divider 2 ... multiple of 1 and 2” in example 1 Added text “The dividers should be .... greater than 2:1 is not supported.” and note “During configuration ... on DMA module.” to Caution

Table 953. Document revision history(Continued)

Date	Rev.	Changes
29-May-2014	3	*Figure 192*: Changed bit 0 field name SYS_UPD_TYPE (was SYS_DIV_RATIO_CHN) Added AUX1_UPD_TYPE (bit 30) and AUX0_UPD_TYPE (bit 31) *Table 259*: Added fields and descriptions for bit 30 named AUX1_UPD_TYPE and bit 31 named AUX0_UPD_TYPE *Figure 194*: Added AUX1_UPD_STAT (bit 30) and AUX0_UPD_STAT (bit 31) *Table 261*: Added fields and descriptions for bit 30 named AUX1_UPD_STAT and bit 31 named AUX0_UPD_STAT *Figure 195*: Changed Reset value of SWTRG field to 100 (was 000). *Figure 197*: Changed Reset value of DIV field to 0001 (was 0000). *Table 264*: Changed DIV field description by replacing “CLOCKOUT” by “AIPS clock” *Figure 214*: Added CGM_SC_DC2 register and clock divider blocks, and CLOCKOUT output *Figure 220*: Removed CGM_AC1_DC1 register and clock divider blocks, and CLOCKOUT output **XOSC** Updated *Section 27.2.2: Oscillator startup delay* Updated *Section 27.3.1.1: XOSC control register (XOSC_CTL)* **IRCOSC** Updated *Section 28.3.1.1: IRCOSC control register (IRCOSC_CTL)*: Changed Read values Removed footnotes **PRAM** Replaced “AXBS” with “XBAR” in *Figure 222: Simplified platform block diagram* Updated *Section 27.2.1: Platform RAM configuration register 1 (PRAMC_PRCR1)* Replaced “64-bit interface” with “32-bit interface” throughout the document. *Table 276: RAM safety feature outputs*: Replaced “pram_addr_co[17:0]” with “pram_addr_co[13:0]” *Section : E2EECC considerations on less-than-32-bit write transactions*: Added note *Section 27.3.1: Read/Write introduction*: Updated the note *Section 27.3.2: Writes*: Replaced “doubleword” with “word” *Section 27.4: Initialization/application information*: Removed the last sentence. *Section 27.2.1: Platform RAM configuration register 1 (PRAMC_PRCR1)*: Updated bits 22, 23, and 24. Were reserved before.

Table 953. Document revision history(Continued)

Date	Rev.	Changes
29-May-2014	3	**PFLASH** Replaced “2-way mini-cache for the flash controller” with “4-way mini-cache for the flash controller” in *Section 30.1: Introduction*, *Section 30.2: Features*, and *Section 30.5: Functional description* *Figure 226: Platform-centric block diagram with flash memory controller*: Updated the figure Replaced “AXBS” with “XBAR” Added “BDRM” field in *Section 30.4.1.3: Platform Flash Configuration Register 3 (PFCR3)* Replaced “Figure:Flash memory controller 4-entry, 2-way mini-cache organization” with *Figure 236: Flash memory controller 4-entry, 4-way mini-cache organization* *Table 296: PFCR3 field descriptions*: Updated BDRM field description. *Section 30.5.8: Line read buffers and prefetch operation*: Replaced “two way set-associative” with “four way set-associative” **Embedded Flash Memory** *Section : Test mode disable* TMDBS write address expressed ad UTEST + offset. *Table 287: MCR field descriptions*: added footnote *Initial maximum (All temperatures) program and erase times* Removed word “High” in the description of registers: *Section : Lock 2 Register (LOCK2)*, *Section : Select 2 register (SEL2)* and *Section : Over-program protection 2 register (OPP2)* SBCE and AIBPE descriptions update in *Table 299* *Section 29.4: Functional description*: replaced k by KB when referring to the address space block sizes Changed LLKx to LOWLOCK[x] (x = 0 to 8) Changed 256KLKx to A256KLOCK[x] (x = 0 to 3) Changed LSLx to LOWSEL[x] (x = 1 to 8) Changed 256KSLx to A256KSEL[x] (x = 0 to 3) Changed HLKx to HIGHLOCK[x] (x = 0 to 7) Changed HSLx to HIGHSEL[x] (x = 0 to 7) Changed all register-field syntax from register_name[field_name] to register_name.field_name Changed UMISRx to UMx (x = 0 to 9) Updated *Section 29.3.2: User register descriptions* *Table 286: Flash User Registers memory map*, updated description of: LOCK0, LOCK1, LOCK2, SEL0, SEL1, SEL2. Removed Access and Reset value columns

Table 953. Document revision history(Continued)

Date	Rev.	Changes
29-May-2014	3	**ADC Configuration** Renamed eCTU to CTU in *Figure 270*. Removed reference to external triggers from *Section 33.2.4* and *Figure 272*. Updated *Figure 273* for matched sw impedance. *Table 336: SARADC_B analog test channel assignment*, added PTAT in the description of channel 120 and CTAT in the description of channel 121 Removed the prefix “SARADCx_” in the names of all registers *Table 340: SARADC_B test channel register definitions*: SARADCB_TCWSELR0 and SARADCB_TCWSELR2: deleted SARADCB_TCWSELR1: Removed fields: WSEL_CH110; WSEL_CH108; WSEL_CH106 Changed field size to 2 bits: WSEL_CH109; WSEL_CH104 Added field: WSEL_CH105 SARADCB_TCWSELR3: Removed field: WSEL_CH122 **SARADC** *Table 345: MCR field descriptions*, replaced “JTRGSEL[1:0]” with “JTRGSEL[3:0]” *Section 34.5: Memory map and registers*, added note: “See the Device Configuration chapter for any chip-specific register information for this module.” -Added FCERR bit to ICDR*34.5.1.16: Internal Channel Data Registers 0–95 (ICDR0–ICDR95)*,TCDR*34.5.1.29: Test Channel Data Registers 96–127 (TCDR96–TCDR127)* *Figure 281: Main Configuration Register (MCR)*: Added footnotes to the figure *Figure 285: Interrupt Mask Register (IMR)*: Added footnote “Not functionally used, but can be written by software” **CTU** *Figure 308: Scheduler subunit*: Added 2 FIFOs. *Section 35.6: Power safety mode*: Removed MDIS section. *Section 35.8: Memory map*: Removed CTUPCR register and removed section Cross triggering unit power control register *Section 35.8.14: FIFO status register (FST)*: updated reset values of EMP0/1/2/3 bits with ‘1’. Replaced ‘r1c’ by ‘w1c’. *Section 35.8.18: Cross triggering unit interrupt flag register (CTUIFR)*: replaced ‘r1c’ by ‘w1c’

Table 953. Document revision history(Continued)

Date	Rev.	Changes
29-May-2014	3	*Section 35.8.15: FIFO Right aligned data n (FRn)* and *Section 35.8.16: FIFO signed Left aligned data n (FLn)*: increased length of DATA field to 12. Updated Reset value to 0xXXXX_XXXX. *Table 397: FIFO registers*: Updated Reset value to 0xXXXX_XXXX for registers FR*n* and FL*n*. *Section 35.8.10: Commands list registers (CLRn)*: updated bit 1 value: was 'FC', is 'LC' (Last command bit). **Temperature Sensor** *Table 222: TSENS calibration constants*: Inserted table footnote. **eTimer** Changed PIPS to SIPS in gated count mode paragraph Corrected the offset values of the registers. In the description of CTRL1n.PRIRC, supported only "Auxiliary input #0 pin" and "Auxiliary input #1 pin" *Table 421: eTimer Memory Map*, replaced "Watchdog Time-out Low Register" with "Watchdog Time-out Low Word Register" and "Watchdog Time-out high Register" with "Watchdog Time-out High Word Register" Corrected reset values of CTRL3 register. *Figure 348: Variable Frequency PWM Mode Timing*: Updated the text in the figure *Figure 349: Compare Register and OFLAG Timing*: Replaced COM1 with COMP1 **SWT** *Section 38.1.2: Features*, removed "Programmable selection of system or oscillator clock, for timer operation" bullet. *Figure 382: SWT Control Register (SWT_CR)*, changed to '1' the read value of bit 28. *Table 458: SWT_SR field descriptions* and *Table 460: SWT_SK field descriptions*, Replaced "If SWT_CR[KEY] is set" with "SWT_CR[SMD] = 01". *Section 38.4.1.2: SWT_TO register*,added "see the device configuration chapter" *Section 38.4.5: Time-out*, replaced "The interrupt then reset bit (SWT_CR[ITR])" with "The SWT_CR[ITR] ("Interrupt Then Reset" bit)" **PIT** Added Chain Mode Bit in *Figure 387: Timer Control Register (PITx_TCTRLn)*

Table 953. Document revision history(Continued)

Date	Rev.	Changes
29-May-2014	3	**FlexCAN** Aligned bits 19, 20 and 21 in *Table 478* with the corresponding fields LPB, TWRN_MSK and RWRN_MSK Added *Table 483* fields and descriptions in line with reg *Figure 408* *Section 40.3.4.11: Rx Individual Mask Registers (RXIMRn)*: Renamed RXIMR0-RXIMR31 by RXIMRn Changed offset values in *Figure 412* Removed table RXIMR0-RXIMR31 addresses. *Section 40.4.7: Rx FIFO*: Replaced “four frames” by “five frames” in the 11th sentence. *Table 486*: deleted description text “BUF31M–BUF0M —” *Table 476*: Replaced “note that all...” by “note: All...” in MCR.IDAM description *Section 40.1.2*: Deleted “with programmable wake up” *Section 40.3.1*: Deleted note “The individual Rx Mask per Message Buffer feature may not ... value of the BCC bit” *Table 478*: Deleted note “This clock selection feature may not... no effect on the module operation.” *Section 40.3.4.11: Rx Individual Mask Registers (RXIMRn)*: Deleted note “The individual Rx Mask per Message Buffer feature may not ... value of the BCC bit” *Section 40.4.5*: Deleted note “The individual Rx Mask per Message Buffer feature may not ... value of the BCC bit.” *Section 40.4.10*: Replaced “, and the Wake Up interrupt ... located in the MCR” by page note (d) *Section 40.5.1*: Deleted “and in MCR for Wake-Up interrupt” *Table 468*: Added a column stating the access type “S/U” (user/supervisor) of registers *Figure 402*: Added note (1) for the reserved bits 5, 9, 12 and13 (MCR) Changed register access mode from “user” to “supervisor” *Section 40.1.3* (in Listen-Only mode bullet) and *Table 478* (in CTRL.LOM description bullet): Removed [Ref. 1] *Section 40.1.3* (in Loop-Back mode bullet): Replaced “fed back” by “feedback” *Table 487*: Changed text “4 out of 6 buffers” by “5 out of 6 buffers” in field BUF61 (bit 25) **DSPI** All CSI, DSI, TSB and ITSB related content removed. Updated PCSIS field in *Section 41.2.9.1: DSPI Module Configuration Register (DSPIx_MCR)* Updated PCS field in *Section 41.2.9.8: DSPI PUSH FIFO Register In Master Mode (DSPIx_PUSHR)*

Table 953. Document revision history(Continued)

Date	Rev.	Changes
29-May-2014	3	Replaced FMSZ[5] field by Reserved bit in *Section 41.2.9.4: DSPI Clock and Transfer Attributes Register (In Slave Mode) (DSPIx_CTAR0_SLAVE)* and in *Section 41.2.9.5: DSPI Clock and Transfer Attributes Register (In Slave Mode) (DSPIx_CTAR1_SLAVE)* *Figure 445: Example of Fast Continuous Selection Format*: Updated the figure **LINFlexD** *Figure 449: Block diagram*: Improved figure quality and Updated “DRF/RFE” to “DRF_RFE” and “DTF/TFF“ to “DTF_TFF“ *Table 537: LINSR field descriptions*: in “AUTOSYNC_COMP” field description, replaced “AUTOSYNC_COMP“ with “Autosynchronization Completed“ *Figure 493: UART Mode Control Register (UARTCR)*: Renamed field relative to bit 12, field relative to bits 13–14 to “SBUR” (was “SBUR[1:0]“) and fields relative to bits 16–18 and bits 19–21 respectively to “TDFL_TFC” and “RDFL_RFC“ (were “TDFL/TFC” and “RDFL/RFC“) Removed footnotes “Register bit is a read/write field.“ and “Register bit is read-only field.“ Added footnote “When TFBM = 0, the field is...” and footnote “When RFBM = 0, the field is...” Changed footnote “Register bit can be read...“ to “This field can be read...“ *Table 539: UARTCR field descriptions*: Renamed fields relative to bits 16–18 and bits 19–21 respectively to “TDFL_TFC” and “RDFL_RFC“ (were “TDFL/TFC” and “RDFL/RFC“) In “DTU_PCETX” field description, changed sentence “When in_ps_bit_remap is cleared” to “When in_ps_bit_remap is reset” *Figure 494: UART Mode Status Register (UARTSR)*: Changed fields relative to bits 29 and 30 respectively to “DRF_RFE” and “DTF_TFF“ (were “DRF/RFE” and “DTF/TFF“) Removed footnotes “Register bit can be read/cleared...“ and “Register bit is read-only field.“ Added footnote “When RFBM = 0, the field is...” and footnote “When TFBM = 0, the field is...” *Table 539: UARTCR field descriptions*: changed fields relative to bits 29 and 30 respectively to “DRF_RFE” and “DTF_TFF“ (were “DRF/RFE” and “DTF/TFF“) Updated the description of field “autosync_comp” from “Only after this bit is set, the contents of LINIBRR and LINFBRR registers can be read in autosynchronization mode” to “Only after this bit is set, the contents of LINIBRR and LINFBRR registers should be read in autosynchronization mode” in *Table 537: LINSR field descriptions* *Figure 490: LIN Interrupt enable register (LINIER)*: changed field name relative to bit 28 to “DBEIE_TOIE” (was “DBIE/TOIE”) *Table 536: LINIER field descriptions*: changed field name relative to bit 28 to “DBEIE_TOIE” (was “DBEIE/TOIE”)

Table 953. Document revision history(Continued)

Date	Rev.	Changes
29-May-2014	3	*Figure 489: LIN Control Register 1 (LINCR1)*: changed bits 20–23 (were four 1-bit fields: “MBL3–MBL0”; are one 4-bit field: “MBL”) *Figure 491: LIN Status Register (LINSR)*: Changed bits 16–19 (were four 1-bit fields: “LINS3–LINS0”; are one 4-bit field: “LINS”) Changed field name relative to bit 24 to “RXBUSY” (was “RXbusy”) *Table 537: LINSR field descriptions*: changed field name relative to bit 24 to “RXBUSY” (was “RXbusy”) *Section 42.4.2.5: UART Mode Control Register (UARTCR)*: changed field names relative to bits 26 and 27 to respectively “RXEN” and “TXEN” (were “RxEn” and “TxEn”) *Figure 494: UART Mode Status Register (UARTSR)*: changed bits 18–21 (were four 1-bit fields: “PE3–PE0”; are one 4-bit field: “PE”) *Table 540: UARTSR field descriptions*: updated description of field “PE” *Figure 495: LIN Time-Out Control Status Register (LINTCSR)*: changed bits 24–31 (were eight 1-bit fields: “CNT7–CNT0”; are one 8-bit field: “CNT”) *Figure 496: LIN Output Compare Register (LINOCR)*: changed bits 16–23 (were eight 1-bit fields: “OC27–OC20”; are one 8-bit field: “OC2”) changed bits 24–31 (were eight 1-bit fields: “OC17–OC10”; are one 8-bit field: “OC1”) *Figure 497: LIN Time-Out Control Register (LINTOCR)*: changed bits 20–23 (were four 1-bit fields: “RTO3–RTO0”; are one 4-bit field: “RTO”) changed bits 25–31 (were seven 1-bit fields: “HTO6–HTO0”; are one 7-bit field: “HTO”) *Figure 498: LIN Fractional Baud Rate Register (LINFBRR)*: changed bits 28–31 (were four 1-bit fields: “FBR3–FBR0”; are one 4-bit field: “FBR”) *Figure 499: LIN Integer Baud Rate Register (LINIBRR)*: changed bits 12–31 (were twenty 1-bit fields: “IBR19–IBR0”; are one 20-bit field: “IBR”) *Figure 500: LIN Checksum Field Register (LINCFR)*: changed bits 24–31 (were eight 1-bit fields: “CF7–CF0”; are one 8-bit field: “CF”) *Figure 502: Buffer Identifier Register (BIDR)*: changed bits 16–21 (were six 1-bit fields: “DFL5–DFL0”; are one 6-bit field: “DFL”) changed bits 26–31 (were six 1-bit fields: “ID5–ID0”; are one 6-bit field: “ID”) *Figure 505: Identifier Filter Enable Register (IFER)*: changed bits 16–31 (were sixteen 1-bit fields: “FACT15–FACT0”; are one 16-bit field: “FACT”) *Figure 506: Identifier Filter Match Index (IFMI)*: changed bits 27–31 (were five 1-bit fields: “IFMI4–IFMI0”; are one 5-bit field: “IFMI”) *Figure 507: Identifier Filter Mode Register (IFMR)*: changed bits 24–31 (were eight 1-bit fields: “IFM7–IFM0”; are one 8-bit field: “IFM”) *Figure 508: Identifier Filter Control Register (IFCR2n)*: changed bits 16–21 (were six 1-bit fields: “DFL5–DFL0”; are one 6-bit field: “DFL”)

Table 953. Document revision history(Continued)

Date	Rev.	Changes
29-May-2014	3	changed bits 26–31 (were six 1-bit fields: “ID5–ID0”; are one 6-bit field: “ID”) *Figure 509: Identifier Filter Control Register (IFCR2n+1)*: changed bits 16–21 (were six 1-bit fields: “DFL5–DFL0”; are one 6-bit field: “DFL”) changed bits 26–31 (were six 1-bit fields: “ID5–ID0”; are one 6-bit field: “ID”) *Figure 511: UART Preset Timeout Register (UARTPTO)*: changed bits 20–31 (were twelve 1-bit fields: “PTO11–PTO0”; are one 12-bit field: “PTO”) *Figure 512: UART Current Timeout Register (UARTCTO)*: changed bits 20–31 (were twelve 1-bit fields: “CTO11–CTO0”; are one 12-bit field: “CTO”) *Figure 513: DMA Tx Enable Register (DMATXE)*: changed bits 16–31 (were sixteen 1-bit fields: “DTE15–DTE0”; are one 16-bit field: “DTE”) *Figure 514: DMA Rx Enable Register (DMARXE)*: changed bits 16–31 (were sixteen 1-bit fields: “DRE15–DRE0”; are one 16-bit field: “DRE”) *Section 42.3.2.5.1: Header error*: in Case 2, changed “LINCR1 = 1“ to “LINCR1 = 0“ *Table 539: UARTCR field descriptions*: Changed description of field “DTU_PCETX” (bit 12) Added note to description of fields “WL0“, “WL1“, “PC0“, and “PC1“ *Table 557: UARTPTO field descriptions*: In PTO[11:0] field description, changed *A zero-value is forbidden* to *A value of 0, 1 or 2 is forbidden* *Section 42.4.2.24: UART Current Timeout Register (UARTCTO)*: Reset list item *UARTCTO becomes equal to UARTPTO* changed to *UARTCTO becomes equal to UARTPTO-1* *Table 539: UARTCR field descriptions*, updated description of TDFL_TFC and RDFL_RFC *Table 540: UARTSR field descriptions*, updated description of DRF_RFE and DTF_TFF Removed Section “External signal description” Added notes in *Section 42.3.4.7: UART transmitter*. In *Section 42.3.2.5.6: Timeout error*, replaced ocr_1 with ocr_2 In *Figure 458: Incomplete response (for example, missing checksum)*, and *Figure 459: No response*, replaced OCR1 with OCR2. In *Table 535: LINCR1 field descriptions*, updated the description of SLEEP and INIT fields. In *Table 539: UARTCR field descriptions*, updated the description of DTU_PCETX, TDFL_TFL, and PCE fields. Updated the notes in *Section 42.4.2.24: UART Current Timeout Register (UARTCTO)* In Figure *Figure 510: Global Control Register (GCR)*, updated the 2nd note. In *Section 42.4.2.8: LIN Output Compare Register (LINOCR)*, updated body text. In *Table 542: LINOCR field descriptions*, added description to OC1 and OC2 fields. In *Section 42.3.4.8: UART receiver*, updated the last bullet. Added *Section 42.3.5.10: Use cases and limitations* In *Table 541: LINTCSR field descriptions*, added 2nd and 3rd note to CNT field.

Table 953. Document revision history(Continued)

Date	Rev.	Changes
29-May-2014	3	Removed figure “Frame timeout”. In *Figure 538: Response timeout*, replaced OC1 with OC2. *Section 42.3.4.7: UART transmitter*: Added notes “Invalid writes to BDRL which generates IPS transfer error, also triggers data transmission at the same time, and 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” **RGM** *Figure 547: Functional Event Status Register (RGM_FES)*: note added for reset value of bit F_ST_DONE *Figure 548: Functional Event Reset Disable Register (RGM_FERD)*: bits D_FCCU_SOFT and D_WAKEUP updated to writable *Figure 549: Functional Event Alternate Request Register (RGM_FEAR)*: added AR_WAKEUP bit in position 30 *Figure 551: Functional Event Short Sequence Register (RGM_FESS)*: bit SS_FCCU_SOFT updated to writable *Figure 554: Peripheral Reset Register 0 (RGM_PRST0)* to *Figure 561: Peripheral Reset Register 7 (RGM_PRST7)*: added field descriptions table for each RGM_PRSTx register bitmap *Figure 559: Peripheral Reset Register 5 (RGM_PRST5)*: moved FCCU_RST bit to position 22 instead of 23 *Figure 561: Peripheral Reset Register 7 (RGM_PRST7)*: moved CTU_RST bit to position 11 instead of 7, and removed field DSPI_3_RST *Table 562: MC_RGM memory map*: added registers RGM_PSTAT*n*. *Section 43.3.2: Register descriptions*: added registers RGM_PSTAT*n* *Section 43.3.2.12: Peripheral Reset Register 0 (RGM_PRST0)* to RGM_PSTAT7 *Section 43.3.2.19: Peripheral Reset Register 7 (RGM_PRST7)* with the same fields than the RGM_PRST*n* registers. *Functional Event Alternate Request Register (RGM_FEAR)*: added AR_FCCU_SOFT bit in position 25. *Functional Reset Escalation Threshold Register (RGM_FRET)*: updated ‘Dest’ Reset for bit 5 to ‘1’. *Figure 570: MC_RGM State Machine*: appended to DRET ‘field of RGM_DRET register’. *Section 43.4.1: Reset state machine*: updated the clock cycles durations for PHASE0 Phase update from three to one hundred and twenty five for PHASE1 Phase update from three hundred fifty to seven hundred and fifty for PHASE3 Phase update from forty to one hundred and sixty Added PMC meaning:‘Power management Controller’. *Section 43.4.2.1: External power-on reset—PORST*: removed part of paragraph. *Section 43.4.4: Alternate event generation*: updated RGM_D/FERD to RGM_DERD/FERD, RGM_D/FEAR to RGM_DEAR/FEAR, RGM_D/FES to RGM_DES/FES. *Figure 542: MC_RGM block diagram*: removed occurrences of ‘destructive’ reset request and ‘functional’ reset.

Table 953. Document revision history(Continued)

Date	Rev.	Changes
29-May-2014	3	Added *Chapter 44: Boot Assist Flash (BAF)* **SSCM** *Table 605*: Removed MBEO field and description *Figure 583*: Removed footnote 2 *Section 45.3.1*: Removed first 2 paragraphs “The SSCM maintains three... via SSCM parameters.” *Table 611*: Added text “(status)” to column header “LC slot value” *Table 604*: Added reset format “0x0000_0y0y” with note (1) at address 0x0034 (LCSTAT) *Section 45.2.1.1: System Status Register (STATUS)*: Removed the NXEN1 filed from this register **Power Management Controller** *Figure 612: Temperature Sensor Configuration Register (CTL_TD)* reverted back to 5 bits for software compatibility *Figure 610: Reset Event Enable Register (REE_TD)* updated reset value *Figure 604: FCCU Event Enable Register (FEE_HV1)* updated reset value *Figure 602: Reset Event Select Register (RES_HV1)*updated reset value *Figure 594: FCCU Event Enable Register (FEE_LV1)* updated reset value *Figure 596: Reset Event Enable Register (REE_HV0)* updated reset value *Figure 601: Reset Event Enable Register (REE_HV1)* updated reset value Updated*Table 593*, *Table 595*, *Table 598*, *Table 602*, *Table 603*, *Table 604*, *Table 605* for correct VD levels Updated *Figure 600* for HVD13_C *Section 44.1: Introduction*, removed sentence about PMU analog output *Section 44.3: IPS bus interface*, added "except for BIST_TIME10, BIST_TIME32, and BIST_TIME65." in the bytewise bullet. *Section 44.4: Memory map/register definition*, added reference to ME_PCU memory map, added 0x0400 to the value of offset for all registers *Table 610: REE_HV1 field descriptions*, updated description of REE14_AD *Section 44.5.16: Event Pending Register (EPR_HV1)*, added note *Section 44.5.28: Temperature Sensor Configuration Register (CTL_TD)*, changed reset value to 0x0000_0001 and changed the bitfield DOUT_EN in MUX_CTRL *Section 44.6: Temperature Sensor Interface Logic*, supported only one enable (analog) *Figure 605: Supply Gauge Status Register (GR_S)*: Added a note to the reset value

Table 953. Document revision history(Continued)

Date	Rev.	Changes
29-May-2014	3	**Core_e200z0h_Debug_Support** Updated all register figures **Mode Entry Module** *Table 645*: Added rows for reserved spaces and added a table footnote concerning address 0x054 Added rows for reserved spaces and added a table footnote concerning address 0x054 Added registers ME_RUN_PC2–6 at addresses 0x088, 0x08C, 0x090, 0x094 and 0x098 (was reserved) and added ME_LP_PC2–6 at addresses 0x0A8, 0x0AC, 0x0B0, 0x0B4 and 0x0B8 (was reserved) *Table 646* at S_PDO field description and in *Section : Pad Outputs-On*: replaced “slew rate control mechanism” by “power sequence driver” reference to *Table 645* Added *Table 654: ME_TEST_MC field descriptions* Added *Table 655: ME_SAFE_MC field descriptions* Added *Table 656: ME_DRUN_MC field descriptions* Added *Table 657: ME_RUNn_MC field descriptions* Added *Table 658: ME_HALT0_MC field descriptions* *Figure 655*: Set address to 0x078 (was 0x06C) *Figure 656*: Set address to 0x07C (was 0x070) *Figure 658*: Added STANDBY0 with figure footnote 1 *Figure 659*: Listed separated addresses in line with memory map *Section : Peripheral Control Register n (ME_PCTLn)*: Removed text and cross Removed feature bullet text “Additional core clock... ME_CCTL0 and CADDR0 registers *Section : HALT0 mode* and *Section : STOP0 mode*: Removed bullet text “the additional core clocks frozen” *Section 47.4.1*, *Section : Processor Low-Power Mode Entry*, *Section : Processor Clocks Disable*, *Section : System and Processor Clocks Enable*, and *Section : Processor Low-Power Mode Exit* Replaced text “each processor” and “all processors” by “the processor” Replaced all CCTL*n* by CCTL0 Replaced all CADDR*n* by CADDR0

Table 953. Document revision history(Continued)

Date	Rev.	Changes
29-May-2014	3	*Figure 636*: At bit field 5, changed Read and Reset values from '1' to '0' *Figure 640*: Changed bit field 26 to reserved (was M_ICONF_CC) *Figure 641*: Changed bit field 26 to S_MRIG (was reserved) *Figure 643*: Added bit field 26 named S_MRIG and its field description *Figure 645*: Set field PWRLVL to R/W (was Read only) *Section : STOP0 Mode Configuration Register (ME_STOP0_MC)*: add sentence "Lower half word write is allowed for clock configurations." to existing note **Debug and Trace** *Figure 676*: Updated picture *Figure 677*: Removed picture and its introduction paragraph Added *Section 50.2.3* with introduction text and *Table 695* included rows for e200_CORE_0, AUX_BUS_MON_0 and AUX_BUS_MON_1 Added *Section 50.2.4* with introduction text and *Section 50.2.4.1: SPU Level1 Input Mux Configuration registers* included L1SELn register fields and descriptions by added following tables: *Table 696* (n=0, MUX 0 to 7) *Table 697* (n=1, MUX 8 to 15) *Table 698* (n=2, MUX15 to 23) *Table 699* (n=3, MUX 24 to 31) *Table 700* (n=4, MUX 31 to 39) *Table 701* (n=5, MUX 40 to 47) *Table 702* (n=6, MUX 48 to 55) *Table 703* (n=7, MUX 56 to 63) *Table 705*: Set direction of MCKO channel to "Output" signal Updated MUX descriptions from *Table 696*, to *Table 703* **JTAGC** *Section 51.3.3: Device identification register*: changed register reset value (was 0x0AF06041; is 0x000A2041); reorganized bitmap and field descriptions *Table 712: 6-bit JTAG instructions*: – modified ACCESS_AUX_NR_PD instruction summary (was "Enables access to the NAR TAP controller on PD"; is "Enables access to the NAR TAP controller on production device") – changed instruction/summary "ACCESS_AUX_CORE_2 / Enables access to E200 Core 2" to "ACCESS_AUX_CORE_0S / Enables access to e200 Core 0s" Removed all content referencing or describing JCOMP and modified content to describe functionality without JCOMP

Table 953. Document revision history(Continued)

Date	Rev.	Changes
29-May-2014	3	Replaced instruction / instruction summary “ACCESS_AUX_NR_PD / Enables access to the NAR TAP controller on production device” with “ACCESS_AUX_NPC_PD / Enables access to the NPC TAP controller on production device” **JTAGM** Removed “RxCRC receive register” and information related to LFAST from *Section 53.1.1: Overview* and updated this section Removed information related to LFAST communication with from *Section 53.2.2: JTAGM Ready signal* and *Section 53.2.4: JTAGM configuration and status monitoring* In *Section 53.4.1.1: Module configuration register (JTAGM_MCR)*: Updated bits [1:7] from Write only to RW Updated description of , JTAGM_MCR[TCKSEL] register bits Replaced “delay” with “cycle” of inter_jtag_frame_timer field description in *Table 722: JTAGM_MCR register field descriptions* Replaced “jtag_JCOMP” with “jtagm_JCOMP” in *Figure 699: JTAGM_MCR register* Updated “TMS_LOW” field from Read only to RW in *Section 53.4.1.4: Data out register 1 (JTAGM_DOR1)* Updated “TDI_LOW” field from Read only to RW in *Section 53.4.1.6: Data out register 3 (JTAGM_DOR3)* Added note in *Section 53.2.3: EVTO and EVTI signals* In *Table 723: JTAGM_SR register field descriptions*: Reserved bits 9, 11, and 13 of “dci_status” field Added evto0_clr and evto1_clr fields **NPC** Updated *Table 734: PCR field descriptions* and *Figure 700: Port Configuration Register (PCR)* (EVT_EN bit removed) Updated *Figure 711: MSEO transfers (for 1-bit MSEO)* and *Section 54.5.2: Auxiliary output port* (MSEO changed from 2-bits to 1-bit) Updated *Table 731: NPC signal properties* (MSEO reset value changed from 2-bits to 1-bit) Updated *Section 54.2.2: Features* : Removed ‘Control of the device wide debug mode’ Removed censorship status generation *Section 54.4.1.4: Port Configuration Register (PCR)*: Updated the DDR_EN field from read only to R/W. Updated the bit from reserved to “Not functionally used but this bit can be written by software.”

Table 953. Document revision history(Continued)

Date	Rev.	Changes
29-May-2014	3	**SPU** *Table 742*: Added all the Mux register fields from Mux47-Mux32 *Table 743*: Added all the Mux register fields from Mux63-Mux48 *Figure 741*: Added figure note “This bit is writable” to bit 30 field *Table 755*: Removed “PMC2”, “PMC3”, “Valid for cores and PMC” and “For PMC1/2/3/4” at row names NXMC1 and NXMC2 *Figure 729*: Changed Mux 60 by Mux60 *Figure 740*: Straddled “w1c” with straddle R fields *Figure 742*: Reformated register field “ACTION1” to R/W rows and set correct reset (0x0) *Table 746*: Changed field name “ANDInput1” to “FirstANDInput1” *Section 55.5.1.6.5: Trace group configuration registers (DTMTGC, PTMTGC, OTMTGC, WTMTGC)* and *Figure 746*: Renamed DTMGC to DTMTGC Renamed PTMGC to PTMTGC Renamed OTMGC to OTMTGC Renamed WTMGC to WTMTGC **Core_e200z0h_Nexus_Module** DC2 (Development Control register 2) : updated *Figure 753* and *Table 752* Removed registers Client Select Control (CSC) and Port Configuration Register (PCR) Removed Section : Execution mode indication **CRC** *Section 58.2*: “3 hard-wired polynomials (CRC-8, CRC-32 Ethernet, and CRC-16-CCITT)” updated to read “4 hard-wired polynomials (two CRC-8, CRC-32 Ethernet, and CRC-16-CCITT)” *Section 58.2.1*: New feature “CRC-8-H2F Autosar polynomial” added *Table 828*: Updated the bit description of field POLYG for “11” configuration from “Reserved” to “CRC-8H2F autosar polynomial” *Section 58.6*: New equation (Equation 4) added: CRC-8-H2F (Autosar 4.0 protocol) $x^5 + x^3 + x^2 + x + 1$ Added *Section 58.7.2* **FCCU** Replaced all “recoverable faults” with “non critical faults” throughout the chapter Renamed the fields from “IRQENEx” to “IRQENx” in *Figure 888* and *Table 864* Renamed the fields from “NMIENEx” to “NMIENx” in *Figure 889* and *Table 866* Renamed the fields from “EOUTENEx” to “EOUTENx” in *Figure 890* and *Table 897* *Section 60.6.1: FCCU Control Register (FCCU_CTRL)* : Added information about glitch filter

Table 953. Document revision history(Continued)

Date	Rev.	Changes
29-May-2014	3	*Section 60.6.28: FCCU EOUT Signaling Enable registers n (FCCU_EOUT_SIG_ENn)*: Added text “Added information about glitch filter” *Figure 890: FCCU EOUT Signaling Enable register n (FCCU_EOUT_SIG_ENn)*: Removed the note “For non existing fault sources, reset is 0” *Section 60.5.9: EOUT interface*: Replaced text “The EOUT frequency is generated by dividing IRCOSC clock by a fixed factor of 218. With an IRCOSC clock of 16 MHz, this drives a signal of 61 Hz on EOUT” with “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” Added text “Resets (except the power on reset, destructive reset and software bit FCCU_RST in RGM_PRST5) should not influence the state of the failure indication i.e. pins” *Table 863: FCCU_CFG field descriptions* : Added note “FOM bit fields doesn't update to default state on software reset through RGM_PRST5 register” to the FOM field. *Section 60.6.6: FCCU RF Status register n (FCCU_RF_Sn)* : Added a cross reference *Table 890: FCCU_TRANS_LOCK field descriptions* : Updated the TRANSKEY field descriptions Updated *Figure 890: FCCU EOUT Signaling Enable register n (FCCU_EOUT_SIG_ENn)* *Section 60.5.7: FAULT interface*: Added a cross reference to “FCCU Failure input table” in Security chapter. Updated *Section 60.5.6: NMI/WKPU interface* **STCU2** *Table 899: STCU2 register set*,*Table 930: STCU2 register map* and registers bitmaps: updated addresses 0000h->0x0000. *Section 61.2: Main features*: Added text “After completion of On-line Self-Test (MBIST or LBIST) a functional reset is generated. So no partition level functional behavior is guaranteed once On-line Self-Test is triggered.” *Table 901: STCU_SKC field descriptions*: updated value 0000h to 0x000. *Table 902: STCU_CFG field descriptions*: updated sign ‘x’ with ‘*’; completed list for CLK_CFG. *Figure 896: STCU2 PLL Configuration Register (STCU_PLL_CFG)*: added missing borders *Table 904: STCU_WDG field descriptions*: updated value 0000h to 0x000 and sign ‘x’ with ‘*’. *Figure 902: STCU2 Off-line LBIST Status Register (STCU_LBS)*: kept one row only for bits 16-31.

Table 953. Document revision history(Continued)

Date	Rev.	Changes
29-May-2014	3	**REG PROT** Added the line “Also, hard lock bit protection can only be cleared by a system reset once set” in *Section 62.1.2: Features* **PASS** Changed the reset value of bit in *Figure 934: Lifecycle Status register (LCSTAT)* from 0 to 1 unconditionally. Change *Section 63.5.3: Censoring and uncensoring the device* to state that “On delivery the device is censored” instead of uncensored. **WKPU** Updated register figures and field description tables. Added notes referring to the device chapter for *NMI Status Flag Register (NSR)* and *NMI Configuration Register (NCR)* **Acronyms_Abbreviations** *Table 949: Acronyms and abbreviations*: removed term “BAM”

Table 953. Document revision history(Continued)

Date	Rev.	Changes
12-Mar-2015	4	Removed “Chassis & Safety” from the title *Chapter 2: Introduction* Replaced “Dual interrupt controller” by “Interrupt controller (INTC) with dedicated interrupt source channels, including software interrupts and 32 priority levels” *Chapter 4: Signal description* *Table 6: Pin multiplexing description*: added column “Direction” and set all I/O values *Table 7: Peripheral multiplexing description*: for MSCR[712] to MSCR[719], in “Source/Functions” column, added ‘output’ to ‘ETC_x’. New *Table 8: Cross-mapping between pads and package pins* to replace Table ‘eTQFP64 and eTQFP100 pinout’. *Chapter 5: Memory map* *Table 11: Peripheral bridge 1 memory map* and *Table 12: Peripheral bridge 0 memory map*, Replaced: XBAR by XBAR_0, AIPS 0 by PBRIDGE_0, AIPS 1 by PBRIDGE_1, 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, CTU by CTU_0, CRC by CRC_0, DMA Mux by DMAMUX_0, PIT by PIT_0, SAR_ADC_0 by SAR_ADC_12BIT_0, SAR_ADC_B by SAR_ADC_12BIT_B0, Replaced Interrupt controller by INTC and SIUL by SIUL2 *Table 11: Peripheral bridge 1 memory map*: – Added “0xFBFB_0280 | 0xFBFB_02BF | CMU 3 (Peripheral Clock) | - | 19”. – Updated next line with “0xFBFB_02C0 | 0xFBFB_FFFF | Reserved | - | -”. *Table 12: Peripheral bridge 0 memory map*: XBIC value in column “On-Platform Peripheral slot” updated from ‘7’ to ‘6’. – In column “Description”, renamed “ADC” with “SAR_ADC_0” and “ADC seq” with “SAR_ADC_B”. – Updated the start address of SAR_ADC_B from “0xFFE0_04000” to “0xFFE3_C000”. – Updated the end address of SAR_ADC_B from “0xFFE0_07FFF” to “0xFFE3_FFFF”. *Chapter 6: Functional safety* *Figure 6: CPU to Flash write path protection* to *Figure 11: CPU to peripheral read path protection*: Updated pictures *Table 17: FCCU failure inputs*: – Sorted by FCCU input numbers (first column) – Input 3 (PMC) - description: added note – Input 4 (RAM) - description: added note – input 5 (SSCM) - input trigger name: changed to “SSCM_XFER_FLASH_ERR” (was SSCM_XFER_ERR) description: added text “Flash memory initialization error” – Inputs 42 (SSCM): removed row – Inputs 55 (SSCM): removed row – Input 44 (SMPU) - description: changed to SMPU (was MPU) input 55 (SMPU) - description: changed to SMPU protection error (was SMPU monitor error alarm)

Table 953. Document revision history(Continued)

Date	Rev.	Changes
12-Mar-2015	4	Added *Section 6.4.11: LBIST/MBIST architecture partitioning* *Section 6.4.2.2: CPU to Flash read path protection*: replaced EEPROM with “data Flash (data for EEPROM emulation)” *Chapter 7: Device configuration* Renamed *Section 7.7.4.3: eTimer registers* Added figure footnote to *Figure 13: Crossbar switch integration* Updated *Section 7.3.4: Peripheral bridge configuration* Added *Section 7.5.1: Flash memory controller (PFLASH) configuration* Added *Section 7.10.3: Self-Test Control Unit (STCU2) configuration* Updated *Table 25: Reset settings for e200z0h resources* and *Table 26: Special purpose registers*. Updated *Table 38: Interrupt sources* and *Table 40: DMAMUX peripheral DMA request to input source mapping*. Updated *Section 7.3.8: Wakeup unit (WKPU) configuration*. Updated *Table 56: DSPI registers memory map*. Updated *Section 7.10.2.3: Reporting tables implementation* in MEMU configuration section. Updated *Table 70: Reset values of selected FCCU registers*. *Chapter 8: Reset and boot* Modified footnote and signal description of F_SOFT_FUNC module in *Table 85: Functional reset events* updated *Figure 32: System boot sequence* *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 93: BAF boot record structure*: Sorted offsets from lowest (0x0) value to biggest value (0xC). *Chapter 9: Device Configuration Format (DCF) Records* Added note “LVD11_C, LVD11_FL & LVD3_C monitors once enabled in DCF cannot be cleared by DCF record.” below *Figure 39: PMC_REE bits DCF Client* *Table 98: DCF client list*: Updated DCF client description: – Changed STCU_MBCFML[31:0] to STCU_MBUFML[31:0] – Added STCU_LBCFM row at: DCF_CS: 000_0000_0000_0100, DCF address: 000000000010000 – STCU - Updated DCF client description: changed STCU_LBCFM to STCU_LBUFM Removed STCU_LB_ PRPGH_0, STCU_LB_PRPGH_1, STCU_LB_PRPGH_2, STCU_LB_MISRELSW_0 and STCU_LB_MISREHSW_0 rows.

Table 953. Document revision history(Continued)

Date	Rev.	Changes
12-Mar-2015	4	*Figure 38: UTEST Miscellaneous DCF Client* and *Table 99: UTEST Miscellaneous DCF Client field descriptions*: Added field 3p3V_5V at bit 20 with field description “Main segment 3.3V vs 5V indicator - 0 selected power supply is 3.3V - 1 selected power supply is 5V”. Added note to OSC EN_40MHZ description (bit 21). *Figure 39: PMC_REE bits DCF Client*: Set reset value to 0x0000_0000 (was 0xXXXX_0000) *Figure 38: UTEST Miscellaneous DCF Client*: aligned reset values *Table 98: DCF client list*.’ *Chapter 10: Power management* Updated: *Table 104: 1.2 V voltage monitors*, *Table 105: 3.3 V voltage monitors*, *Table 106: 5 V voltage monitors* *Table 107: Voltage monitor features*, int the column “Reset” replaced D with POR. *Chapter 12: e200z0h Core description* Updated e200 to e200z0h wherever needed. Removed autovectored interupt support in *Section 12.2: Microarchitecture summary* Added ‘Nexus Debug Unit’ box in *Figure 48: e200z0h block diagram* Updated *Section 12.4: Features* (no memory array in SPC570Sx) Updated *Section 12.4.3: Load/Store unit features* and *Section 12.5: Core registers and programmer’s model* Updated *Section 12.4.4: e200z0h system bus features* (no cache supported in z0h core) Updated *Figure 49: e200z0h SUPERVISOR mode program model SPRs* (only one register MMUCFG, removed DVC1/2 registers, added footnote to L1CFG0 register) Updated *Figure 50: e200z0h USER mode program model* (added footnote to L1CFG0 register) *Chapter 13: System Integration Unit Lite2 (SIUL2)* *Table 112: SIUL2 memory map*: – Updated latest reserved space to 0x17FF (was 0x17FC) – Added table footnote to MSCR registers – Added table footnote to the offset 0x00C4–0x023F *Figure 64: Multiplexed Signal Configuration Registers for Multiplexed Input Selection (SIUL2_MSCR512–SIUL2_MSCR1023)*: Changed offset from 0XA40–0x123F to 0xA40–0x123C *Table 132: SIUL2_MPGPDOn field descriptions*: Added MASK[x] = 0 and MASK[x] = 1 in field description *Table 130: SIUL2_PGPDOn field descriptions*: Added this table. *Table 131: SIUL2_PGPDIn field descriptions*: Added this table. *Table 112: SIUL2 memory map*: Added two table footnotes. *Figure 52: SIUL2 MCU ID Register #1 (SIUL2_MIDR1)*: Changed reset value to 0x570104XX (was 0x57010400) *Table 113: 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)

Table 953. Document revision history(Continued)

Date	Rev.	Changes
12-Mar-2015	4	*Chapter 14: Crossbar switch (XBAR)* *Figure 74: XBAR Control Register (XBAR_CRSn)*: PARK field changed from 3-bits to 2-bits wide, Set bit 29 as Reserved with figure footnote 1. "This bit is RW and must not be used". New MGPCRn registers added in *Section 14.1.2.3: Master General Purpose Control Registers (XBAR_MGPCRn)* Modified registers in *Section 14.1.2.1: XBAR Priority Registers Slave (XBAR_PRSn)* Modified registers in *Section 14.1.2.2: XBAR Control Register (XBAR_CRSn)* Removed “Priority assignment” and “locked transfer” feature from *Section 14.2.3* *Chapter 15: Crossbar Integrity Checker (XBIC)* *Section 15.1: Overview*: Added text “The attribute information contains: -Several AHB signals such as htrans, hmaster, hwrite, hsize, hprot, hunalign, hburst -decorated attributes 'hdecorated/hdecor' and a note “Decorated attribute signals are all zeros.” *Section 15.2: Features*: Suppressed text bullet “EDC (72,64) code protects against single and double bit errors” *Figure 78: XBIC Module Control Register (XBIC_MCR)*: Deleted footnote at ME2 bit 10 *Section Table 139.: XBIC_MCR field descriptions*: Replaced SEn and MEn by SE0-7 (bits 0 to 7) and ME0-7 (bits 8 to 15) *Figure 80: XBIC Error Status Register (XBIC_ESR)*: Deleted footnote Set 0 to DPME5-7 and DPSE3-7 *Table 141: XBIC_ESR field descriptions*: Replaced DPSEn and DPMEn by DPSE0-4 (bits 1 to 5) and DPME0-2 (bits 9 to 11) *Section 15.5.2.3: XBIC Error Status Register (XBIC_ESR)* and *Section 15.6: Functional description*: Replaced DPSE0-7 and DPME0-7 by DPSE0-4 and DPME0-2 *Section 15.6: Functional description*: Added sentence in the 2nd paragraph “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.” *Chapter 16: Peripheral bridge* *Table 145: Peripheral bridge memory map*: added table_note “Access to this address space will terminate in an error.” to follong rows: 0x004–0x1F, 0x030–0x3F, 0x084–0xFF and 0x130–0x133 *Chapter 17: System Memory Protection Unit (SMPU)* Updated footnote of *Figure 94: SMPU Region Descriptor n – Word 2 Register (SMPU_RGDn_WORD2_FMT0)* *Table 151: SMPU memory map*: Added note “In case of ... pre-fetching in the pipeline).” *Figure 94: SMPU Region Descriptor n – Word 2 Register (SMPU_RGDn_WORD2_FMT0)*: Replaced M2PE, M2SM and M2UM reg fields (bits 14 to 19) by “Reserved” with footnote, added M1UM field, deleted “r w x” in M0UM field. Changed bit 10 of reset value for n=0, from ‘0’ to ‘1’ (n=0 reset value is 0x0061_8600). added figure footnote (1) to Read field from bit 0 to bit 13 *Table 157: SMPU_RGDn_WORD2_FMT0 field descriptions*: Removed M2PE, M2SM and M2UM field and description rows

Table 953. Document revision history(Continued)

Date	Rev.	Changes
12-Mar-2015	4	*Section 17.6.4.4: SMPU Region Descriptor n – Word 3 (SMPU_RGDn_WORD3)*: Replaced "... " by ", then SMPU_RGDn.Word2" in the second paragraph *Table 142: SMPU port allocation*: Suppressed CPU TCM backdoor row *Section 17.6.4.1: SMPU Region Descriptor n – Word 0 (SMPU_RGDn_WORD0)* and *Table 155: SMPU_RGDn_WORD0 field descriptions* description: Suppressed "0-modulo-32 byte" *Section 17.6.4.2: SMPU Region Descriptor n – Word 1 (SMPU_RGDn_WORD1)* and *Table 156: SMPU_RGDn_WORD1 field descriptions* description: Suppressed "31-modulo-32 byte" *Section 17.6.4.3: SMPU Region Descriptor n – Word 2 FMT0 (SMPU_RGDn_WORD2_FMT0)* and *Section 17.6.4.4: SMPU Region Descriptor n – Word 3 (SMPU_RGDn_WORD3)*: Deleted "and always works in Supervisor Mode" in text notes *Figure 96: SMPU RGD Alternate Access Control n (SMPU_RGDAACn)* and *Figure 98: Access evaluation macro critical timing path*: – Added M1UM field and description (bit 23:25) Replaced M2PE, M2SM and M2UM reg fields (bits 14 to 19) by "Reserved" with figure note *Chapter 18: Interrupt Controller (INTC)* *Table 163: INTC memory map*: – Updated with reserved spaces – Added reserved space 0x0004-0x0007 with table footnote *Figure 100: INTC Block Configuration Register (INTC_BCR)* and *Table 164: INTC_BCR field descriptions*: Updated field HVENn to HVEN0. *Section 18.7.1: Initialization flow*: Updated field HVENn to HVEN0, and field VBTAn to VBTA. *Chapter 19: Enhanced direct memory access (eDMA)* Removed "32-byte burst transfer support" in *Section 19.1.1: Features* "100 Reserved" and "101 32-byte burst" replaced by "100 16-byte" and "101 Reserved" in *Table 192: DMA_TCDn_ATTR field descriptions* *Chapter 20: DMA channel multiplexer (DMACHMUX)* *Section 20.1.3: Modes of operation*, in the "Periodic Trigger Mode" added text "for each DMACHMUX instances." Added *Figure 151: DMACHMUX triggered channels* Updated *Section 20.5.2: Enabling and configuring sources* *Chapter 23: Clocking* *Table 225: JTAG frequencies*: updated 'JTAG usable at 1/2 CPU frequency' to 'JTAG usable up to 2 MHz'. *Figure 159: SPC570S dual PLL digital interface block diagram*: updated with detailed PLLs diagrams *Section 23.1: Introduction*: Removed the note about protocol clocks; Removed the reference to the high speed domain; Removed "if backup clock is enabled". Clarified that the internal clocks are generated from both PLLs. Included CMU_3 in *Section 23.2: Clock generation* and *Table 228: Clock input sources*.

Table 953. Document revision history(Continued)

Date	Rev.	Changes
12-Mar-2015	4	Changed the source of the clock out divider in *Figure 158: Clock generation architecture*. *Table 223: MC_CGM relationship to clocks*: Added CGM_SC_DC2 and removed CGM_AC1_DC1 CGM_AC1_SC. Specified the working limitations for the CORE_CLK and PBRIDGE0_CLK in the note that follows *Table 224: Maximum system level clock frequencies*. Reference clock to PLL1 as well as PLL0 in *Section 23.5.2: External oscillator (XOSC)*. Updated bit numbering for MSB=0 in *Section 23.5.2.1: XOSC startup*. Added the precision that a failure is the loss of IRCOSC clock in *Section 23.8.2: Loss of IRCOSC clock*. Updated *Figure 160: Clock distribution* removing reference to DSPI3 *Chapter 24: Dual PLL Digital Interface (PLLDIG)* Updated example in *Section 24.6.4: Frequency modulation* In example *on page 493*: added bullet 'Output divider RFD: 5', updated bullet 'PLL output frequency'. *Chapter 26: Clock Generation Module (MC_CGM)* *Table 247: MC_CGM register description*: Added following registers: CGM_SC_DIV_RC (0x07D0), CGM_DIV_UPD_TYPE (0x07D4), CGM_DIV_UPD_TRIG (0x07D8), CGM_DIV_UPD_STAT (0x07DC), CGM_SC_DC2 (0x07EC). Changed address offset from 0x07E0-0x07FF to 0x07E0-0x07FF (Reserved) Suppressed row at address offset 0x082C (CGM_AC1_DC1) Changed "Reserved" address from 0x0830-0x083F to 0x082C-0x083F from *Section 26.2.1.11: System Clock Divider Ratio Change Register (CGM_SC_DIV_RC)* to *Section 26.2.1.14: Divider Update Status Register (CGM_DIV_UPD_STAT)*: Added register fields and descriptions *Section 26.2.1.18: System Clock Divider 2 Configuration Register (CGM_SC_DC2)*: Added register fields and descriptions Added *Section 26.3.2.2: Divider update triggering* *Figure 203: Auxiliary Clock 0 Divider 2 Configuration Register (CGM_AC0_DC2)*: Change address 0x0000 by 0x0810 *Section 26.3.1.4: System clock dividers*: Added bullet with text: "CLOCKOUT: controlled by the CGM_SC_DC2 register" *Table 282: MC_CGM Example System Clock Division Values Compatible with Divider Synchronization*: Added row for Divider = 2 Added text "System clock divider 2 ... imultiple of 1 and 2" in example 1 Added text "The dividers should be .... greater than 2:1 is not supported." and note "During configuration ... on DMA module." to Caution *Table 246: CGM_SC_DIV_RC field descriptions*: Changed bit 0 field name SYS_UPD_TYPE (was SYS_DIV_RATIO_CHN) Added AUX1_UPD_TYPE (bit 30) and AUX0_UPD_TYPE (bit 31) *Table 259: CGM_DIV_UPD_TYPE field descriptions*: Added fields and descriptions for bit 30 named AUX1_UPD_TYPE and bit 31 named AUX0_UPD_TYPE *Table 248: CGM_DIV_UPD_TRIG field descriptions*: Added AUX1_UPD_STAT (bit 30) and AUX0_UPD_STAT (bit 31)

Table 953. Document revision history(Continued)

Date	Rev.	Changes
12-Mar-2015	4	*Table 261: CGM_DIV_UPD_STAT field descriptions*: Added fields and descriptions for bit 30 named AUX1_UPD_STAT and bit 31 named AUX0_UPD_STAT *Figure 195: System Clock Select Status Register (CGM_SC_SS)*: Changed Reset value of SWTRG field to 100 (was 000). *Figure 196: System Clock Divider 0 Configuration Register (CGM_SC_DC0)*: Added a 0 below bit 11 and bit 12. *Figure 197: System Clock Divider 1 Configuration Register (CGM_SC_DC1)*: Changed Reset value of DIV field to 0001 (was 0000) and added a 0 below bit 11. *Table 264: CGM_SC_DC1 field descriptions*: Changed DIV field description by replacing “CLOCKOUT” by “AIPS clock” *Figure 218: MC_CGM auxiliary clock 1 generation overview*: Removed CGM_AC1_DC1 register and clock divider blocks, and CLOCKOUT output *Figure 214: MC_CGM system clock generation overview*: Added CGM_SC_DC2 register and clock divider blocks, and CLOCKOUT output *Table 265: CGM_SC_DC2 field descriptions*: In columnn “Description“ of DIV field added a space between “divider 2” and “is disabled”. Updated offsets in *Table 247: MC_CGM register description* and throughout the chapter: – 0x07xx -> 0x00xx – 0x08xx -> 0x01xx. *Figure 197: System Clock Divider 1 Configuration Register (CGM_SC_DC1)*: fixed Address. *Table 271: CGM_AC0_DC3 field descriptions: Changed DE field numbering to 0 (was 0:31)* *Table 279: CGM_AC3_SC field descriptions: Added table title* *Chapter 26: IRCOSC digital interface* Removed Temperature Sensor feature and IRCOSC_TT register. *Section 28.3.1.1: IRCOSC control register (IRCOSC_CTL)*: added note “IRCOSC_CTL is writable only in supervisor mode.” to register access. *Figure 223: IRCOSC Native Trimming register (IRCOSC_NT)* and *Table 289: IRCOSC_NT field descriptions*: Reduced field description to FT_CODE and RCTRIM only. *Chapter 27: Platform RAM controller (PRAM)* *Figure 224: Simplified platform block diagram*, added NVM block *Table 290: PRAM controller memory map*: Removed “mode” access column and add “reset value (hex)” and “location” column *Figure 225: Platform RAM configuration register 1 (PRCR1)*: add a figure footnote to bit 26 and 27. *Table 291: PRCR1 field descriptions*: Added note to field FT_DIS. Removed: “Read-modify-write” datapath and associated text. *Section 29.5.1.1: E2EECC considerations on less-than-32-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 titles and borders for *Figure 226* to *Figure 229*.

Table 953. Document revision history(Continued)

Date	Rev.	Changes
12-Mar-2015	4	Added “The PRAM controller” to the third bullet. *Section 29.5.2: Transaction monitor*: Removed Table “RAM safety feature outputs” *Figure 230: RAM controller transaction monitor block diagram*: fixed format issue. *Chapter 28: Flash memory controller (PFLASH Controller)* *Table 293: Flash memory controller memory map*, replaced reset value of PFCR1 with 0x0000_0601 and updated table footnotes. In *Section 30.4.1.3: Platform Flash Configuration Register 3 (PFCR3)* replaced BDRM field by Reserved bit Updated fl_data value in *Section 28.5.12.1.1: e2eEDC and flash accesses* Updated *Figure 231: Platform-centric block diagram with flash memory controller* *Chapter 29: Embedded Flash Memory* Added Flash format addresses mapping with Absolute addresses in *Section 31.3.2.9: Address register (ADR)* *Table 301: Flash User Registers memory map*, updated description of: LOCK0, LOCK1, LOCK2, SEL0, SEL1, SEL2. 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: Analog-to-digital converter (ADC) configuration* Updated *Figure 270: ADC subsystem block diagram* *Chapter 31: Successive Approximation Register Analog-to-Digital Converter (SARADC) Digital Interface* *Table 344: CTR0–CTRL3 field descriptions*: Removed text “for most of the SARADC it is greater than 2” in PRECHG field description. *Chapter 35: Cross Triggering Unit (CTU)* *Section 35.2: CTU overview*: appended to ‘12 inputs’ ‘(EXT_IN0-EXT_IN11)’, updated ‘GPIO — 1 input (EXT_IN15)’ added ‘Refer to ... for description of EXT_IN.’ *Section 35.4.3: ADC results*: – bullet ‘Unsigned right-justified’ updated with ‘Bits[11:0] are used for 12-bit resolution and bits[15:12] always return zero when read.’ – bullet ‘Signed left-justified’ updated with ‘Bits[14:3] are used for 12-bit resolution and bits[2:0] always return zero when read.’ *Section 35.8.1: Trigger Generator Sub-unit Input Selection Register (TGSISR)*: – updated bits 2 to 7 to ‘Reserved’ and added note ‘This bit is not implemented ... no impact.’ – updated field descriptions with EXT_IN0 to EXT_IN11, EXT_IN15. *Figure 310: Cross triggering unit diagram*, *Figure 311: TGS in triggered mode*, *Figure 313: TGS in sequential mode*: replaced input arrows to TGS unit by 13 arrows: EXT_IN0 to EXT_IN11, EXT_IN15. Removed text ‘(as PWM)’ from ‘CTU Clock’.

Table 953. Document revision history(Continued)

Date	Rev.	Changes
12-Mar-2015	4	*Section 35.3.3: TGS in triggered mode*: updated 'The TGS has 16 input signals' by '13'. *Section 35.3.4: TGS in sequential mode*: updated 'The 32 input events (16 signals...' by 'The 26 input events (13 signals...'. *Figure 316: Scheduler subunit*: updated FIFOs box with 4 FIFOs inside, removed 2 input arrows, renamed FIFO_0/1 by ADC_DATA_0/1. *Figure 310: Cross triggering unit diagram*: renamed FIFO_0/1 by ADC_DATA_0/1 *Chapter 34: Enhanced Motor Control Timer (eTimer)* Added Note to *Section 34.1.1: Overview* Removed fault inputs references in *Section 36.2: External signal descriptions* and in *Section 36.7: Interrupts*. Removed *Section 1.2.3: FAULT[n:0] - Fault Inputs* FIE bitfield changed to Reserved conditional tag in *Section 36.4.6.1: Fault Control Register (FCTRL)* *Chapter 38: Software Watchdog Timer (SWT)* Updated reset value in *Figure 382: SWT Counter Output Register (SWT_CO)* *Section 38.4.4: Servicing operations*, removed sentence: (Some devices or instances only support the fixed service sequence and keyed service sequence modes.) *Chapter 39: Periodic Interrupt Timer (PIT)* *Table 461: PIT memory map*: Reformatted Reserved base address offsets to 0x0004-0x00FF. Removed CHN bit field in *Section 39.2.1.4: PIT Timer Control Register n (TCTRLn)* *Chapter 43: Reset Generation Module (MC_RGM)* Added *Section 43.3.2.4: Destructive Bidirectional Reset Enable Register (RGM_DBRE)* and *Section 43.3.2.8: Functional Bidirectional Reset Enable Register (RGM_FBRE)*. *Table 562: MC_RGM memory map*: updated with new registers RGM_DBRE and RGM_FBRE. *Figure 543: Destructive Event Status Register (RGM_DES)*: moved F_SSCM from position 14 to 17. *Figure 544: Destructive Event Reset Disable Register (RGM_DERD)*: moved D_SSCM from position 14 to 17. *Table 567: Functional Event Status Register (RGM_FES) field descriptions*: updated description for F_SOFT_FUNC (bit 28) *Table 568: RGM_FERD register field descriptions*: updated description for D_SOFT_FUNC (bit 28) *Table 571: RGM_FESS register field descriptions*: updated description for SS_SOFT_FUNC (bit 28) *Figure 549: Functional Event Alternate Request Register (RGM_FEAR)*: updated AR_FCCU_SOFT(bit 25) to RO. *Table 569: RGM_FEAR register field descriptions*: added '1' in AR_VOR_FUNC (bit 7) description. *Figure 556: Peripheral Reset Register 2 (RGM_PRST2)* and *Table 576: RGM_PRST2 register field descriptions*: added MCAN_0_2_RAM_RST (bit 21).

Table 953. Document revision history(Continued)

Date	Rev.	Changes
12-Mar-2015	4	*Figure 564: Peripheral Reset Status Register 2 (RGM_PSTAT2)* and *Table 584: RGM_PSTAT2 register field descriptions*: added MCAN_0_2_RAM_STAT (bit 21). *Figure 550*: removed all 'w1c' and made the fields writeable. *Figure 542: MC_RGM block diagram* changed 'FCCU failure to react reset ' into 'FCCU failure detected by FOSU' *Table 567: Functional Event Status Register (RGM_FES) field descriptions* modified the description of F_SOFT_FUNC bitfield *Table 568: RGM_FERD register field descriptions* modified the description of D_SOFT_FUNC bitfield *Table 571: RGM_FESS register field descriptions* modified the description of SS_SOFT_FUNC bitfield *Chapter 44: Boot Assist Flash (BAF)* *Table 593: BAF image header*: updated "Field Description" values for "Address offset" "0000h, 0004h, 0008h" *Chapter 45: System Status and Configuration Module (SSCM)* Updated VLE bitfield definition in *Table 605: STATUS field descriptions* *Section 45.3.2: ECC error monitoring*: Updated paragrah text to "SSCM will request a destructive reset" (was SSCM will request a reset) *Table 613: Resulting Life Cycle*: Removed the 3 latest rows including table footnote (1) and (3). *Chapter 46: Power management controller digital interface (PMC_Dig)* Added *Section 46.5.30: ESR0 Configuration Register (ESR0_CFG)* and its record on *Table 614.: Power Management Controller Memory Map* Register updates: *Table 617: EPR_LV0 field descriptions* Bits 11, 15, 23 and 31 reserved *Table 622: EPR_LV1 field descriptions* Bits 3 and 7 reserved *Table 627: EPR_HV0 field descriptions* Bits 11, 15, 23 and 31 reserved *Table 616: EPR_HV1 field descriptions* Bit 7 reserved Updated the description of bitfields TRIM_ADJ_OVER and TRIM_ADJ_UNDER in *Table 644: CTL_TD field descriptions* in *Table 641: EPR_TD field descriptions* changed the temperature sensor point of TEMP_2 flag (from 150°C to 165°C), and set the temperature sensor point of TEMP_0 flag to -40°C. in *Section 46.5.28: Temperature Sensor Configuration Register (CTL_TD)* removed bitfields TRIM_ADJ_OVER and TRIM_ADJ_UNDER *Chapter 49: e200z0h Core Debug Support* Updated e200 to E200z0h wherever needed. Changed DBCR0-32 into DBCR0-2 in *Section 49.2: Software debug facilities* and in *Section 49.3: Software debug events and exceptions*

Table 953. Document revision history(Continued)

Date	Rev.	Changes
12-Mar-2015	4	*Chapter 48: Mode Entry Module (MC_ME)* *Table 657: MC_ME register description*: Instance of DSPI_3 removed *Figure 637: Interrupt Mask Register (ME_IM)* : Bit 26 added “M_ICONF_CC” *Table 662: ME_IM field descriptions* : Updated with “M_ICONF_CC” field description. *Figure 653: Peripheral Status Register 7 (ME_PS7)*: Bit 29 made Reserved (was earlier S_DSPI_3) *Figure 48.3.2.8: RESET Mode Configuration Register (ME_RESET_MC)* through *Figure 48.3.2.14: STOP0 Mode Configuration Register (ME_STOP0_MC)*: Added note “Half word access is only allowed to the lower half word of the register.” *Figure 649: Peripheral Status Register 2 (ME_PS2)*: Removed 0x1A2 (ME_PCTL226) address offset as periph 226 not exist. *Chapter 50: Debug and Trace* Removed the whole section 1.3: Lock step behavior *Chapter 51: JTAG Controller (JTAGC)* *Figure 686: Device identification register* and *Table 710: Device identification register field descriptions*: update register and table. *Chapter 52: JTAG Data Communication (JDC)* *Table 713: JDC memory map*: Updated cross-references format. *Figure 692: Module Status Register (JDC_MSR)*: Changes JIN_ RDY to JIN_RDY for XML generation. *Chapter 54: Nexus Port Controller (NPC)* Added a footnote to *Figure 710: Port Configuration Register (PCR)* Removed a footnote in *Table 731: NPC signal properties* Added line Number of $\overline{\text{MSEO}}$ pins available in *Table 730: Device parameter values* Changed NEXCFG field by Reserved in *Section 54.4.1.4: Port Configuration Register (PCR)* *Chapter 53: Sequence Processing Unit (SPU)* *Figure 738: Sequence control (SCTRL)* and *Table 751: SCTRL register field descriptions*: Changed fields name SnST2, SnST1 and SnST0 for n=4 to 0 (Was ST2, ST1 and ST0) *Chapter 56: e200z0h Nexus 2+ Module* Extensive updates throughout chapter *Chapter 55: Nexus Crossbar Multi-Master Client (NXMC)* *Figure 798: Development control register 1 (DC1)* : Added footnote “This bit is not implemented and it’s value has no impact on the system” *Figure 799: Development control register 2 (DC2)* : Added footnotes “This bit is not implemented and it’s value has no impact on the system” *Table 817: DC2 register field description*: “Writing bits 2,3,6,7 has no impact and read value will be 0 for these bits.” *Section 57.5: Programming considerations (RESET)*: Added text “after negation of TRST and should be programmed before deassertion of system reset” at the end.

Table 953. Document revision history(Continued)

Date	Rev.	Changes
12-Mar-2015	4	*Chapter 56: Cyclic Redundancy Check (CRC) Unit* Specified the index “n” in all register definition section *Chapter 59: Memory Error Management Unit (MEMU)* Replaced “x” with “*” in all register adresses or offsets and associated memory map Added field length to field name into *Figure 830*, *Figure 832*,*Figure 833: System RAM concurrent overflow registers (SYS_RAM_OFLWn)*, *Figure 835*, *Figure 837*, *Figure 838*, *Figure 840*, *Figure 842*, *Figure 843* *Chapter 58: Fault Collection and Control Unit (FCCU)* *Table 860: FCCU memory map* : The table name updated to “FCCU memory map” from earlier “FCCU register set”. Removed Section “FCCU register map” Removed glitch filter related text of *Section 60.6.1: FCCU Control Register (FCCU_CTRL)* *Section 60.6.14: FCCU NA Freeze Status Register (N2AF_STATUS)*, *Section 60.6.16: FCCU NF Freeze Status Register (N2FF_STATUS)* and *Section 60.6.15: FCCU AF Freeze Status Register (A2FF_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 60.6.17: FCCU FA Freeze Status Register (F2A_STATUS)*, The reset value of field FAFS[24:31] changed from “0” to “w1c”. Adeed *Figure 858: Time switching protocol* *Table 861: FCCU_CTRL field descriptions*: – deleted FILTER_BYPASS and FILTER_WIDTH – in the description of “OPR”, “01000” is made as Reserved *Section 60.5.4: Fault priority scheme and nesting*: “NCT timer” changed to “alarm timer” *Table 890: FCCU_TRANS_LOCK field descriptions*: In field “TRANSKEY”, “COnfiguration unlocked” key value changed from “10111100” to “BCh” *Section 60.5.5: Fault recovery*: in ALARM-> FAULT state, RUN->SAFE and SAFE->RUN transition is removed *Table 863: FCCU_CFG field descriptions*: Description of bit “FOP” and “FCCU_SET_CLEAR” updated.

Table 953. Document revision history(Continued)

Date	Rev.	Changes
12-Mar-2015	4	*Chapter 59: Self-Test Control Unit (STCU2)* *Table 921: 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 903: 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)....” *Table 899: STCU2 register set*, *Table 930: STCU2 register map* : Removed the following registers: – STCU_RUNSW – STCU_LBSSW – STCU_LBESW – STCU_MBSLSW – STCU_MBSMSW – STCU_MBSHSW – STCU_MBELSW – STCU_MBEMSW – STCU_MBEHSW – STCU_LB_RELSW – STCU_LB_REHSW – STCU_LB_REHSW – STCU_LB_RRLSW – STCU_LB_RRHSW *Figure 900: STCU2 Error Register (STCU_ERR_STAT)*: The following bits made Reserved: – Bit 5 (was ABORTHW) – Bit 11 (was LOCKESW) – Bit 12 (was WDTOSW) – Bit 14 (was ENGESW) – Bit 15 (was INVPSW) *Table 899: STCU2 register set*: Added Reserved spaces at the following addresses: 0x0004-0x0007 0x0034-0x003B 0x005C-0x0073 0x0120-0x05FF *Section 61.4.2.2: On-line reset generation (only MBIST)* : – Removed “The On-line execution of the MBIST does not generate any reset from STCU2 at the end of the execution. This is becasue the software application has to manage all the ...” – Added “At the end of the execution, the On-line execution of the MBIST generates a global functional reset.”

Table 953. Document revision history(Continued)

Date	Rev.	Changes
12-Mar-2015	4	*Section 61.7.2: On-line Self-Test Sequence*: – Added “After online selftest completion, a functional reset is triggered in the SoC. In case core tries to access any peripheral/memory during online selftest, access is not guaranteed” Removed “If at least one of the selected LBISTs enables the Global functional reset, the STCU2 requests ...” *Chapter 63: Password and Device Security Module (PASS)* Added new *Table 945: Debug Interface Enable Truth Table* *Figure 940: Password Group n - Lock 2 status register (LOCK2_PGn)* and *Table 942: LOCK2_PGn register field descriptions*: Replaced field L_256LCK by L_128LCK - Lower 128KByte Block Lock”. *Figure 941: Password Group n - Lock 3 status register (LOCK3_PGn)* and *Table 943: LOCK3_PGn register field descriptions*: Added RL4 and RL3 with same description as other RLn. Deleted note in all RLn. added same footnote for all LOCK1_PGn reset *Section 63.4.3: Flash Memory Read Protection*: Replaced “(3 to 5 depending on the device)” by a reference to the “Device configuration” chapter *Figure 941: Password Group n - Lock 3 status register (LOCK3_PGn)*: Added figure footnote to RL1-RL4 and U_256LCK. *Chapter 21: Wakeup unit (WKPU)* Updated *Figure 154: Wakeup unit Block Diagram*. *Section 21.1.1: Overview*: Added text “This NMI external source can also generate a functional reset event to the MC_RGM”. *NMI Status Flag Register (NSR)*, updated reset values and renamed the bit-fields as listed below: – NIF3 → RIF – NOVF3 → ROVF *Section 21.2.2.2: NMI Configuration Register (NCR)*, updated reset values and renamed the bit-fields as listed below: – NDSS3 → RDSS3 – NWRE3 → RWRE3 – NREE3 → RREE3 – NFEE3 → RFEE3 – NFE3 → RFE3 – NLOCK3 → RLOCK – RDSS3 → RDSS – RWRE3 → RWRE – RREE3 → RREE – RFEE3 → RFEE – RFE3 → RFE Removed the NMI pad diagram.
23-Mar-2015	5	Removed “ST Restricted” watermark.

Table 953. Document revision history(Continued)

Date	Rev.	Changes
16-Dec-2015	6	*Chapter 2: Introduction* *Section 2.1: SPC570Sx microcontroller* Replaced ‘computational shell’ by ‘architecture’ in the subsections of this chapter *Section 2.2: Features* Updated the feature list to correspond to the SPC570Sx family *Figure 2: Block diagram*, – Renamed AIPS0 and AIPS1 with PBRIDGE_0 and PBRIDGE_1 – Added CMU_3 in the PBRIDGE_1 block – renamed CMU_PLL with CMU_0 in the PBRIDGE_0 block – Removed ADCSAR_SEQ_0 and ADCSAR_SEQ_B blocks from PBRIDGE_0 block *Chapter 3: Embedded memories* *Section 3.2: SRAM* Updated the size of the general-purpose on-chip ECC SRAM to correspond to the SPC570Sx family *Section 3.3.2: Flash memory array* Updated the size of code Flash to correspond to the SPC570Sx family *Chapter 4: Signal description* *Table 6: Pin multiplexing description*: Column “strength” divided into 4 sub-columns: Very Strong, Strong, Medium and Weak. All “LinFlex” updated to “LinFlexD” For PD[8], Signal name updated to “EXT_TRG” and Full Signal Name updated to “CTU External Trigger” For Instance eTimer_0, Function ETC_0: Source Peripheral updated to “CTU” and Source Function updated to “ETIMER0_TRG”. For Instance eTimer_0, Function ETC_1: Source Peripheral updated to “CTU” and Source Function updated to “ETIMER1_TRG”. For Instance eTimer_1, Function ETC_0: Source Peripheral updated to “CTU” and Source Function updated to “ETIMER0_TRG”. For Instance eTimer_2, Function ETC_0: Source Peripheral updated to “CTU” and Source Function updated to “ETIMER0_TRG”. For Instance eTimer_2, Function ETC_1: Source Peripheral updated to “CTU” and Source Function updated to “ETIMER1_TRG”. For Instance eTimer_3, Function ETC_0: Source Peripheral updated to “CTU” and Source Function updated to “ETIMER0_TRG”. For Instance eTimer_3, Function ETC_1: Source Peripheral updated to “CTU” and Source Function updated to “ETIMER1_TRG”.

Table 953. Document revision history(Continued)

Date	Rev.	Changes
16-Dec-2015	6	*Chapter 5: Memory map* *Table 11: Peripheral bridge 1 memory map*: Replaced “LinFlex 1” by “LINFlexD_1” *Table 12: Peripheral bridge 0 memory map*: Replaced: – “LinFlex 0” by “LINFlexD_0” – “Wakeup” by “WKPU” – “RCOSC dig” by “RC16M_DIG (IRCOSC)” – “XOSC dig” by “OSC40M_DIG (XOSC)” – “PLL” dig” by “PLLDIG” – “CGM” dig” by “MC_CGM” PCU/PMC/Temperature sensor by 2 rows describe with MC_PCU (0xFFFA_0000) and PMCDIG (0xFFFA_0400). Reworked reserved space 0xFFFF_C000–xFFFF_FFFF. Added EIM entry at start address 0xFC00_8000 *Chapter 6: Functional safety* Renamed all occurrences of AIPS as PBRIDGE. *Chapter 7: Device configuration* *Figure 13: Crossbar switch integration*: “AIPS0” and “AIPS1” updated to “PBRIDGE_0” and “PBEIDGE_1” respectively. *Section 7.3.3.1.1: SMPU_0 logical bus master assignments*: Added “On this device, the SMPU (SMPU_0) is attached to the slave side of XBAR_0. The SMPU port allocation is shown in *Table 35: SMPU port allocation*.” *Table 38: Interrupt sources*: – IRQ # 44-47 updated to Platform periodic timer (STM) channel 0, 1, 2 and 3 respectively. The source names also updated to STM_CIR0[CIF] 0, 1, 2 and 3 respectively. – IRQ #477 source description updated: Added User BIST Interrupt – IRQ #477 source name updated: Added “BIST_CTRL[IRQST]” *Section 7.3.9: Crossbar integrity checker (XBIC)* New section *Section 7.7.2: Software Watchdog Timer (SWT) configuration* Feature “Programmable selection of...timer operation” updated to “Internal oscillator... (IRCOSC)” *Section 7.8.3: LINFlexD* Renamed LINFlex to LINFlexD *Section 7.9.1.1: Search boot header and boot options* Replaced redundant text by a cross reference *Section 7.9.2.2: Device-specific features* Added *Table 61: SSCM startup information addresses* *Section 7.10.3: FCCU configuration* *Table 69: FCCU output signals*: Changed Chip-top signal names from F0/1 to FCCU_F0/1 *Table 70: Reset values of selected FCCU registers*: Changed register naming from NCF (Noncritical Fault) to RF (Recoverable Fault)

Table 953. Document revision history(Continued)

Date	Rev.	Changes
16-Dec-2015	6	*Chapter 8: Reset and boot* *Section 8.3: Reset sequence* updated P1, P2 and P3 for FCCU module in *Table 87: Module status during the reset phase* *Section 8.7: System boot*: replaced AIPS with PBRIDGE in both text and *Figure 32: System boot sequence* *Section 8.8: HW boot record search (SSCM search)*: updated text and *Figure 33: Valid boot header search* *Section 8.9: SW boot record search (BAF search)*: updated *Chapter 9: Device Configuration Format (DCF) Records* *Section 9.2: DCF clients* Added the note “Please refer to....Device Datasheet” to the field description of “XOSC LOAD CAP SEL” in *Table 99: UTEST Miscellaneous DCF Client field descriptions* *Chapter 10: Power management* *Figure 41: Power supply scheme*: updated the voltage names *Figure 42: Power segment and voltage monitor architecture*: moved LVD400 *Table 103: Device power management controller external signals*: renamed VDD_HV_PMC_OSC to VDD_HV_OSC_PMC *Table 107: Voltage monitor features*: updated the reset trigger generation for HVD140_C *Section 10.4.7: BIST details*: updated “The timing diagram in *Figure 47* explains the sequence for checking three different monitors: LVD_290C,LVD_290_F_LVD_290A.” *Table 109: Timing descriptions*: updated *Table 110: Calculating the BIST_TIME values*: added *Table 111: Monitors to check*: added *Chapter 11: Security* *Chapter 11.3: Detailed security information*: updated *Chapter 13: System Integration Unit Lite2 (SIUL2)* *Section 13.2.2.2: SIUL2 MCU ID Register #2 (SIUL2_MIDR2)*: replaced field name “PARTNUM” by “FAMILYNUM”, updated footnote (1) to all bits 1:31 and changed the reset value of bits 28:31 to 0xF (was 0x0) in *Figure 53: SIUL2 MCU ID Register #2 (SIUL2_MIDR2)* and *Table 114: SIUL2_MIDR2 field description*. *Section 13.2.2.12: I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR0–SIUL2_MSCR511)*: changed the dimension of the ODC bit field to bits 6:7 (was 5:7) and updatedthe OERC field description with “Specifies the output impedance, drive strength, and slew rate of the associated pin.” in *Figure 63: I/O Pin Multiplexed Signal Configuration Registers (SIUL2_MSCR0–SIUL2_MSCR511)* and *Table 123: SIUL2_MSCR0–SIUL2_MSCR511 field description*. *Chapter 15: Crossbar Integrity Checker (XBIC)* *Chapter Table 138.: XBIC memory map*: removed “Size” and “Access” columns.

Table 953. Document revision history(Continued)

Date	Rev.	Changes
16-Dec-2015	6	*Chapter 16: Peripheral bridge* Replaced all occurences of AIPS with PBRIDGE *Section 16.1.1: Features*: Feature “Supports a pair...memory accesses” updated to “Supports 32 bit transaction”. *Section 16.1.2: General operation*: – Removed all 64-bit read write operation information. – “z0” updated to “e200z0h” *Section 16.3.1: Access support*: Removed text “The peripheral bridge performs … single transfer”. *Chapter 17: System Memory Protection Unit (SMPU)* *Section 17.3: Features*: moved the *SMPU port allocation* table to *SMPU_0 logical bus master assignments*. *Table 151: SMPU memory map*: reformatted the table. *Section 17.6.4.3: SMPU Region Descriptor n – Word 2 FMT0 (SMPU_RGDn_WORD2_FMT0)* added _FMT0 suffix to SMPU_RGDn_WORD2. *Chapter 18: Interrupt Controller (INTC)* *Section 18.4.1: Software vector mode*: “HVEN_PRCn bit” updated to “HVENn bit”. *Section 18.5.2.6: INTC Priority Select Register n (INTC_PSRn)*: changed the implementation of the PRC_SEL bitfield to PRC_SEL*n* where *n* is the number of the processor(s). *Section 20: DMA channel multiplexer (DMACHMUX)* Changed the channel configuration register from CHCONFIG to CHCFG throughout the chapter. *Figure 149: DMACHMUX block diagram*: replaced Trigger #4 by Trigger #3. *Section 20.1.2: Features*: updated the number of channels that provide a trigger functionality. *Chapter 21: Wakeup unit (WKPU)* *Section 21.3.2: Non-Maskable Interrupts*: Number of non-maskable interrupts updated from “4” to “3”. *Table 219: WKPU Memory Map*: – deleted Size column – renamed Use column to Register – deleted the Abbreviation column and moved the abbreviation into the Register column *Chapter 22: Temperature sensor (TSENS)* *Table 222: TSENS calibration constants*: replaced P1, P2, C1, and C2 with P2, C2, P1, and C1 respectivety in note below the table.

Table 953. Document revision history(Continued)

Date	Rev.	Changes
16-Dec-2015	6	*Chapter 23: Clocking* “AIPS” updated to “PBRIDGE” throughout the chapter *Figure 158: Clock generation architecture* updated *Figure 160: Clock distribution*: – Removed the mux betwen RCOSC and safe_clk – Updated all “LinFlex” instances to “LinFlexD” *Table 225: JTAG frequencies*: – For 80MHz clock, internal JTAG frequency updated to 40 MHz. – For 80MHz clock, “JTAG usable upto 2 MHz” updated to “JTAG usable at half CPU frequency”. *Section 23.5.2: External oscillator (XOSC)*: – Text “The XOSC provides...series resistor” updated to “The XOSC...load capacitors”. – Added note “The user must...in the user code”. *Section 23.6.2: System clock output pin*: heading updated. Was initially “System clock” *Chapter 24: Dual PLL Digital Interface (PLLDIG)* Simplified Equation *13*. *Chapter 26: Clock Generation Module (MC_CGM)* “AIPS” updated to “PBRIDGE” throughout the chapter “LinFlex” updated to “LinFlexD” *Chapter 27: OSC Digital Interface (XOSC)* *Table 283: XOSC configurations*: removed ground XTAL and external clock EXTAL configuration *Chapter 29: Platform RAM controller (PRAM)* *Section 29.1: Introduction*: Removed sentence “The second AHB port provides a connection from the flash controller to facilitate calibration overlay access.” *Table 290: PRAM controller memory map*: – Removed “Width” and “Access” columns. – Changed PRCR1 register reset value to 0x0000_0200 (was 0x000002X0). *Section 29.3.1.2: Optional read wait-state*: Removed note about RAM access variations.

Table 953. Document revision history(Continued)

Date	Rev.	Changes
16-Dec-2015	6	*Chapter 30: Flash memory controller (PFLASH Controller)* *Table 293: Flash memory controller memory map*: removed Access and Reset value columns. *Figure 234: Platform Flash Configuration Register 3 (PFCR3)*: updated the note removing “This is a Read only bit”. *Section 30.5.6: Flash error response operation*: added a reference to the Embedded Flash Memory chapter *Section 30.5.7: Censorship*: replaced “TEST flash” by “UTEST flash” *Section 30.5.12.1.1: e2eEDC and flash accesses*: replaced “illegal code value” by the value itself, “FFFF” *Section 30.5.12.2: 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 30.5.13: Array integrity considerations*: removed internal naming “fl_aid” *Chapter 31: Embedded Flash Memory* *Section 31.1.1: Overview*: added ‘bits’ unit to: – the value for reading code partitions – the code Flash memory reads *Section 31.3.2.1: Module Configuration Register (MCR)*: added DWEE field *Chapter 32: Error Injection Module (EIM)* New chapter *Chapter 35: Cross Triggering Unit (CTU)* *Section 35.4: Scheduler subunit (SU)*: In the first bullet list: – replaced “eTimer_0 AUX0” with “eTimer_0/1/2/3 ETC_0”. – replaced “eTimer_1 AUX0” with “eTimer_0/1/2/3 ETC_1”. – added EXT_TRG to External trigger pulse – added the footnote referring to *Table 7: Peripheral multiplexing description* *Section 35.7.2: CTU faults and errors*: Replaced: – “eT0 Trigger Generator” with “eT1 Trigger Generator” – “eT1 Trigger Generator” with “eT2 Trigger Generator” *Chapter 37: System Timer Module (STM)* Removed the register convention table *Chapter 38: Software Watchdog Timer (SWT)* *Section 38.1: Introduction*: Added note referring to the chip configuration information.

Table 953. Document revision history(Continued)

Date	Rev.	Changes
16-Dec-2015	6	*Chapter 41: Deserial Serial Peripheral Interface (DSPI)* *Figure 415: DSPI block diagram (1)*: removed 'Slave Bus Interface' and 'Clock/Reset' *Table 495: DSPI memory map*: Added Reserved space for '0x0004-0x0007', updated 0x0078 by 0x007B. *Section 41.3: Register classification for safety*: removed the footnote "LBIST **not** supported on this device" from the 'Safety Latent' bullet. *Chapter 42: LINFlexD* *Section 42.1: Introduction*: added 2.2 to the list of supported LIN protocol versions. *Chapter 43: Reset Generation Module (MC_RGM)* *Figure 541: MC_RGM block diagram*: replaced "ESR1" by "NMI". *Section 43.1.3: Reset sources*: replaced "ESR1 external reset" by "NMI". *Chapter 44: Boot Assist Flash (BAF)* *Section 44.1: Introduction*: – replaced OSC by IRCOSC – removed the phrase "allowing the 'ASC@CAN' operation" – added the paragraph starting "The MCU is booted…" – added the paragraph starting "The BAF code then checks…" *Section 44.2: 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" *Table 593: BAF memory organization*: replaced 'BAF version number' by 'BAF image header'. *Section 44.2.1: BAF image header*: removed 'The header contains important information such as BAF version.' *Table 594: BAF image header*: updated "Address offset" and "Field Description" values: – 0000h: SSCM boot record tag – 0004h: BAF start address – 0010h: BAF version number *Table 594: BAF image header*: merged Reserved rows. *Section 44.3.1: Boot modes* – In *Application code execution* deleted the sentence "The boot header contains flags that enable/disable individual cores and their reset vectors". *Section 44.3.2: Device initialization*: – replaced "D-MEM" by "System RAM" – added bullet point about BAF using system RAM – added bullet point about functional safety considerations

Table 953. Document revision history(Continued)

Date	Rev.	Changes
16-Dec-2015	6	*Figure 570: Flow of control*: – removed “secure watchdog” – changed ‘Life Cycle check’ to ‘lifecycle is < “IN_FIELD”?’ and the responses to ‘In Field’ and ‘Customer delivery, OEM production’ – removed footnote about serial boot *Section 44.3.4.1: ‘Soft’ DCF clients*: added note after *Table 596: BAF DCF client list* *Section 44.3.4.3: BAF DCF client - DATAX*: – added Caution text – added *Table 598: DATA32 DCF client configuration* – added *Table 599: DATA16 DCF client configuration* – added *Table 600: DATA8 DCF client configuration*. *Section 44.3.4.4: CALLBACK – BAF callback DCF client*: replaced “www.power.org” with “Power Architecture web site“ *Section 44.3.4.5: SER_BOOT_CBACK – BAF serial boot DCF client*: replaced “www.power.org” with “Power Architecture web site“ *Section 44.3.5: Optionally perform a serial boot*: added the sentence “The BAF is executed via the XOSC as clock source” *Figure 575: Flow of control during serial boot*: replaced LINFlex by LINFlexD *Section 44.3.5.1: Serial boot mode protocol*: – added that the download protocol is over FlexCAN and LIN interfaces – bullet 2: added the text about the “NO_ECHO bit” and the note about VLE code – bullet 4: changed CPU0 to ‘boot CPU’ – replaced CAN by FlexCAN *Section 44.3.5.2: LINFlexD download protocol*: – added more detail about the received data echoed by the MCU – added the “NO_ECHO bit” to the third bullet of data order *Section 44.3.5.3: FlexCAN download protocol*:replaced CAN by FlexCAN in the heading *Section 44.3.5.5: Download start address, VLE bit, NO_ECHO bit and code size*: added “NO_ECHO bit” to the heading and description. *Figure 577: VLE bit, NO_ECHO bit and code size*: added the NO_ECHO bit. *Section 44.3.6.1: LINFlexD configuration*: removed “such as LIN or CAN” *Section 44.3.6.2: FlexCAN configuration*:replaced CAN by FlexCAN in the heading *Table 602: LinFlexD_0 and FlexCAN_1 baud rates*: replaced LINFlex by LINFlexD *Section 44.3.6.4: LINFlexD and FlexCAN pin configuration*:replaced CAN by FlexCAN in the heading *Table 603: LINFlexD serial boot mode download protocol*: added “NO_ECHO bit” to the host message sent in step 3 and the BAF responses in steps 3 and 4. *Table 604: FlexCAN serial boot mode download protocol*: added “NO_ECHO bit” to the host message sent and the BAF response in step 2. *Section 44.4: Resources accessed by BAF code execution*: – renamed “clocks” bullet to “magic carpet” – bullet LINFlexD: added text “or from the SER_BOOT_CBACK callback function” – added PFLASH bulletand following text – added SSCM bullet and following text

Table 953. Document revision history(Continued)

Date	Rev.	Changes
16-Dec-2015	6	*Chapter 45: System Status and Configuration Module (SSCM)* *Section 45.1: Introduction*: moved the glossary to the end of the RM *Section 45.3.1: DCF mechanism*: added footnote referencing the Device configuration chapter *Section 45.3.3: Boot mode functionality*: – added 'BOOT_ID = 0x5A' to the bullet point – added precision of boot location for SSCM and BAF *Section 45.3.4.2: Reset Configuration Half-Word*: updated boot sector sentence *Chapter 46: Power management controller digital interface (PMC_Dig)* *Section 46.5.25: Event Pending Register (EPR_TD)*: removed TEMP_2 bit field *Section 46.5.26: Reset Event Enable Register (REE_TD)*: removed TEMP_2 bit field *Section 46.5.27: Reset Event Select Register (RES_TD)*: removed TEMP_2 bit field *Section 46.5.28: Temperature Sensor Configuration Register (CTL_TD)*: removed TS2IE bit field *Section 46.5.28: Temperature Sensor Configuration Register (CTL_TD)*: removed FEE_TS2 bit field *Section 46.5.31: User BIST Flags Register (BIST_FLAGS)*: updated the descriptions of the monitor flags bit by bit for FLAGS_P1 and FLAGS_P2 in *Table 648: BIST_FLAGS field description* *Table 649: BIST_CTRL field descriptions* – removed "It is a R/W0 register" from the description of IRQST – rephrased the description of NCFST – replaced "ipg_hard_async_reset_dest_b" by "reset" – replaced "ipg_clk" by "PBRIDGE0_CLK" *Section 46.5.33: User BIST Time1 and Time0 Register (BIST_TIME10)*, *Section 46.5.34: User BIST Time3 and Time2 Register (BIST_TIME32)* and *Section 46.5.35: User BIST Time6 and Time5 Register (BIST_TIME65)*: – added note "For timing values to configure, see *Section 10.4.7: BIST details*" – replaced "ipg_hard_async_reset_dest_b" by "reset" – replaced "ipg_clk" by "PBRIDGE0_CLK" *Chapter 50: Debug and Trace* *Section 50.4.1.2: Break control*: – replaced "EVTI[1:0] signals from EVTI pads" with "EVTI[0] signal from EVTI pads" – replaced "The $\overline{\text{EVTI}}$[1:0] signals coming from the pads are enabled for debug control using DCI_CR[5:4] bits" with "The $\overline{\text{EVTI}}$[0] signal coming from the pads are enabled for debug control using DCI_CR[4] bit" *Chapter 51: JTAG Controller (JTAGC)* *Table 713: 6-bit JTAG instructions*: updated for cores

Table 953. Document revision history(Continued)

Date	Rev.	Changes
16-Dec-2015	6	*Chapter 54: Nexus Port Controller (NPC)* *Section 54.2.2: Features*: removed “System clock locked status indication via MDO[0] following power-on reset” *Section 54.4.1.4: Port Configuration Register (PCR)*: marked bit 6 and 8 as “Reserved” *Table 736: PCR[MCKO_DIV] values*: added footnote about not using the SYS_CLK/3 setting in DDR mode Removed “System clock locked indication” section *Chapter 60: Fault Collection and Control Unit (FCCU)* Used recoverable fault (RF) instead of noncritical fault (NCF) throughout the chapter *Section 60.2: Features*: – removed phrases about the self-checking procedure – moved the glossary to the end of the RM *Figure 844: FCCU block diagram*: replaced “ipg_clk” by “System clock” *Section 60.5.2: FSM description*: – added “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” – changed “an interrupt (irq_alarm)” to “an interrupt request (ALARM state)” *Table 855: Input reset*: – replaced ipg_hard_asynch_rst_clk_b by “System reset” – replaced ipg_hard_asynch_rst_clk_safe_b by “System reset (RC oscillator)” *Table 856: Reset reactions*: – replaced irst_lfunc_b by “Long functional reset” – replaced rst_sfunc_b by “Short functional reset” *Table 857: Reset sources*: – replaced ipg_hard_asynch_rst_*_b by “System reset signal” *Table 858: MC actions*: – replaced irst_lfunc_b by “Long functional reset” – replaced rst_sfunc_b by “Short functional reset” *Section 60.5.5: Fault recovery* – removed SAFE state – *Figure 848: RF(ALARM -> FAULT state) recovery*: removed SAFE state and renamed FCCU_F to “FCCU fault pin state” *Section 60.5.6: NMI/WKPU interface* – replaced nmi_wkup_b by “Nonmaskable interrupt” – removed “(ipg_clk)” and “(nmi_b)” – replaced wkup by “Wake-up indication” – replaced mc_state by “MC state” – replaced sw_mode_change_req by “MC mode SW change request” *Section 60.5.7: FAULT interface* – removed “(fccu_ncf_clr[])” – replaced fccu_ncf_clr[] by “FCCU recoverable faults clear” – replaced “FCCU_CF_Sx/FCCU_UF_Sxor FCCU_NCF_Sx” by FCCU_RF_Sn *Figure 850: FAULT injection* and *Figure 851: FAULT clearing* – replaced fccu_ncf[y] by “FCCU recoverable fault y” – replaced fccu_ncf_set[y] by “FCCU recoverable fault clear y”

Table 953. Document revision history(Continued)

Date	Rev.	Changes
16-Dec-2015	6	*Section 60.5.8: STCU interface* – replaced fccu_eout_msk by “EOUT mask” *Section 60.5.9: EOUT interface* – replaced fccu_eout by “FCCU Error output” – replaced ERROR/error (states) by FAULT *Section 60.5.9.1: Dual rail protocol*: removed “with a given frequency of 61 Hz” *Section 60.5.9.2: Time switching protocol*: – removed “with frequency of 61 Hz” – replaced fccu_eout by “FCCU Error output pin” *Section 60.5.9.3: Bi-stable protocol*: – replaced fccu_eout by “FCCU Error output” *Section 60.5.10: Error signal flow for RF case*: – replaced FCCU_NCF_E by FCCU_RF_E – replaced FCCU_NCF_TOE by FCCU_RF_TOE *Figure 861: Error signal flow* – renamed all signals containing NCF by RF *Table 863: FCCU memory map*: changed register names from NCF (noncritical fault) to RF (recoverable fault): – FCCU_NCF_CFG0 → FCCU_RF_CFG0 – FCCU_NCF_CFG1 → FCCU_RF_CFG1 – FCCU_NCF_CFG2 → FCCU_RF_CFG2 – FCCU_NCF_CFG3 → FCCU_RF_CFG3 – FCCU_NCFS_CFG0 → FCCU_RFS_CFG0 – FCCU_NCFS_CFG1 → FCCU_RFS_CFG1 – FCCU_NCFS_CFG2 → FCCU_RFS_CFG2 – FCCU_NCFS_CFG3 → FCCU_RFS_CFG3 – FCCU_NCFS_CFG4 → FCCU_RFS_CFG4 – FCCU_NCFS_CFG5 → FCCU_RFS_CFG5 – FCCU_NCFS_CFG6 → FCCU_RFS_CFG6 – FCCU_NCFS_CFG7 → FCCU_RFS_CFG7 – FCCU_NCF_S0 → FCCU_RF_S0 – FCCU_NCF_S1 → FCCU_RF_S1 – FCCU_NCF_S2 → FCCU_RF_S2 – FCCU_NCF_S3 → FCCU_RF_S3 – FCCU_NCFK → FCCU_RFK – FCCU_NCF_E0 → FCCU_RF_E0 – FCCU_NCF_E1 → FCCU_RF_E1 – FCCU_NCF_E2 → FCCU_RF_E2 – FCCU_NCF_E3 → FCCU_RF_E3 – FCCU_NCF_TOE0 → FCCU_RF_TOE0 – FCCU_NCF_TOE1 → FCCU_RF_TOE1 – FCCU_NCF_TOE2 → FCCU_RF_TOE2 – FCCU_NCF_TOE3 → FCCU_RF_TOE3 – FCCU_NCF_TO → FCCU_RF_TO – FCCU_NCFF → FCCU_RFF

Table 953. Document revision history(Continued)

Date	Rev.	Changes
16-Dec-2015	6	*Table 864: FCCU_CTRL field descriptions*: – replaced “ipg_debug” by “Debug” in the DEBUG field description *Table 866: FCCU_CFG field descriptions*: – replaced “fcc_eout” by “FCCU Error output” in the PS and FOM field descriptions – replaced “ipg_clk_safe” by “RC oscillator” in the FOP field description *Figure 867: FCCU RF Status register n (FCCU_RF_Sn)* Changed the reset value from 0x0 to “The reset value is chip-specific; see the chapter that describes how modules are configured and connected.” *Figure 868: FCCU RF Key register (FCCU_RFK)*: updated access to “User Write-only” *Section 60.6.12: FCCU IO Control Register (FCCU_EINOUT)*: – replaced “fcc_eout” by “FCCU Error output” *Figure 873: FCCU IO Control Register (FCCU_EINOUT)*: – replaced “fccu.fccu_ein” by “FCCU Error input” in footnote 1 – added footnote 2 (originally footnote to *Table 881: FCCU_EINOUT field descriptions*) *Table 881: FCCU_EINOUT field descriptions*: replaced “fccu.fccu_ein” by “FCCU Error input” *Section 60.6.13: FCCU Status Register (FCCU_STAT)*: removed “ipg_clk_safe” and “ipg_clk” *Table 883: N2AF_STATUS field descriptions*:replaced “fccu_ncf” by “FCCU recoverable fault” *Table 884: A2FF_STATUS field descriptions*:replaced “fccu_ncf” by “FCCU recoverable fault” *Section 60.6.16: FCCU NF Freeze Status Register (N2FF_STATUS)*: replaced “fccu_uf[x]” by “recoverable fault” *Table 885: N2FF_STATUS field descriptions*:replaced “fccu_ncf” by “FCCU recoverable fault” *Section 60.6.17: FCCU FA Freeze Status Register (F2A_STATUS)*: replaced “fccu_uf[x]” by “recoverable fault” *Table 886: F2AF_STATUS field descriptions*:replaced “fccu_ncf” by “FCCU recoverable fault” *Figure 879: FCCU RF Fake Register (FCCU_RFF)*: updated access to “User Write-only” *Section 60.6.19: FCCU IRQ Status Register (FCCU_IRQ_STAT)*: replaced “irq_misc” by “(interrupt request)” *Figure 880: FCCU IRQ Status Register (FCCU_IRQ_STAT)*: – added footnotes 1 and 2 (originally footnotes to *Table 888: FCCU_IRQ_STAT field descriptions*) *Section 60.6.20: FCCU IRQ Enable Register (FCCU_IRQ_EN)*: replaced “irq_misc” by “interrupt request (miscellaneous conditions)” *Section 60.6.21: FCCU XTMR Register (FCCU_XTMR)*: updated the phrase about ETMR *Figure 884: FCCU Transient Register (FCCU_TRANS_LOCK)*: updated access to “Supervisor Write-only” *Figure 885: FCCU Permanent Lock Register (FCCU_PERMNT_LOCK)*: updated access to “Supervisor Write-only”

Table 953. Document revision history(Continued)

Date	Rev.	Changes
16-Dec-2015	6	*Figure 886: FCCU Delta T Register (FCCU_DELTA_T)*: – added footnote 1 (originally footnote to *Table 895: FCCU_DELTA_T field descriptions*) *Section 60.7: FCCU Output Supervision Unit*: added “This signal is automatically deasserted ...in CONFIG state.” *Section 60.8.1: Misconfigurations*: updated point 5 *Chapter 61: Self-Test Control Unit (STCU2)* *Table 902: STCU2 register set*: added the footnote *See LBIST and NMCUT values in Table 98: DCF client list of Chapter 9: Device Configuration Format (DCF) Records.* *Section 61.5.2: STCU2 SK Code Register (STCU_SKC)*: updated the phrase about invalid access or sequence *Chapter 62: Register Protection (REG_PROT)* *Section 62.3.2: Register descriptions*: removed the “Key to Register Fields” figure

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.

© 2015 STMicroelectronics – All rights reserved